

# **Design und Implementierung eines Systems zur schnellen Rekonstruktion dreidimensionaler Modelle aus Stereobildern**

**Inauguraldissertation  
zur Erlangung des akademischen Grades  
eines Doktors der Naturwissenschaften  
der Universität Mannheim**

**vorgelegt von  
Dipl.-Ing. Karsten Mühlmann  
aus Bonn**

**Mannheim, 2002**



Dekan: Professor Dr. Herbert Popp, Universität Mannheim  
Hauptreferent: Professor Dr. Reinhard Männer, Universität Mannheim  
Korreferent: Professor Dr. Bernd Jähne, Universität Heidelberg

Tag der mündlichen Prüfung: 6. August 2002

---

# Inhaltsverzeichnis

<b>1</b>	<b>Einleitung</b>	<b>3</b>
1.1	Motivation . . . . .	4
1.2	Aufbau der Arbeit . . . . .	5
<b>I</b>	<b>Stand der Technik</b>	<b>7</b>
<b>2</b>	<b>Verfahren zur optischen 3D-Rekonstruktion</b>	<b>9</b>
2.1	Aktive Verfahren . . . . .	9
2.1.1	Time-Of-Flight Verfahren . . . . .	10
2.1.2	Laserscanner . . . . .	10
2.1.3	Strukturiertes Licht . . . . .	11
2.2	Passive Verfahren . . . . .	12
2.2.1	„Shape From Shading“ . . . . .	12
2.2.2	Volumenrekonstruktion . . . . .	13
2.2.3	Stereo-Rekonstruktion . . . . .	14
<b>3</b>	<b>Komplettsysteme zur schnellen Stereo-Rekonstruktion</b>	<b>17</b>
3.1	Forschungsprojekte . . . . .	17
3.1.1	CMU Stereo Machine . . . . .	18
3.1.2	PARTS . . . . .	18
3.2	Kommerziell erhältliche Systeme . . . . .	18
3.2.1	Small Vision System . . . . .	18
3.2.2	Triclops . . . . .	19
3.2.3	DeepSea . . . . .	19
<b>II</b>	<b>Hardware</b>	<b>21</b>
<b>4</b>	<b>Stereokamera</b>	<b>23</b>

4.1	Kamerablock . . . . .	25
4.2	Stereoendoskop . . . . .	26
<b>5</b>	<b>Stereoframegrabber</b>	<b>27</b>
5.1	FPGA Karte „microEnable“ . . . . .	28
5.2	Field Programmable Gate Arrays . . . . .	29
5.3	Video Analog-Digital Wandler . . . . .	31
5.4	Hardwarebeschreibung mit CHDL . . . . .	34
5.4.1	Schaltungsentwurf und -synthese . . . . .	34
5.4.2	Simulation . . . . .	36
5.4.3	Realisierung des Framegrabbers . . . . .	38
<b>III</b>	<b>Software</b>	<b>43</b>
<b>6</b>	<b>Kalibrierung</b>	<b>45</b>
6.1	Kameramodell . . . . .	45
6.1.1	Interne Parameter . . . . .	45
6.1.2	Externe Parameter . . . . .	47
6.1.3	Projektionsmatrix und ihre Zerlegung . . . . .	48
6.2	Direkte Lineare Kalibrierung . . . . .	49
6.2.1	Exakte Lösung . . . . .	50
6.2.2	Überbestimmtes System . . . . .	50
6.2.3	Normalisierung . . . . .	51
6.2.4	Minimierung des Reprojektionsfehlers . . . . .	52
6.2.5	Praxistauglichkeit . . . . .	52
6.3	Verfahren von Tsai . . . . .	53
6.3.1	Kalibriermuster . . . . .	53
6.3.2	Linsenverzerrung . . . . .	54
6.3.3	Ablauf der Kalibrierung . . . . .	55
6.3.4	Der unbekannte Skalierungsfaktor . . . . .	56
6.3.5	Eignung für die Kalibrierung der Stereokamera . . . . .	56
6.4	Verfahren von Zhang . . . . .	57
6.4.1	Kalibriermuster . . . . .	57
6.4.2	Aufstellen der Bestimmungsgleichung . . . . .	58
6.4.3	Bestimmung der internen Parameter . . . . .	59
6.4.4	Bestimmung der externen Parameter . . . . .	59
6.4.5	Linsenverzerrung . . . . .	60
6.4.6	Maximum Likelihood Schätzung . . . . .	60
6.5	Stereokalibrierung . . . . .	61
<b>7</b>	<b>Bildvorverarbeitung</b>	<b>65</b>

7.1	Epipolargeometrie und Fundamentalmatrix . . . . .	65
7.2	Rektifikation . . . . .	67
7.2.1	Geometrisches Verfahren . . . . .	67
7.2.2	Algebraisches Verfahren . . . . .	67
7.2.3	Algorithmus . . . . .	69
7.3	Implementierung der Entzerrung . . . . .	71
<b>8</b>	<b>Disparitätsberechnung</b>	<b>75</b>
8.1	Disparität . . . . .	75
8.2	Disparitätsvolumen . . . . .	76
8.3	Ähnlichkeitsmaß . . . . .	77
8.4	Speicherorganisation . . . . .	79
8.4.1	$x, y, d$ . . . . .	79
8.4.2	$x, d, y$ . . . . .	80
8.4.3	$d, x, y$ . . . . .	80
8.5	Aufsummieren der Fenster . . . . .	80
8.6	Minimumsuche . . . . .	82
8.7	Eindeutigkeitstest . . . . .	85
8.8	Subpixelgenauigkeit . . . . .	86
8.9	Medianfilter . . . . .	88
8.10	Algorithmus . . . . .	88
<b>9</b>	<b>3D-Modell Erstellung</b>	<b>91</b>
9.1	Berechnung der 3D-Punkte . . . . .	91
9.2	Generierung eines Dreiecksnetzes . . . . .	93
<b>IV</b>	<b>Ergebnisse</b>	<b>97</b>
<b>10</b>	<b>Kalibrierung</b>	<b>99</b>
10.1	Reprojektionsfehler . . . . .	99
10.2	Wiederholgenauigkeit . . . . .	100
<b>11</b>	<b>Stereoalgorithmus</b>	<b>105</b>
11.1	Bildentzerrung . . . . .	105
11.2	Disparitäten . . . . .	105
11.3	Laufzeiten . . . . .	108
11.4	Medianfilter . . . . .	116
<b>12</b>	<b>Gesamtssystem</b>	<b>119</b>
<b>13</b>	<b>Diskussion, Ausblick und Zusammenfassung</b>	<b>125</b>
13.1	Diskussion . . . . .	125

*Inhaltsverzeichnis*

---

13.2 Ausblick . . . . .	126
13.3 Zusammenfassung . . . . .	127
<b>Literaturverzeichnis</b>	<b>135</b>

# Formelzeichen und verwendete Abkürzungen

## Formelzeichen

Skalare sind klein und kursiv gesetzt, ausgenommen die Komponenten von Weltkoordinaten, die groß und kursiv sind. Kleine fette Buchstaben bezeichnen Vektoren, große fette Matrizen.

Einfach gestrichene Größen gehören zur „jeweils anderen“ Kamera oder zum „jeweils anderen“ Bild.

$\mathbf{x} = (x, y)^T$	inhomogene 2D-Koordinaten
$\mathbf{x} = (x, y, w)^T$	homogene 2D-Koordinaten
$\mathbf{X} = (X, Y, Z)^T$	inhomogene 3D-Koordinaten
$\mathbf{X} = (X, Y, Z, W)^T$	homogene 3D-Koordinaten
$\mathbf{P}$	Projektionsmatrix
$\mathbf{K}$	interne Kalibriermatrix
$\omega$	image of the absolute conic (IAC)
$f$	Brennweite in Einheiten der Weltkoordinaten
$f_x, f_y$	Brennweiten in Pixeln
$p_x, p_y$	Hauptpunktkoordinaten
$s$	Schiefe
$\mathbf{R}$	Rotationsmatrix
$\mathbf{t}$	Translationsvektor
$\mathbf{C}$	Projektionszentrum
$\kappa_1, \kappa_2$	radiale Koeffizienten der Linsenverzerrung
$\rho_1, \rho_2$	tangentiale Koeffizienten der Linsenverzerrung
$\mathbf{H}$	planare Homographie
$\mathbf{e}$	Epipol
$\mathbf{F}$	Fundamentalmatrix
$d$	Disparität

## Abkürzungen

2D	zweidimensional
3D	dreidimensional
CCD	Charge Coupled Device
CHDL	C++ based Hardware Description Language
CLB	Configurable Logic Block
DMA	Direct Memory Access
DLT	Direct Linear Transform
DSP	Digital Signal Processor
FIFO	First In First Out
FPGA	Field Programmable Gate Array
GUI	Graphical User Interface
IEEE	Institute of Electrical and Electronic Engineers
I <sup>2</sup> C	Inter Integrated Circuits
LCD	Liquid Crystal Display
LVDS	Low-Voltage Difference Signal (IEEE 1596.3)
MEG	Magnetoenzephalographie
MMX	Multi-Media Extension
NTSC	National Television System Committee (aka Never Twice Same Color)
OpenCV	Open Computer Vision Library
OpenGL	Open Graphics Library
PAL	Phase Alternation Line
PCI	Peripheral Component Interconnect
RDRAM	Rambus Dynamic Random Access Memory
RGB	Rot, Grün, Blau
SAD	Sum of Absolute Differences
SDRAM	Synchronous Dynamic Random Access Memory
SIMD	Singe Instruction Multiple Data
SRAM	Static Random Access Memory
SSD	Sum of Squared Differences
SSE	Streaming SIMD Extensions
SVD	Singular Value Decomposition
S-VHS	Super Video Home System
VRML	Virtual Reality Modeling Language

# 1 Einleitung

Digitale Bilderfassung und -bearbeitung sind zu einem Massenmarkt geworden. Von den fallenden Preisen und der steigenden Qualität der Sensoren profitieren auch die Bildverarbeitung und ihre Anwender. Der Preisverfall und der direkte Zugang zur Bildinformation sorgen dafür, daß Kameras an immer mehr Stellen als Sensoren eingesetzt werden. Seit Farbsensoren in großer Zahl verfügbar sind, wird dabei zunehmend auch Farbe als ein wichtiger Träger von Information berücksichtigt. Durch die gleichzeitig gewachsene Rechen- und Speicherkapazität der eingesetzten Computer lassen sich sowohl die anfallenden großen Datenmengen verarbeiten als auch aufwendigere Algorithmen einsetzen.

Zu den Verfahren, bei denen sehr viel Rechenleistung benötigt wird, gehört auch die 3D-Rekonstruktion. Sie beschäftigt sich damit, die bei der Projektion einer dreidimensionalen Szene in zweidimensionale Bilder verloren gegangene Information über die Tiefe wieder herzustellen. Bei der Stereo-Rekonstruktion geht man von zwei Bildern der selben Szene aus, die von unterschiedlichen Positionen aus aufgenommen wurden. Aus den Unterschieden in den Bildern kann auf die Tiefe der einzelnen Objekte geschlossen werden und man erhält ein dreidimensionales Modell, das ihre Form, Lage und Position wiedergibt. Auf diese Weise lassen sich in sehr kurzer Zeit Informationen über das aufgenommene Objekt gewinnen, die sich zum Beispiel durch mechanische Meßverfahren nur sehr schwer erfassen lassen.

Während die Vermessung mit Hilfe von herkömmlichen Fotografien, die klassische Photogrammetrie, sehr aufwendig und teuer ist und nur die Position einzelner Punkte liefert, erlaubt der Einsatz von digitalen Aufnahmen die automatische Rekonstruktion sehr detailreicher und vollständiger Modelle. Die Modelle sofort im Rechner verfügbar zu haben und weiterverarbeiten zu können, macht einen großen Teil der Attraktivität dieser Verfahren aus.

## 1.1 Motivation

Auslöser für den Beginn dieser Arbeit war ein Problem, das bei der Magnetoenzephalographie (MEG) auftritt. Aus einer Messung des Magnetfeldes an der Kopfoberfläche wird dabei auf die Position und Stärke von Aktivität in Form von Stromdipolen im Gehirn geschlossen. Um Magnetfelder messen zu können, die um ein vielfaches schwächer als das Erdmagnetfeld sind, müssen bestimmte Voraussetzungen gegeben sein. Die Messung findet in einem Raum statt, der alle Magnetfelder von außen abschirmt und in dem keine Störfelder durch elektrische Geräte vorkommen dürfen. Desweiteren sind die Meßspulen supraleitend und entsprechend stark gekühlt, was bedeutet, daß sie nicht fest mit der Kopfoberfläche verbunden werden können. Der Kopf des Patienten befindet sich unter einem Helm, der die Spulen enthält. Er kann seine Position verändern, entweder unmerklich während der mehrere Minuten dauernden Messung, oder durch Schluck- und Atembewegungen.

Ist man in der Lage, die Kopfposition während der Messung aufzuzeichnen, kann die Genauigkeit der Quellenlokalisierung verbessert werden, indem das bei der Berechnung verwendete Kopfmodell die Bewegung nachvollzieht. Um die allmähliche Bewegung zu erfassen, reicht es aus, im Abstand von wenigen Sekunden zu messen. Für die schnellen Bewegungen ist nur eine Detektion notwendig, die die entsprechenden Zeitabschnitte automatisch kennzeichnet oder aus der Rekonstruktion ausblendet.

Die Messung soll passiv erfolgen, um im visuellen Kortex keine Signale durch aktive Beleuchtung oder beim Patienten zusätzlichen Streß zu verursachen. Der Kopf des Patienten wird dazu mit einer Stereokamera aufgenommen, die sich außerhalb der Kammer befindet, damit sie nicht abgeschirmt werden muß. Aus den Stereobildern soll ein dreidimensionales Modell des Kopfes und daraus seine Lage und Orientierung berechnet werden.

Bei einer zweiten potentiellen Anwendung kommt es mehr auf die Geschwindigkeit als auf die Komplettheit der Modelle an. Abschnitt 4.2 stellt ein Stereo-Endoskop vor, mit dem Laparoskopien bei Tieren durchgeführt werden. Mit ihm soll untersucht werden, ob ein operationsbegleitend rekonstruiertes 3D-Modell die Orientierung erleichtern kann und ob mit dem Endoskop Messungen an Organoberflächen durchgeführt werden können.

Es ist sinnvoll, die 3D-Rekonstruktion von diesen speziellen Anwendungen zu trennen. Auf diese Weise entstand die Idee, ein allgemeines System zur Rekonstruktion von Oberflächen zu entwickeln, das unter anderem in den eben beschriebenen Szenarien einsetzbar sein soll. Die hier vorgestellten Algorithmen fließen in eine Software-Bibliothek ein, die nicht nur die Rekonstruktion aus einem kalibrierten

Stereobildpaar beinhaltet, sondern auch die Berechnung kompletter 3D-Modelle aus unkalibrierten Bildsequenzen.

## 1.2 Aufbau der Arbeit

Die Arbeit gliedert sich in vier Teile. Im ersten Teil gibt Kapitel 2 einen Überblick über die Verfahren, die zur berührungslosen Oberflächenerfassung verwendet werden können. Kapitel 3 stellt vergleichbare Projekte zur schnellen 3D-Rekonstruktion, ihre Leistungsdaten und Grenzen vor.

Der zweite Teil der Arbeit behandelt die Entwicklung der Hardware, aufgeteilt in Kapitel 4, Stereokamera, und Kapitel 5, Framegrabber.

Im dritten Teil werden die Softwarekomponenten beschrieben. Die Kalibrierung in Kapitel 6 dient der Modellierung der Kameras. Durch Rektifikation (Kapitel 7), eine Vorverarbeitung der Bilder, kann die Disparitätsberechnung (Kapitel 8), die Suche nach korrespondierenden Bildpunkten, sehr effizient implementiert werden. Das letzte Software-Kapitel beschreibt die Berechnung einer dreidimensionalen Oberfläche aus den Ergebnissen der Kalibrierung und der Disparitätsberechnung.

Die Präsentation der Ergebnisse der einzelnen Komponenten schließt sich im vierten Teil an. Kapitel 10 mißt die Genauigkeit der Kalibrierung, da sie die Grundlage für die Genauigkeit des Gesamtsystems bildet. Kapitel 11 widmet sich den berechneten Disparitäten, vorrangig der Geschwindigkeit verschiedener Implementierungen. Wie aus den einzelnen Komponenten ein Komplettsystem wird, mit dem mehrere 3D-Modelle pro Sekunde rekonstruiert werden können, beschreibt Kapitel 12.

Eine Diskussion der Ergebnisse, ein Ausblick und die Zusammenfassung schließen die Arbeit ab.



**Teil I**

**Stand der Technik**



# 2 Verfahren zur optischen 3D-Rekonstruktion

Dieses Kapitel stellt gebräuchliche Verfahren vor, die die Form und Lage von Objekten optisch erfassen können. Für spezielle Anwendungsgebiete existieren eine Vielzahl anderer Meßprinzipien, die jedoch nicht betrachtet werden, da sie von vorneherein für die geplanten Applikationen ausgeschlossen werden können. Eine detaillierte Klassifikation optischer Verfahren findet sich in [34]. Die Wahl fiel aus mehreren Gründen auf die Verwendung eines optischen Verfahrens. Erstens läßt sich damit ein ungleich größeres Spektrum an Anwendungen abdecken als zum Beispiel mit mechanischen Verfahren, zweitens werden bei der Endoskopie bereits Kamerabilder verwendet und drittens stellte alles andere als eine berührungslose, unkomplizierte Meßvorrichtung beim MEG eine erhebliche zusätzliche psychische Belastung für den Patienten dar.

Die existierenden Verfahren lassen sich anhand mehrerer Merkmale in Klassen einteilen. Hier wurde die Unterscheidung zwischen aktiven und passiven Verfahren gewählt. Aufgrund ihrer Eigenschaften sind in beiden Kategorien diejenigen Verfahren am interessantesten, die nach dem Prinzip der Triangulation arbeiten. Während die aktiven die Interaktion der Objektoberfläche mit dem vom Scanner ausgesandten Licht registrieren, beschränken sich die passiven auf die Aufnahme von Bildern und nutzen die natürliche Beleuchtung der Szene.

## 2.1 Aktive Verfahren

Aktive Verfahren arbeiten mit einer eigenen Lichtquelle, die durch die Beleuchtung einen Teil des Objektes hervorhebt. Das Objekt wird dabei von einem Sensor be-

obachtet, bei dem es sich je nach Verfahren um eine Photozelle oder eine Kamera handeln kann. Der Hauptvorteil aktiver Verfahren ist, daß durch die kontrollierte Beleuchtung Mehrdeutigkeiten ausgeschlossen werden, die den passiven Verfahren Probleme bereiten. Je kleiner der Teil des Objektes ist, der beleuchtet wird, desto länger dauert es bis der Scanvorgang alle Teile erfaßt hat. Während dieser Zeit, die bei manchen Verfahren im Sekunden- oder gar Minutenbereich liegen kann, darf sich das Objekt natürlich nicht bewegen. Die zusätzlich zu den Kameras benötigten optischen und mechanischen Komponenten machen die Scanner teilweise sehr teuer. Allen aktiven Verfahren ist gemeinsam, daß sie das Meßvolumen ausleuchten müssen. Je größer es ist, desto mehr Leistung wird dazu benötigt. Umgekehrt bedeutet dies, daß die beschränkt verfügbare Lichtleistung das nutzbare Volumen begrenzt. Zusätzlich muß verhindert werden, daß zum Beispiel Tageslicht die Messung stört.

### 2.1.1 Time-Of-Flight Verfahren

Die Entfernung zwischen Sensor und Oberfläche wird durch Messung der Laufzeit des Lichts ermittelt. Der Detektor liegt sehr nah bei der Lichtquelle und registriert das zurückgestreute Licht. Durch geeignete Modulation und Demodulation wird sichergestellt, daß nur das gesendete Signal vom Empfänger registriert wird.

Bekannte Vertreter sind zum Beispiel die Scanner der Firma Sick<sup>1</sup>. Ein Drehspiegel schwenkt einen Laserstrahl in einer Ebene, über einen Winkelbereich von 180° wird eine Entfernungsmessung pro Grad durchgeführt. Die Messung ist zwar sehr schnell, beinhaltet aber nur wenige Punkte, die zudem alle in dieser Ebene liegen.

Vor allem ihre hohe Geschwindigkeit macht Time-of-Flight Verfahren attraktiv. Nachteile sind die sehr hohen Anforderungen an die signalverarbeitenden Komponenten und daß nur sequentiell einzelne Punkte gemessen werden können. Beide Probleme in den Griff zu bekommen, versucht das sogenannte „Photonic Mixing Device“, bei dem die Demodulation des Empfangssignals bereits auf dem Detektor vorgenommen wird. Es ist bereits gelungen, eine kleine Matrix (12×7) dieser Detektorzellen herzustellen [26], bei der alle Pixel gleichzeitig Entfernungen messen.

### 2.1.2 Laserscanner

Abbildung 2.1 zeigt den prinzipiellen Aufbau eines Laserscanners. Eine Zylinderlinse weitet den Laserstrahl zu einer Linie auf, die mit einem beweglichen Spiegel über das Objekt bewegt wird. Die Linie auf dem Objekt wird von einer Kamera aufgenommen.

---

<sup>1</sup><http://www.sick.de>

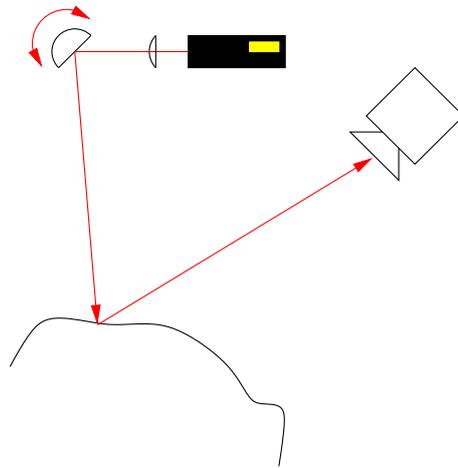


Abbildung 2.1: Funktionsprinzip eines Laserscanners.

Laserscanner arbeiten nach dem Triangulationsprinzip. Der Abstand von Spiegel und Kamera ist bekannt, der Winkel unter dem der Strahl abgelenkt wird, ergibt sich aus der Stellung des Spiegels. Zusammen mit dem Winkel, unter dem die Kamera die Laserlinie auf dem Objekt sieht, ergeben sich drei Größen. Sie erlauben die Bestimmung des Dreiecks, das durch Spiegel, Kamera und Oberflächenpunkt aufgespannt wird. Damit ist die Lage des Punktes relativ zum Laser oder zur Kamera bekannt.

### 2.1.3 Strukturiertes Licht

Ein Nachteil der Laserscanner ist, daß immer nur eine Linie des Objektes pro Bild aufgenommen wird. Beim Scannen mit strukturiertem Licht werden mehrere unterschiedliche Streifenmuster projiziert. Halbiert sich die Breite der Streifen jedesmal, benötigt man für  $n$  Linien nur noch  $\log_2(n)$  Aufnahmen.

Für die Erzeugung der Streifen kommt zum Beispiel das Prinzip des Diaprojektors zum Einsatz, bei dem geätzte Glasplatten durchleuchtet werden. Die Abmessungen können so sehr kompakt gehalten werden, die projizierten Muster sind aber fest vorgegeben. Flexibler ist der Einsatz von LCD-Projektoren, die vom Rechner aus gesteuert werden.

Scanner mit Streifenprojektion können eine vergleichbare Genauigkeit wie Laserscanner liefern und sind etwas schneller. Ein weiterer Vorteil ist, daß die keinen Laser benötigen, der bei größeren Meßvolumen und dadurch höherer Leistung gefährlich für die Augen werden kann. Sie haben allerdings die gleiche Einschränkung, nicht

für die Aufnahme bewegter Objekte geeignet zu sein.

## 2.2 Passive Verfahren

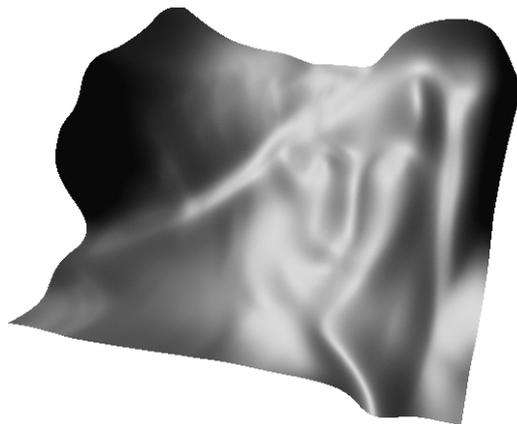
Passive Verfahren berechnen die 3D-Rekonstruktion allein aus den Kamerabildern. Durch den Verzicht auf die zusätzlichen Komponenten ist ihr Einsatz wesentlich preisgünstiger, allerdings erreichen sie nicht die hohe Genauigkeit von Scannern, die mit Laser oder Streifenprojektion arbeiten. Bei einigen Anwendungen, wie zum Beispiel mobilen Robotern oder der Rekonstruktion digitaler Geländemodelle aus Luftaufnahmen, ist man von vorneherein auf passive Verfahren beschränkt. Durch die Anpassung von Baugröße und Optik der Kamera läßt sich das erfaßbare Volumen, anders als bei den aktiven Verfahren, leicht in mehreren Größenordnungen verändern.

### 2.2.1 „Shape From Shading“

Die Lichtmenge, die von einem Oberflächenpunkt in Richtung der Kamera gestreut wird, hängt unter anderem vom Winkel zwischen der Oberflächennormalen und der Lichtquelle ab. Bei „Shape From Shading“ wird versucht, aus der Änderung der Helligkeit zurück auf die Normalen zu schließen. Sind gewisse Integrierbarkeitsbedingungen erfüllt [63], kann das zu dem Normalenfeld gehörende skalare Höhenfeld berechnet werden.



(a) Testbild „Lenna“ [45].



(b) Nach dem Verfahren von Lee und Kuo [44] berechnete Oberfläche.

Abbildung 2.2: Beispiel einer „Shape From Shading“-Berechnung.

Leider funktionieren diese Verfahren nur unter sehr eingeschränkten Bedingungen. Es darf nur eine Lichtquelle vorhanden sein, deren Position vor der Berechnung bekannt sein muß. Die Oberfläche darf nur diffus reflektieren, muß sich also streng nach dem Lambert-Gesetz [67] verhalten.

Rühren die Helligkeitsänderungen der Oberfläche zum Beispiel nicht von einer Änderung der Normalen, sondern von einer Textur her oder ist die Reflektion nicht rein diffus, schlägt die Berechnung fehl (vgl. Abb. 2.2). Bei künstlichen Bildern oder Gipsbüsten vor schwarzem Hintergrund kann „Shape From Shading“ funktionieren [75], für die Praxis ist es allerdings unbrauchbar.

### 2.2.2 Volumenrekonstruktion

Bei den volumenorientierten Verfahren wird versucht, das Meßvolumen in „Objekt“ und „nicht Objekt“ zu segmentieren [59]. Eine Möglichkeit ist, den zu digitalisierenden Gegenstand auf einen Drehteller zu stellen und in gleichmäßigen Winkelabständen Aufnahmen zu machen. Besitzen der Teller so wie der Hintergrund eine einheitliche Farbe, läßt sich die Silhouette des Objektes leicht ermitteln. Die Tangenten an den Umriß, die durch das Kamerazentrum verlaufen, schließen ein Volumen ein, in dem das Objekt liegen muß. Die Schnittmenge der Volumina aller Aufnahmen ist eine Approximation der Objektform (Abb. 2.3) [43].

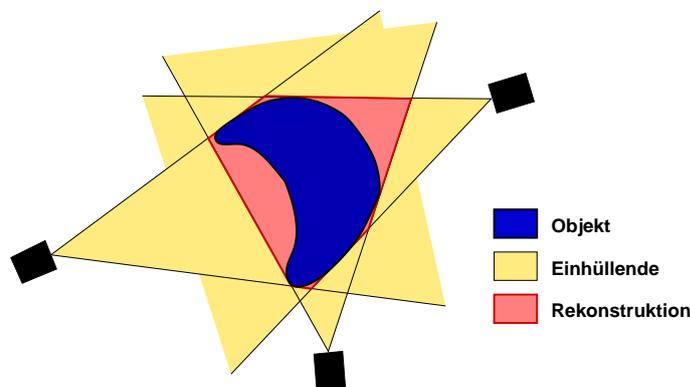


Abbildung 2.3: Rekonstruktion aus Objektkonturen.

Je genauer die Rekonstruktion sein soll, um so mehr Aufnahmen werden benötigt. Konkave Teile des Objektes können auf diese Weise jedoch nicht erfaßt werden. Die rot schraffierten Bereiche in Abbildung 2.3 zeigen die Unterschiede zwischen der echten und der rekonstruierten Objektkontur.

Eine andere Möglichkeit ist, das Volumen in Voxel zu unterteilen. Man versucht dann, jedem Voxel eine Farbe zuzuordnen, so daß seine Projektion mit den beobachteten Bildern konsistent ist („voxel coloring“). Voxel, für die das nicht gelingt, können nicht zur Oberfläche gehören und werden aus der Rekonstruktion entfernt.

Der Nachteil dieser Verfahren ist, daß sehr viele Aufnahmen benötigt werden und alle Kamerapositionen und -orientierungen genau bekannt sein müssen. Aufgrund der großen Datenmengen benötigt die Rekonstruktion zudem relativ viel Zeit.

### 2.2.3 Stereo-Rekonstruktion

Bei der Stereo-Rekonstruktion wird ein Aspekt des räumlichen Sehvermögens des Menschen nachgeahmt. Ausgangspunkt sind zwei Bilder der selben Szene, die von verschiedenen Punkten aus aufgenommen wurden. Ein Objekt wird sich im allgemeinen in den beiden Bildern an unterschiedliche Stellen abbilden (vgl. Abb. 8.1, S. 76). Aus diesem Unterschied, der mit „Disparität“ bezeichnet wird, läßt sich auf seine Entfernung schließen. Genau wie der Laserscanner beruht Stereo auf dem Prinzip der Triangulation, nur daß an die Stelle der Lichtquelle eine zweite Kamera tritt. Die Aufgabe des Algorithmus ist, herauszufinden welche Punkte in den beiden Bildern zusammengehören („matching“), also Bilder des selben Weltpunktes sind. Das Ergebnis einer Stereo-Rekonstruktion ist ein Disparitätsbild, das für jedes Pixel, an dem sie berechnet werden konnte, an Stelle von Farbwerten eine Disparität enthält.

Die Stereorekonstruktion hat sich überall dort durchgesetzt, wo sich die Szene oder die Kameras bewegen. Werden die Bilder simultan aufgenommen, zeigt die Rekonstruktion eine Momentaufnahme der Szene. Einige der im nächsten Kapitel vorgestellten Systeme sind ausreichend schnell, um die Tiefeninformation für jedes Einzelbild einer Videoaufnahme berechnen zu können.

#### Punktbasierte Verfahren

Aus beiden Bildern werden zunächst Punkte extrahiert, die eine besonders hohe Wahrscheinlichkeit haben, im anderen Bild wiedergefunden zu werden. Meistens werden mit einfachen Operatoren die Kanten oder Ecken im Bild extrahiert [33] und die stärksten davon ausgewählt. Ihre Anzahl beträgt nur noch einen Bruchteil der ursprünglich im Bild vorhandenen Pixel und sie können dann mit verhältnismäßig geringem Rechenaufwand einander zugeordnet werden.

Als Ergebnis erhält man spärlich besetzte Disparitätsbilder, bei denen nur an den Kantenpositionen eine Schätzung durchgeführt wurde. Um daraus ein komplettes Bild zu bekommen, wird zum Beispiel zwischen den gemessenen Punkten linear interpoliert. Diese Methode funktioniert für Umgebungen, die sich gut durch Polygone

annähern lassen, etwa bei der Navigation eines Roboters in Innenräumen oder bei der Rekonstruktion von Gebäuden. Ist dies nicht der Fall, oder gibt es keine ausgeprägten Kanten, wie bei Bildern aus dem medizinischen Bereich, ist eine punktbasierte Rekonstruktion nicht geeignet.

### **Flächenbasierte Verfahren**

Nicht zuletzt durch die rasante Entwicklung der Rechenleistung moderner Prozessoren ist es möglich geworden, die Disparität für alle Pixel eines Bildes zu ermitteln. Man nennt diese Rekonstruktionen deshalb auch „dicht“. Ein sehr guter Vergleich und eine Klassifizierung der auf den ersten Blick sehr großen Zahl existierender Algorithmen zur dichten Rekonstruktion finden sich in [57]. An dieser Stelle soll lediglich die Entscheidung für das gewählte Verfahren nachvollziehbar gemacht werden, ohne auf alle Details oder Unterscheidungsmerkmale einzugehen.

Alle Systeme, die echtzeitfähig sind, arbeiten nach dem Prinzip der Korrelation. Aufgrund des unterschiedlichen Samplings durch den CCD-Sensor und der leicht unterschiedlichen Ansicht des Objektes ist es nicht möglich, einzelne Pixel einander zuzuordnen. Man verwendet deshalb eine kleine Umgebung um das Pixel und versucht, diese im anderen Bild wiederzufinden. Bei fester Größe dieses sogenannten Korrelationsfensters läßt sich die Berechnung sehr effizient implementieren ([48, 16, 17], Kapitel 8). Als korrespondierend wird die Position des Fensters abgespeichert, bei der die Kreuzkorrelation zwischen ihm und dem anderen Bild maximal beziehungsweise die Differenz minimal ist. Gleichmäßig gefärbte Flächen, oder solche mit sich wiederholender Textur, können nicht eindeutig zugeordnet werden, weil dort alle Korrelationswerte identisch sind oder die Differenzfunktion mehrere Minima besitzt. Meistens wird für diese Teile des Bildes keine Schätzung der Disparität abgegeben. Einen detaillierteren Vergleich der gewählten Korrelationsfunktion mit denen, die in anderen Arbeiten verwendet werden, und die Abwägung der damit verbundenen Vor- und Nachteile gibt Abschnitt 8.3.

Anders als bei der Korrelation, die immer nur eine lokale Umgebung des Pixels betrachtet, versuchen globalen Methoden das Zuordnungsproblem simultan für alle Pixel zu lösen. Eine Möglichkeit ist, eine Oberfläche zu finden, die eine Energiefunktion minimiert. Diese Funktion wird so formuliert, daß sie sowohl Ähnlichkeit zwischen einander zugeordneten Pixeln als auch Glattheit der Oberfläche „belohnt“. Der Glattheitsterm kann dabei zum Beispiel vom Bild abhängen, um in schwach texturierten Gebieten mehr Einfluß zu haben, oder anisotrop sein und die Oberfläche entlang von Kanten glätten aber nicht quer dazu [2]. Globale Methoden liefern zum Teil deutlich bessere Ergebnisse als die Korrelation, benötigen aber selbst bei

optimierter Implementierung sehr viel mehr Zeit ([57] gibt ungefähr zehn Minuten bis eine halbe Stunde für die dort getesteten Verfahren an). Oft wird eine durch Korrelation berechnete Anfangslösung benötigt oder verwendet, um die Konvergenz zu beschleunigen.

Es existieren viele Mischformen zwischen den lokalen und globalen Ansätzen. In [77] wird zum Beispiel versucht, ein Zwischenergebnis der Korrelation, das Disparitätsvolumen (s. 8.2, S. 76), iterativ zu verbessern, bevor das Minimum gesucht wird. Bei der dynamischen Optimierung wird nicht über das ganze Bild optimiert, sondern nur eindimensional, indem ein optimaler Pfad durch die Kreuzkorrelationsmatrix der beiden Bildzeilen gesucht wird (z.B. [32]). Durch die deutlich kleinere Komplexität liegen die Rechenzeiten dieser Verfahren näher an denen der Korrelation, erreichen sie aber nicht ganz (eine bis drei Sekunden gegenüber 100–300 ms laut [57]). Ein Nachteil der dynamischen Programmierung ist der fehlende Zusammenhang zwischen den Zeilen, die in vertikaler Richtung nicht immer zusammenpassen. Neuere Ansätze verhindern dies, indem sie eine Stetigkeitsbedingung in die zu optimierende Funktion integrieren [49].

# 3

## Komplettsysteme zur schnellen Stereo-Rekonstruktion

Die hier vorgestellten Systeme zur Rekonstruktion dichter Disparitätsbilder in Echtzeit stellen den Stand der Technik zu Beginn dieser Arbeit dar. Sie arbeiten alle mit flächenbasierter Korrelation und zielen auf verschiedene Anwendungsgebiete mit vergleichbaren Anforderungen, wie zum Beispiel Kollisionserkennung und Umgebungsmodellierung autonomer Roboter, Telepräsenz oder einfache Objekterkennung. Das neueste System, DeepSea (s. 3.2.3) wird erwähnt, weil es zum Zeitpunkt der Drucklegung das leistungsfähigste System ist.

Allen Systemen ist gemeinsam, daß ihre Kameras einen großen Öffnungswinkel und eine große Tiefenschärfe besitzen. Für die oben aufgezählten Anwendungen ist dies sicherlich sinnvoll, zum Headtracking mit der geforderten Genauigkeit aber nicht. Zum einen ist der MEG-Raum meistens etwas abgedunkelt, man benötigt also lichtstärkere Objektive. Zum anderen sollte keine Auflösung verschenkt werden, der Kopf das Bild also so weit wie möglich ausfüllen, was bei einer Entfernung von drei bis vier Metern zur Kamera eine relativ große Brennweite erfordert. Ebenso kann der meistens auf 64, 32 oder sogar weniger Tiefenstufen beschränkte Suchbereich beim Stereoendoskop aufgrund der besonderen Optik zu klein sein.

### 3.1 Forschungsprojekte

Die folgenden beiden Projekte sind speziell unter dem Gesichtspunkt entworfen worden, die Wiederholfrequenz von Videobildern zu erreichen. Sie verwenden sehr aufwendige, spezielle Hardware, weil die Rechenleistung von normalen Workstations bei weitem nicht ausreichte, um diskutable Bildgrößen verarbeiten zu können. Selbst

mit der Spezialhardware sind die verarbeitbaren Bilder noch weit von der vollen Videoauflösung entfernt. Die Systeme wurden für genau eine Bildgröße und einen festen Suchbereich entworfen und sind daher sehr unflexibel. Beide haben gemeinsam, daß sie die Disparitäten nur pixelgenau ausgeben und nicht versuchen, durch Interpolation Subpixelgenauigkeit zu erreichen.

#### 3.1.1 CMU Stereo Machine

Die „Video Rate Stereo Machine“ der Carnegie Mellon Universität [35] wurde unter anderem dafür gebaut, reale und computergenerierte Objekte in einem virtuellen Studio zusammen agieren zu lassen und dabei Verdeckungen korrekt zu berücksichtigen („Z-Keying“). Das System besteht aus acht DSPs (Texas Instruments TMS320C40) für die Korrelation und weiterer Hardware, die zusammen in einem 19-Zoll Schrank untergebracht sind. Um die Qualität des Tiefenbildes zu verbessern, kommen nicht nur zwei sondern bis zu sechs Kameras zum Einsatz. Bei einer Bildgröße von  $200 \times 200$  Pixeln und 23 durchsuchten Disparitäten konnten die von NTSC benötigten 30 Hz erreicht werden.

#### 3.1.2 PARTS

Bei PARTS<sup>1</sup> handelt es sich um eine PCI-Karte, auf der sich 16 FPGAs (Xilinx 4025, s. Abschnitt 5.2) und 16 SRAM Bänke von je 1 MB befinden. Auf ihr wurde eine Stereo-Korrelation implementiert [71], die auf dem Census-Algorithmus basiert [74]. Das Board erreicht 42 Bilder pro Sekunde bei Bildern der Größe  $320 \times 240$  Pixel und 24 Tiefenstufen. Die Karte besitzt kein Kamerainterface, deshalb müssen die Bilder vor der Berechnung über den PCI-Bus auf die Karte übertragen werden.

## 3.2 Kommerziell erhältliche Systeme

### 3.2.1 Small Vision System

Das Small Vision Module [39] war das erste kommerziell verfügbare Komplettsystem zur schnellen 3D-Rekonstruktion. Es beinhaltet eine Platine mit zwei Kameras und einem DSP (ADSP 2181) und Software zur Kalibrierung und Triangulierung. Vom Korrelationsalgorithmus existieren mehreren Implementierungen, eine davon ist die auf dem DSP laufende. Das Small Vision System ist die Implementierung für skalare Prozessoren in C, zusätzlich existiert eine MMX-Version, die als etwa viermal so

---

<sup>1</sup>„Programmable And Reconfigurable Tool Set“

schnell wie die C-Version angegeben wird. Bei  $320 \times 240$  Pixeln und 32 Disparitäten kommt ein Pentium II mit 233 MHz damit auf 12 Bilder pro Sekunde.

Um diese Geschwindigkeit zu erreichen, sind nur bestimmte vorgegebene Suchbereiche (8, 16, 32 und 64) und beschränkte Fenstergrößen (5–13) wählbar, für die der Algorithmus feste Optimierungen besitzt.

### 3.2.2 Triclops

Von der Firma Point Grey Research<sup>2</sup> wird ein System vertrieben, bei dem eine Software-Bibliothek für Intel Prozessoren mit verschiedenen Kameras kombinierbar ist, die über Firewire angebunden werden. Zu Beginn dieser Arbeit existierte nur eine Dreifachkamera mit Schwarzweiß Sensoren. Inzwischen sind Stereomodelle und Farbsensoren erhältlich, die Farbinformation wird allerdings nur für die Textur und nicht zur Korrelation verwendet. Die Kameras sind fest in ein Gehäuse eingebaut und ab Werk kalibriert. Sie synchronisieren sich über Firewire, was bedeutet daß nur spezielle Kameras verwendet werden können. Mit der Triclops Bibliothek kann ein Pentium III mit 800 MHz die Stereorekonstruktion aus  $320 \times 240$  Pixeln und 32 Tiefenstufen mit 12 Hz berechnen, bei der Verwendung von drei Kameras geht die Geschwindigkeit auf 8,6 Hz zurück.

### 3.2.3 DeepSea

Das von der Firma Tyzx<sup>3</sup> entwickelte Board, ein Nachfolger des PARTS Systems aus Abschnitt 3.1.2, wurde Ende 2001 vorgestellt. Es vereint die Funktion der ursprünglich 16 FPGAs in einem Baustein von deutlich höherer Integrationsdichte. Hinzugekommen ist ein digitales Kamerainterface (LVDS), so daß die Bilder direkt mit der Karte erfaßt werden können. Bei einer Bildgröße von  $512 \times 480$  Pixeln und 52 Tiefenstufen werden 30 Hz erreicht. Als Besonderheit ist eine Subpixelinterpolation mit 5 Bit Auflösung in Hardware verfügbar.

---

<sup>2</sup><http://www.ptgrey.com>

<sup>3</sup>Die Aussprache soll an „physics“ erinnern, <http://www.tyzx.com>



# **Teil II**

## **Hardware**



# 4 Stereokamera

Die Stereokamera soll für die in Abschnitt 1.1 beschriebene Rekonstruktion von Köpfen geeignet sein, aber auch dazu dienen, einfach nur Erfahrung mit dem System sammeln zu können. Zu Beginn der Arbeit liefen die Entwicklung der Kamera und der Software parallel. Es wurde angestrebt, möglichst früh reale Bilder zu haben, da sie sich doch merklich von den im Netz erhältlichen Stereobildpaaren unterscheiden, mit denen die Software bis dahin getestet wurde.

Wie zu Beginn von Abschnitt 3.2 erwähnt, sollen die Kameras eine relativ große Brennweite haben können. Beim Test in einem normalen Büroraum ist es dagegen sinnvoll, die drei bis vier Meter Abstand von Kamera zum Objekt zu verkleinern. Die ausgewählten Kamerablöcke haben eine variable Brennweite, der horizontale Öffnungswinkel variiert zwischen  $49^\circ$  und  $4,3^\circ$ . Nach einer Änderung des Zooms müssen sie natürlich neu kalibriert werden, was aber durch die einfache Handhabung der Kalibrierung (s. Abschnitt 6.4, S. 57) keine Einschränkung für den Betrieb darstellt.

Die zweite Anforderung ist, daß es sich um eine Farbkamera handeln soll. Nach [38, 42] verbessern sich die Ergebnisse der Disparitätsberechnung (Kapitel 8) gegenüber der Verwendung von Schwarzweißbildern etwa um 20–25 %. Dies ist besonders bei der Aufnahme von Gesichtern von Vorteil, die zu großen Teilen aus nur schwach texturierten Flächen bestehen.

Es wurden Kameras ausgewählt, die ein analoges Ausgangssignal liefern. Digitale Kameras waren zu Beginn der Entwicklung nur im industriellen Umfeld und mit Festbrennweite zu finden und sehr viel teurer als das hier verwendete Modell. Das Stereo-Endoskop arbeitet ebenfalls mit analogen Kameras, so daß es sich einfach an Stelle der Stereokamera an den Framegrabber (s. Kapitel 5) anschließen läßt.

Die Kameras sind zusammen in ein Gehäuse eingebaut, das verschoben oder auf

verschiedene Objekte ausgerichtet werden kann, ohne die Kalibrierung neu durchführen zu müssen (vorausgesetzt die Kameras ändern den Zoom nicht). Abbildung 4.1 zeigt das Gehäuse mit den beiden Kamerablöcken, die im nächsten Abschnitt vorgestellt werden.

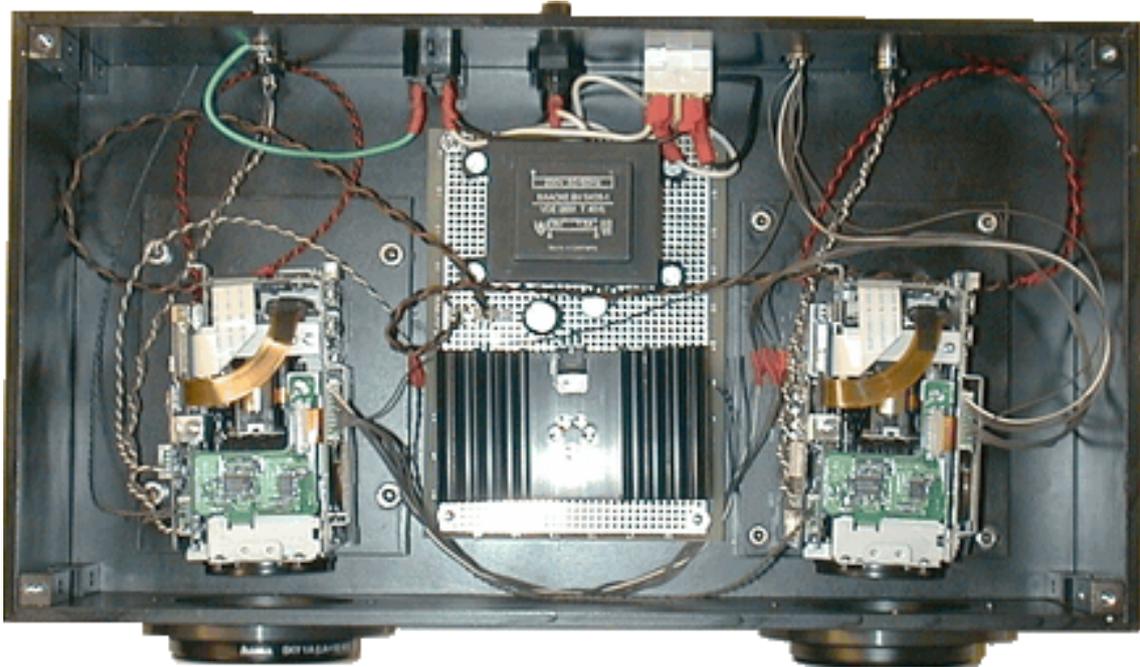


Abbildung 4.1: Aufbau der Stereokamera.

Durch Tausch der Position von Netzteil und einer Kamera kann der Abstand zwischen den Kameras entweder 20 cm oder 6 cm betragen, damit ist eine einfache Adaption an die Objektentfernung möglich. Die Genauigkeit der Triangulation ist am größten, wenn die Kameras aus  $90^\circ$  unterschiedlichen Richtungen auf das Objekt gerichtet sind. Leider funktioniert damit die Korrelation nicht mehr, weil die Bilder sehr unterschiedliche Ansichten des Objektes darstellen. Für sie wäre es am besten, wenn die beiden Bilder aus der selben Position aufgenommen würden, dann wird aber von den Kameras und dem Objektpunkt kein Dreieck mehr aufgespannt und die Position des Punktes kann nicht berechnet werden. Als funktionierender Kompromiß hat sich ein Winkel von  $5\text{--}10^\circ$  zwischen den Kameras erwiesen. Während 20 cm Abstand der Kameras für etwa 3 m Entfernung geeignet sind, klappt die Korrelation bei Objekten in nur 1 m Abstand besser, wenn die Kameras in der anderen Position eingebaut sind (die vordere Blende wird dazu entfernt).

## 4.1 Kamerablock

Von allen in Betracht gezogenen Kameras erfüllt eine Blockkamera von SONY die Anforderungen am besten. Zwei der Kameramodelle der EVI-370 Serie [61] lassen sich auf ein externes Videosignal synchronisieren. Von diesen beiden wird die PAL Variante dem NTSC Modell vorgezogen, da sie mehr Zeilen auflöst und PAL in Europa weiter verbreitet ist als NTSC. Ihre technischen Daten sind:

- Kompakter Block zum Einbau in ein eigenes Gehäuse (Abb. 4.2)
- 1-Chip  $\frac{1}{3}$ -Zoll CCD,  $752 \times 582$  Pixel,  $450 \times 400$  Linien Auflösung
- 12-fach Motorzoom (5,4–64,8 mm Brennweite), 2-fach Digitalzoom
- Deutlich größere Apertur als z.B. Boardkameras, dadurch größere Lichtstärke ( $F=1.8-2.8$ ) und besseres Signal-zu-Rausch-Verhältnis
- Autofokus, Shutter von  $\frac{1}{50}$  s– $\frac{1}{10000}$  s
- Weißabgleich: Automatik, Kunstlicht, Tageslicht oder getriggert
- Externe Synchronisation auf ein Videosignal (Gen-Lock, d.h. H/V und Farbträger werden synchronisiert)
- Ausgang Composite und Y/C (S-VHS)
- Alle Parameter der Kamera sind mit dem VISCA-Protokoll [62] über die serielle Schnittstelle einstell- und abfragbar. Bis zu sieben Kameras können an eine RS-232C Schnittstelle angeschlossen werden („daisy chain“).

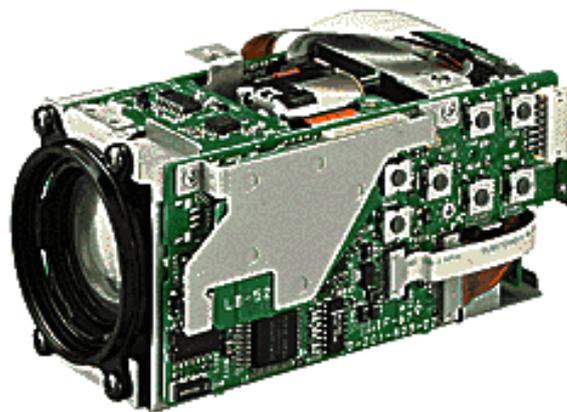


Abbildung 4.2: Kamerablock der Serie EVI-370.

Der Composite Ausgang mischt die Helligkeitsinformation mit dem auf eine Trägerfrequenz aufmodulierten Farbsignal, während bei Y/C Helligkeit und Farbe auf getrennten Adern übertragen werden. Die Übertragung des Farbsignals im Basisband ohne Modulation resultiert in einer höheren Qualität, deshalb wird der Y/C Ausgang der Kameras verwendet.

Das Videosignal am Composite Ausgang der linken Kamera dient dazu, die rechte zu synchronisieren. Es wird dazu nicht das Y-Signal für den Framegrabber abgespaltet, da sich daraus Probleme mit der Anpassung der Signale aufgrund des dann nicht mehr übereinstimmenden Wellenwiderstandes ergeben.

### 4.2 Stereoendoskop

In der Motivation (1.1) wurde bereits erwähnt, daß das System bei der Laparoskopie von Tieren eingesetzt werden soll, um Organe zu vermessen [28]. Für erste Tests wurde vom Forschungszentrum für Medizintechnik und Biotechnologie e.V.<sup>1</sup> das in Abbildung 4.3 dargestellte Stereoendoskop zur Verfügung gestellt. Es handelt sich um ein starres Endoskop mit zwei Stablinsenoptiken, 10 mm Apertur und parallelen optische Achsen. Die Kameras sind in den Griff integriert. Ausgangssignal sind zwei S-VHS Videosignale, die bei diesem Prototyp leider noch nicht synchronisiert sind, da dies bisher zur reinen Anzeige für den Operateur nicht notwendig war.



(a) Gesamtansicht.



(b) Stereo-Optik, umgeben von den Lichtleitern zur Beleuchtung.

Abbildung 4.3: Stereoendoskop.

---

<sup>1</sup><http://www.fzmb.de>

# 5 Stereoframegrabber

Die Stereokamera liefert zwei synchronisierte, analoge Videosignale. Der Framegrabber hat die Aufgabe, diese Signale zu digitalisieren, damit sie mit dem Rechner weiterverarbeitet werden können.

Zu Beginn der Arbeit existierten keine kommerziell erhältlichen Framegrabber, die zwei Farbsignale gleichzeitig aufnehmen konnten. Es standen nur die folgenden drei Varianten zur Auswahl:

- Als stereofähig beworbene Produkte teilten lediglich die R-, G- und B-Kanäle eines Farbsignals so auf, daß drei einzelne Schwarzweißkameras angeschlossen werden konnten.
- Bei den Modellen mit mehreren Farbeingängen wurden diese gemultiplext. Die Bilder zweier Kameras konnten nacheinander erfaßt werden aber nicht gleichzeitig.
- Zwei Farb-Framegrabber parallel zu betreiben funktioniert nur eingeschränkt, da sich die beiden Karten bei der gleichzeitigen Übertragung der Bilder den Busmaster-Betrieb streitig machen. Dadurch findet keine kontinuierliche Übertragung statt und die Datenrate bleibt deutlich hinter der maximal mit dem PCI-Bus erzielbaren zurück. Zusätzlich geht der synchrone Charakter der Aufnahmen verloren, die Software muß herausfinden, welche der Bilder gleichzeitig aufgenommen wurden.

Keine der verfügbaren Lösungen erfüllte also die Anforderungen, weshalb der farbfähige Stereo-Framegrabber auf Basis der im nächsten Unterkapitel beschriebenen FPGA-Karte entwickelt wurde [12]. Abbildung 5.1 zeigt die Karte zusammen mit dem in Abschnitt 5.3 beschriebenen Modul zur simultanen Digitalisierung zweier Analogquellen.

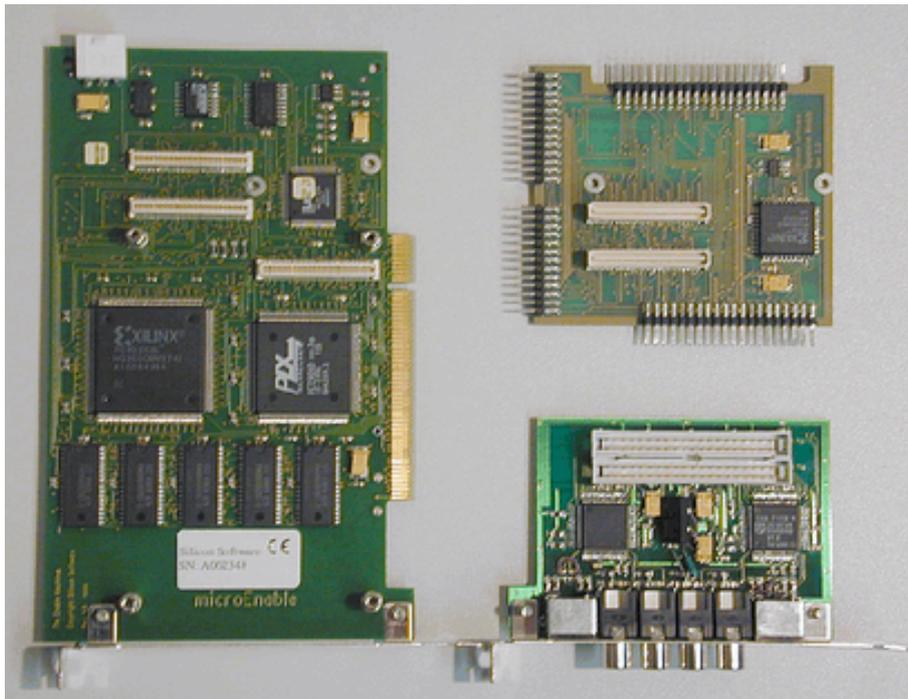


Abbildung 5.1: FPGA-Karte microEnable (links), Adapterplatine (rechts oben) und Kamerainterface mit den zwei Video A/D-Wandlern (rechts unten).

### 5.1 FPGA Karte „microEnable“

Mit der Entwicklung des Framegrabbers mußte glücklicherweise nicht bei Null begonnen werden. Als Basis dient die PCI-Karte der Firma Silicon-Software<sup>1</sup>, einer Spin-Off Firma des Lehrstuhls für Informatik V der Universität Mannheim [10].

Sie besteht aus einem programmierbaren Logikbaustein der Serie 4000 von Xilinx [72], 512 kB oder 2 MB lokalem SRAM, einem Baustein zur Anbindung an den PCI-Bus (PLX 9080) und mehreren Steckverbindern zum Anschluß von Erweiterungen. Die Treiber für Windows und Linux können zum Transfer der Daten verschiedene Modi verwenden, für die Videodaten kommt der sogenannte „scatter-gather-DMA“ zum Einsatz. Im Gegensatz zum herkömmlichen DMA-Modus werden die Daten nicht in einen physikalisch zusammenhängenden Zwischenpuffer im Adreßraum des Betriebssystemkerns übertragen und dann in den Adreßraum des Programmes umkopiert, sondern der PLX-Baustein wird so programmiert, daß er diejenigen Speicherseiten beschreibt, auf die das Anwendungsprogramm direkt zugreifen kann. Die

<sup>1</sup><http://www.silicon-software.com>

Seiten liegen dabei physikalisch verstreut („scatter“) im Speicher und werden bei der Übertragung zusammengesammelt („gather“). Der Treiber stellt sicher, daß sie während des Schreibvorgangs nicht von der Speicherverwaltung verschoben oder ausgelagert werden. In diesem Modus können über 90 % der theoretisch möglichen Bandbreite des PCI-Busses von 132 MB/s erreicht werden. Besonders von Vorteil ist, daß der Transfer ohne Zutun der CPU geschieht, die lediglich durch einen Interrupt vom Ende der Übertragung eines Bildes erfährt.

Die microEnable wurde bis dahin bereits als Framegrabber für industrielle Kameras mit digitalem Interface verwendet. Sowohl die Treiber als auch eine vorhandene Software-Bibliothek, die ein einheitliches Interface für Framegrabber-Anwendungen beinhaltet [58], konnten nach kleineren Anpassungen für die analoge Stereokamera übernommen werden. Neu zu entwickeln waren dagegen die Erweiterungsplatine zur Digitalisierung der Videosignale (5.3) und die Konfiguration des FPGAs (5.4).

## 5.2 Field Programmable Gate Arrays

Der FPGA ist das Kernstück der microEnable Karte. Er besteht aus Logikblöcken, deren Funktion konfiguriert werden kann, und programmierbaren Verbindungen zwischen diesen Blöcken. Die folgenden Ausführungen behandeln die auf der microEnable verwendete 4000er Serie der Firma Xilinx. Im Rahmen dieser Arbeit kann und soll lediglich auf das Funktionsprinzip der Bausteine eingegangen werden, um verständlich zu machen wie der Framegrabber damit realisiert wurde.

Die wichtigsten Bestandteile eines Logikblocks („Configurable Logic Block“, CLB, Abb. 5.2(a)) sind zwei Funktionsgeneratoren, mehrere Multiplexer und zwei Flip-Flops. Die Funktionsgeneratoren (F und G) besitzen jeweils vier Eingänge und einen Ausgang. Sie können eine frei programmierbare Funktion der vier Eingangsvariablen enthalten oder als 16 Bit On-Chip RAM-Zelle verwendet werden. Die Funktionsgeneratoren arbeiten asynchron, d.h. ihr Ausgang ändert seinen Zustand sofort (nach der Laufzeit durch das Bauteil) mit jeder Änderung der Eingangssignale. Zur Synchronisation mit Taktsignalen dienen D-Flip-Flops. Die Verbindungen zwischen den Generatoren, den Flip-Flops und den Ein- und Ausgängen des CLBs sind teilweise fest vorgegeben und teilweise durch Multiplexer realisiert, die allerdings nicht zur Laufzeit sondern durch die Konfiguration geschaltet werden.

Die Ein-/Ausgabe-Blöcke (Abb. 5.2(b)) können gepuffert oder ungepuffert betrieben werden. Der gepufferte Modus dient dazu, das Signal auf eines der Taktsignale `Input Clock` oder `Output Clock` zu synchronisieren, je nachdem ob der Block als Eingang oder Ausgang dient. Die Ausgänge können hochohmig geschaltet werden,



sind also busfähig. Nutzt man den dazu vorgesehenen Tri-State Eingang T und legt den eigentlichen Eingang Out auf Masse, erhält man einen Open-Drain Ausgang (Abb. 5.2(c)). Zur Ansteuerung des im nächsten Abschnitt beschriebenen Video-Wandlers über den I<sup>2</sup>C-Bus ist dann nur noch ein zusätzlicher externer Pull-Up Widerstand nötig.

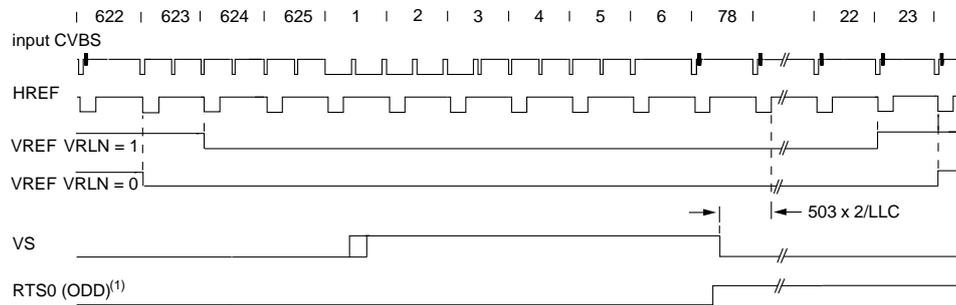
Die CLBs sind in einer Matrix angeordnet, in deren Größe sich die einzelnen Bausteine der Serie unterscheiden. Der Xilinx 4036 enthält zum Beispiel  $36 \times 36 = 1296$  CLBs. Die I/O-Blöcke befinden sich an den vier Seiten des Chips. Zwischen den Blöcken verlaufen horizontale und vertikale Leitungen, die sie miteinander verbinden (Abb. 5.2(d)). „Single-Length Lines“ verbinden benachbarte CLBs, „Double-Length Lines“ überspringen immer einen CLB und „Long Lines“ verlaufen über die gesamte Länge des Chips. Genau wie die Funktion der CLBs sind die Verbindungen zwischen den Leitungen programmierbar, sie werden wie RAM-Zellen beschrieben. Die Konfiguration des gesamten FPGAs erfolgt durch einen seriellen Bitstrom und dauert je nach Größe etwa 20–50 ms.

## 5.3 Video Analog-Digital Wandler

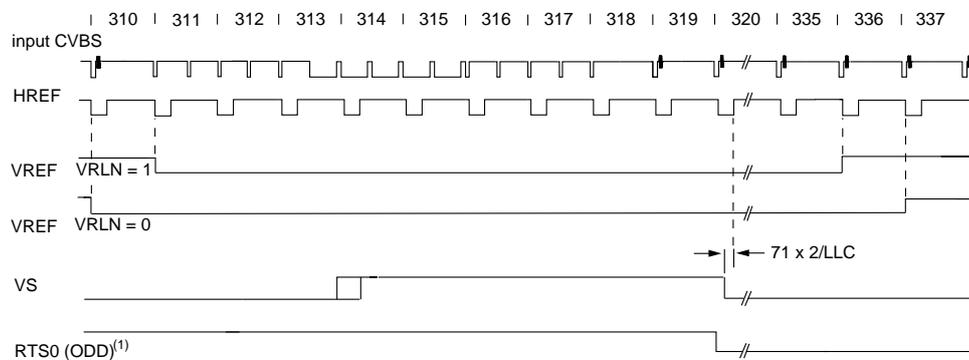
Die Digitalisierung der Videosignale übernimmt ein integrierter Schaltkreis von Philips (SAA7111 [55]). Für die Zusammenarbeit mit dem FPGA ist es von Vorteil, daß er ein einfaches paralleles und kein PCI-Interface besitzt, wie zum Beispiel die weit verbreiteten ICs der Firma Brooktree, und daß er mit einer geringen Anzahl externer Bauteile auskommt. Mit ihm können die gängigen analogen Videoformate nach PAL- oder NTSC-Norm digitalisiert werden. Die Einstellung des Video- und Ausgangsformats, der Helligkeit und Farbsättigung werden über einen I<sup>2</sup>C-Bus vorgenommen.

Der FPGA wertet die Ausgangssignale aus und muß entscheiden, wann ein Bild anfängt, wann es aufhört und wann gültige Videodaten vorliegen. Ein PAL-Bild („frame“) besteht aus 625 Zeilen, die in zwei Halbbildern („fields“) übertragen werden, zuerst alle ungeraden Zeilen, dann die geraden. Die Abbildungen 5.3(a) und 5.3(b) zeigen den Verlauf der vertikalen Synchronisationssignale zwischen den Halbbildern. Das Signal VREF markiert die vertikale Austastlücke. Geht es auf H-Pegel, sind die Zeilen gültig. Der Chip ist so programmierbar, daß er entweder 286 oder 288 Zeilen pro Halbbild signalisiert. VS wird nicht verwendet, da sein Timing nicht so eng an das Videosignal gekoppelt ist, wie das von VREF; analoges gilt für HS bei der horizontalen Synchronisierung. ODD dient dazu, die Halbbilder voneinander unterscheiden zu können. Während des ersten Halbbildes ist es gesetzt, beim zweiten

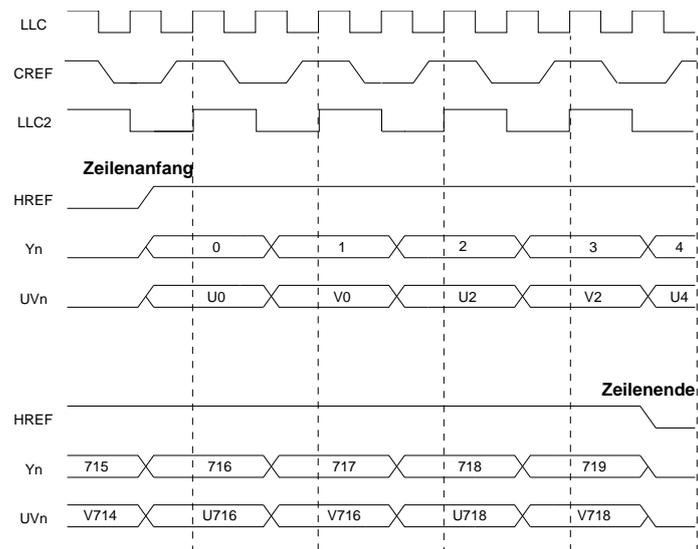
## 5 Stereoframegrabber



(a) Vertikales Timing, erstes Halbbild.



(b) Vertikales Timing, zweites Halbbild.



(c) Horizontales Timing im YUV422 Modus.

Abbildung 5.3: Vertikale und horizontale Synchronisationssignale am Ausgang des Video Analog-Digital Wandlers SAA7111.

nicht.

Von jeder Zeile werden 720 Pixel digitalisiert. Abbildung 5.3(c) zeigt die horizontale Synchronisation zu Beginn und Ende einer Zeile. Das Synchronisationssignal **HREF** dient der Unterscheidung zwischen den Videodaten einer Zeile und der horizontalen Austastlücke. **CREF** signalisiert, daß bei steigender Flanke von **LLC** Daten vom Video-Bus (**VPO**) übernommen werden können. Dargestellt ist das digitale Ausgangsformat **YUV422**, bei dem für zwei nebeneinander liegende Pixel zwei Helligkeitswerte **Y** und je ein gemeinsamer Farbwert **U** und **V** übertragen werden. Der **SAA7111** kann die Daten auch direkt in **RGB** liefern. Im Modus **RGB565** ist das Timing identisch zu dem bei **YUV422**, pro Pixel werden auf dem 16 Bit breiten Bus je 5 Bit für Rot und Blau und 6 Bit für Grün übertragen.

Für die Weiterverarbeitung mit der Software am geeignetesten ist das Format **RGB888**, bei dem für jeden Farbkanal 8 Bit verwendet werden. Das Timing weicht von dem der anderen Modi ab, weil hier 24 Bit pro Takt übertragen werden müssen, aber nur 16 Leitungen zur Verfügung stehen (Abb. 5.4). Die oberen 8 Bit des Busses behalten ihre Daten bei, während die unteren 8 Bit einmal bei H-Pegel von **CREF** gelesen werden und einmal bei L-Pegel. Ausschlaggebend für den Auslesezeitpunkt ist beide Male die steigende Flanke von **LLC**. Die in [55] vorgeschlagene Kombination der drei Teile zu einem **RGB888** Wert mit einem Register (74HCT574) wird in den **FPGA** verlegt. Die kombinierten Pixeldaten kommen dann mit einer Frequenz von 13,5 MHz an, wie in den anderen Modi auch.

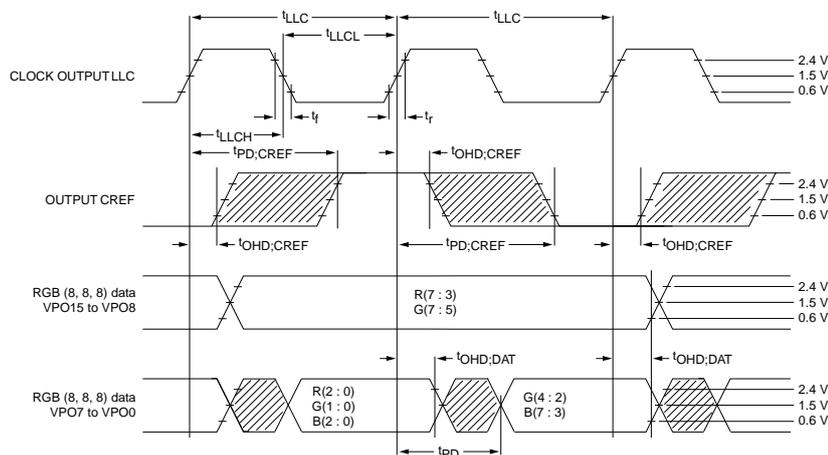


Abbildung 5.4: Übertragung eines Pixels im **RGB888** Modus.

Leider hat das Zeilensprungverfahren den großen Nachteil, daß die Halbbilder nicht nur nacheinander übertragen, sondern auch aufgenommen werden. Zwischen

der Aufnahme zweier zu einem Bild gehörender Halbbilder liegen 20 ms. Bewegen sich Objekt oder Kamera in dieser Zeit, passen die beiden Halbbilder nicht mehr zusammen, es entstehen kammförmige Artefakte. Will man bewegte Szenen aufnehmen, muß man sich auf ein Halbbild beschränken, wodurch sich die verfügbare vertikale Auflösung halbiert. Soll das Seitenverhältnis der Bilder wieder 4:3 sein, muß die Auflösung in horizontaler Richtung ebenfalls halbiert werden, die Bilder sind dann  $360 \times 288$  Pixel groß und die Datenrate auf dem PCI-Bus geht von ungefähr  $60 \text{ MB/s}$  ( $2 \text{ Kameras} \times 720 \times 576 \text{ Pixel} \times 3 \text{ Byte/Pixel} \times 25 \text{ Bilder/s}$ ) auf  $15 \text{ MB/s}$  zurück.

## 5.4 Hardwarebeschreibung mit CHDL

Um aus dem FPGA einen Framegrabber zu machen, der die gerade beschriebenen Signale auswertet und die Bilddaten in den Hauptspeicher des Rechners transportiert, muß er entsprechend konfiguriert werden. Während bei Software das zeitliche Verhalten direkt durch das Programm bestimmt wird, entsteht es bei Hardware durch die Dynamik der einzelnen Bausteine und der Eingangssignale. Die Verschaltung wird nur einmal festgelegt, bei FPGAs bei jeder Konfiguration neu, bei nicht rekonfigurierbaren Schaltkreisen während der Produktion. Sie bleibt während des Ablaufs der Software unverändert.

Für den Entwurf des Framegrabbers kam keine der herkömmlichen Hardwarebeschreibungssprachen zum Einsatz, wie zum Beispiel VHDL<sup>2</sup> [3] oder Verilog [4], sondern das am Lehrstuhl entwickelte CHDL [40, 41].

### 5.4.1 Schaltungsentwurf und -synthese

Bei CHDL handelt es sich um eine C++ Bibliothek, die aus der Schaltungsbeschreibung eine Netzliste generiert. Die wichtigsten Eigenschaften sind die konsequente Nutzung von Objektorientierung und das Überladen von Operatoren. Alle Bauteile, Ein- und Ausgänge sind als Klassen realisiert, die Schaltung wird als C++ Programm formuliert. Zum Beispiel wird ein Bauteil, das je einen Ein- und Ausgang besitzt, diese als Membervariablen enthalten. Ihre Verschaltung realisiert der Zuweisungsoperator „=“.

Algorithmus 5.1 zeigt ein einfaches Beispiel, bei dem zwei Register mit je 32 Bit vom PCI-Bus aus beschrieben und gelesen werden können. Nach dem Einbinden der Header-Dateien, in denen die Definitionen für CHDL und der microEnable spezifi-

---

<sup>2</sup>VHSIC (Very High Speed Integrated Circuit) Hardware Description Language

**Algorithmus 5.1** CHDL-Beispiel

---

```
#include "chdl.h"
#include "menable.h"

class myFirst : public CHIP4036XL {
public:
    myFirst();
    virtual void Simulate();
};

myFirst::myFirst() : CHIP4036XL() {
    PCI_SLAVE PCI(0x00000000, 17, 0, 0, 0x01, "PCI", this);
    DRegister REG1(32, "REG1", PCI.CLK), REG2(32, "REG2", PCI.CLK);

    REG1.D = PCI.DO;
    REG2.D = PCI.DO;
    REG1.CE = PCI.wr_rdy0 & !PCI.A[0];
    REG2.CE = PCI.wr_rdy0 & PCI.A[0];
    PCI.DI = PCI.rd_rdy0 & ((!PCI.A[0] & REG1.Q) | (PCI.A[0] & REG2.Q));
}
```

---

schen Bauteile enthalten sind, wird die Klasse der Beispielschaltung deklariert. Sie ist von einer Klasse abgeleitet, die die Grundfunktionalität des verwendeten FPGAs bereitstellt. Im einfachsten Fall enthält der Konstruktor die Schaltung, das Erzeugen einer Instanz dieser Klasse genügt, um die Netzliste zu generieren. Die hier deklarierte `Simulate()` Methode behandelt Abschnitt 5.4.2.

Der `PCI_SLAVE` dient der Kommunikation mit dem PLX-Chip, der das Interface zum PCI-Bus zur Verfügung stellt. Die Komplexität der Funktionalität ist gekapselt, die Klasse wird über ein einfaches Interface angesprochen. Zur einfacheren Ansteuerung sind Ein- und Ausgabe auf dem PCI-Bus logisch getrennt. Datentransfer vom Bus zum FPGA kommt am Ausgang `PCI.DO` an, sind die Daten gültig, ist das Signal `wr_rdy0` gesetzt. Der FPGA schreibt seine Daten für den Bus bei aktivem `rd_rdy0` auf `PCI.DI`. Übernahmezeitpunkt ist jeweils die steigende Flanke von `PCI.CLK`.

Der Konstruktor der beiden Register bekommt die Breite in Bits, einen symbolischen Namen und das zu verwendende Taktsignal übergeben. Sie sind intern als Parallelschaltung von D-Flip-Flops realisiert, ihre D-Eingänge werden mit dem Ausgang des PCI-Busses verbunden. Die niederwertigste Adressleitung `A[0]` entscheidet an den „Clock Enable“ Eingängen `CE`, welches Register beschrieben werden kann. Al-

le geraden Adressen schreiben oder lesen **REG1**, alle ungeraden **REG2**. Die Operatoren **!** und **&** sind so überladen, daß sie einen Inverter beziehungsweise ein Und-Gatter generieren. Auf diese Weise kann die Schaltung gut lesbar als Ausdruck in Boolescher Algebra formuliert werden.

Auf der rechten Seite des Ausdrucks, der **PCI.DI** zugewiesen wird, verknüpfen Operatoren eine einzelne Leitung mit einem Bus. Dies ist eine Abkürzung für die Verknüpfung jeder Leitung des Busses mit dem einzelnen Signal. Wenn die Adresleitung Null ist, bestimmt **REG1** die Daten, die an den PCI-Bus gehen, sonst **REG2**.

Soll aus dem Quelltext eine lauffähige Konfiguration des FPGAs werden, müssen folgende Schritte der Synthese durchlaufen werden:

1. Das Design der Schaltung wird als Quelltext formuliert, mit einem C++-Compiler übersetzt und mit der CHDL Bibliothek gelinkt.
2. Ausführen der Datei resultiert in einer Netzliste im XNF- oder EDIF-Format. Wird bei der Ausführung des Programms der Parameter **sim** angegeben, erzeugt CHDL eine Simulationsdatei (s. Abschnitt 5.4.2).
3. Ab hier kommen die Tools des FPGA-Herstellers zum Einsatz. Aus der Netzliste werden die benötigten Ressourcen wie Flip-Flops und Funktionsgeneratoren ermittelt und vom sogenannten „placer“ auf die verfügbaren CLBs verteilt.
4. Die Verbindungen der CLBs untereinander und mit den I/O-Blöcken ermittelt der „router“.
5. Schließlich wird die Bitfolge generiert, die die Konfiguration des FPGAs beinhaltet.

Router- und Placer-Stufe bestehen meistens aus mehreren Durchläufen, wobei nach verschiedenen Kriterien wie Größe oder Geschwindigkeit der Schaltung optimiert werden kann.

### 5.4.2 Simulation

Ein großer Vorteil von CHDL gegenüber den herkömmlichen Hardwarebeschreibungssprachen ist die nahtlose Integration der Simulation in den Designprozeß. Es müssen keine Testvektoren generiert und Ausgangssignale interpretiert werden, die Simulation bezieht sowohl die Hard- als auch Software ein. Sie kann mit genau dem gleichen Code erfolgen, der nachher den echten FPGA ansteuert [9].

Für das RAM und den PLX-Chip, die ja nicht als CHDL-Quelltext vorliegen, also nicht automatisch mitsimuliert werden, existieren sogenannte Simulationsmodelle,

die das Verhalten so exakt wie möglich in Software nachbilden. Bei der Entwicklung des Framegrabbers wurde ein solches Modell für den SAA7111 erstellt, um der Simulation realistische Eingangsdaten liefern zu können. Abschnitt 5.4.3 gibt einen Einblick in dieses Modul und die damit erzielten Ergebnisse.

An dieser Stelle soll kurz auf die Erstellung und Auswertung einer Simulation eingegangen werden. Algorithmus 5.2 zeigt die Simulationsroutine der bereits vorgestellten Beispielklasse.

---

**Algorithmus 5.2** Simulationsroutine des Beispiels von Algorithmus 5.1

---

```
void myFirst::Simulate() {
    AddPin("PCI/CLK");      AddPin("PCI/A<0>");
    AddPin("PCI/wr_rdy0");  AddBus("PCI/D0", 0, 31);
    AddBus("REG1", 0, 31);  AddBus("REG2", 0, 31);
    AddPin("PCI/rd_rdy0");  AddBus("PCI/DI", 0, 31);
    /* ... */
    volatile long unsigned int* ptr = GetDesignBase();
    ptr[0] = 0x1234;
    ptr[1] = 0x4321;
    cout << hex << ptr[0] << endl;
    cout << hex << ptr[1] << endl;
}
```

---

Zuerst werden alle Signale bestimmt, die während der Ausführung aufgezeichnet werden sollen. Die Auswahl erfolgt über die bei der Konstruktion der Objekte spezifizierten Namen, wobei die innerhalb einer anderen Klasse erzeugten Namen durch einen Schrägstrich getrennt angehängt werden. Bei Bussen werden das erste und letzte einzuschließende Bit angegeben. Um das Beispiel auf das Wesentliche zu beschränken, wurden die Einbindung der Simulationsmodelle und Einstellungen zur Simulation weggelassen.

Beim echten Betrieb des FPGAs liefert die Routine `GetDesignBase()` die Adresse, an der der Adreßraum des PLX vom Treiber eingeblendet wurde. Bei der Simulation zeigt sie auf einen Bereich von speziellen Speicherseiten („guard-pages“). Jeder Zugriff auf diese Seiten erzeugt eine Ausnahmebehandlung, die abgefangen und an die Simulation umgeleitet wird.

Im vorliegenden Beispiel werden beide Register beschrieben und wieder ausgelesen. Abbildung 5.5 zeigt den Ablauf der Signale bei diesen Zugriffen.

Die Taktflanke zum Simulationszeitpunkt 48 ist diejenige, bei der der Wert `0x1234` von `PCI.D0` in das erste Register `REG1` übernommen wird, zum Zeitpunkt 90 übernimmt `REG2` den Wert `0x4321`. Beide Male zeigt `wr_rdy0` an, daß die Daten auf dem

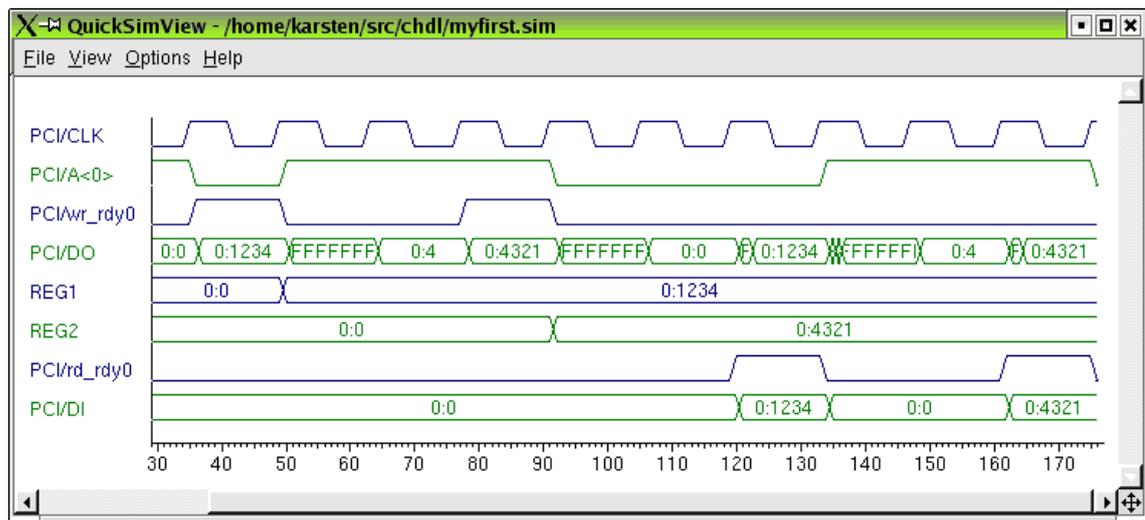


Abbildung 5.5: Darstellung der Simulation von Algorithmus 5.2.

Bus gültig sind, die beiden Zugriffe unterscheiden sich lediglich im Status der Adressleitung. Die Lesezugriffe verlaufen analog, nur daß das Signal `rd_rdy0` ausschlaggebend ist und `PCI.DI` die Daten annimmt. Auf der Konsole gibt das Programm die beiden Zahlen aus, wie beim Betrieb mit der echten Hardware auch.

### 5.4.3 Realisierung des Framegrabbers

Die Simulation war ein sehr wichtiges Hilfsmittel beim Entwurf des Framegrabbers, da es bei der Entwicklung von Hardware ohne sie oft nur die Unterscheidung zwischen „geht“ und „geht nicht“ gibt. An dieser Stelle soll allerdings nur ein Überblick über die Funktionsweise gegeben und nicht bis in die Tiefen des Framegrabbers abgestiegen werden.

Um die Datenmenge bei der Simulation überschaubar zu halten, ist das Simulationsmodell nicht fest an die Größe eines PAL-Bildes gebunden, sondern flexibel. Im gezeigten Fall liefert es ein  $12 \times 10$  Pixel großes Bild, von dem ein Halbbild mit auf 4:3 korrigiertem Seitenverhältnis erfaßt wird, also 5 Zeilen à 6 Pixel. Jedes Pixel besteht aus 16 Bit, die zum Testen einfach hochgezählt werden. Abbildung 5.6 zeigt die Simulation eines solchen Halbbildes.

Die oberen drei Zeilen beinhalten den Ausgangsteil des PCI-Busses, über den ein Reset-Signal erzeugt wird, das die Schaltung in einen definierten Ausgangszustand versetzt. Die fünf folgenden Signale sind die Ausgänge des Simulationsmodells des

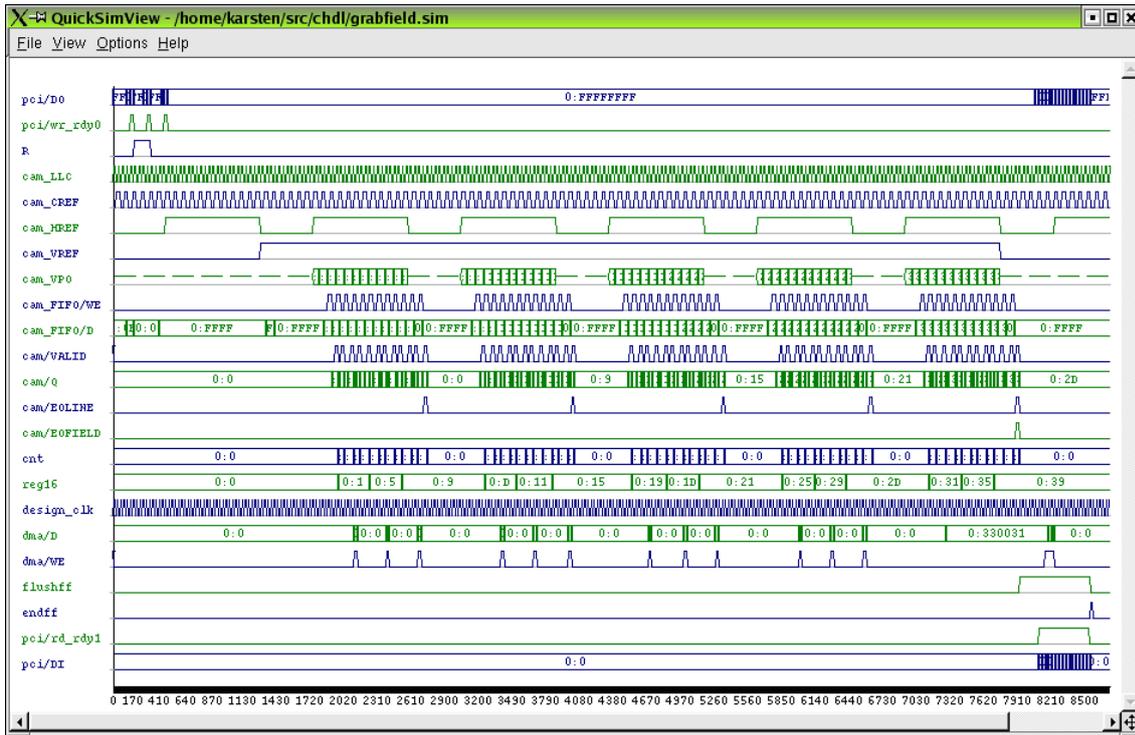


Abbildung 5.6: Simulation eines Halbbildes mit CHDL.

Video A/D-Wandlers (vgl. Abb. 5.3(a) und 5.3(b)). Gut zu erkennen sind die fünf Zeilen des Bildes an den fünf Datenpaketen auf dem Videobus VPO.

Durch ein asynchrones FIFO wird die Taktrate der weiterverarbeitenden Schaltung von der des Videosignals entkoppelt. `cam_FIFO/WE` und `cam_FIFO/D` sind die Eingangsseite, die mit 13,5 MHz arbeitet, der Ausgang `cam/VALID` und `cam/Q` ist synchron zu `design_clk`. Die Frequenz von 20 MHz dieses Taktes ist nicht kritisch, sie muß nur schneller sein als die Frequenz am Eingang des FIFOs, damit der nicht überlaufen kann.

`EOLINE` und `EOFIELD` zeigen das Ende einer Zeile beziehungsweise eines Halbbildes an. Die Komponenten `cnt` und `reg16` sorgen dafür, daß jedes zweite Pixel übersprungen wird und die verbleibenden von 16 auf 32 Bit gepackt werden, da das der Breite des RAMs entspricht. Die Pixel werden mit jeweils drei Zugriffen pro Zeile an den Puffer für die DMA-Übertragung geschickt und am Ende des Bildes über den PCI-Bus ausgelesen. Dies ist der einzige Teil, in dem die Simulation nicht mit dem späteren Ablauf übereinstimmt, bei dem der Lesevorgang bereits während des Bildes beginnt. Erstens wäre die Asynchronität der Übertragung in der Simulation

nur sehr schwer nachzuahmen gewesen. Zweitens war dieser Teil der Schaltung nicht Gegenstand der Untersuchung, weil die Puffer- und DMA-Logik aus der microEnable Bibliothek übernommen wurden.

Abbildung 5.7 zeigt einen Ausschnitt, der die Erfassung der zweiten Zeile vergrößert darstellt. `cnt` zählt die Pixel der Zeile modulo vier. Beim Zählerstand von Null wird das erste Pixel `0x000D` in das Register `reg16` zwischengespeichert. Das übernächste Pixel `0x000F` wird dann zusammen mit dem Inhalt des Registers an den 32 Bit breiten Eingang `dma/D` weitergereicht, wie an der Cursorposition durch den Wert `0x000F000D` angezeigt wird.

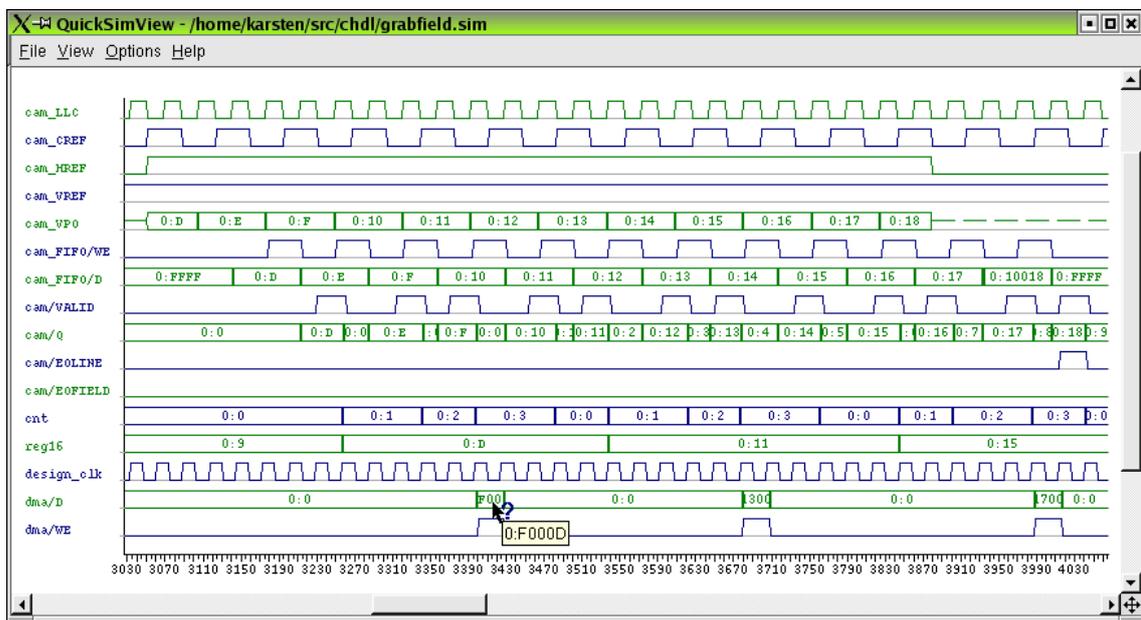


Abbildung 5.7: Aufnahme einer Zeile in der Simulation.

Nach der erfolgreichen Simulation mit einer Kamera, wurde das Design auf die endgültige Form mit zwei Kameras erweitert. Abbildung 5.8 zeigt den Datenfluß der Videodaten von den Kameras durch den A/D-Wandler und die microEnable in den Rechner.

Die microEnable besitzt leider nur eine RAM-Bank, die sich beide Kameras teilen müssen. Schreib- und Lesezugriffe der beiden Kameras und der DMA-Logik werden zeitlich ineinander verwoben. Damit keine Daten verloren gehen, muß der Takt auf der RAM-Seite größer sein als die Summe der Taktraten der angeschlossenen Schreib- und Lese-Kanäle.

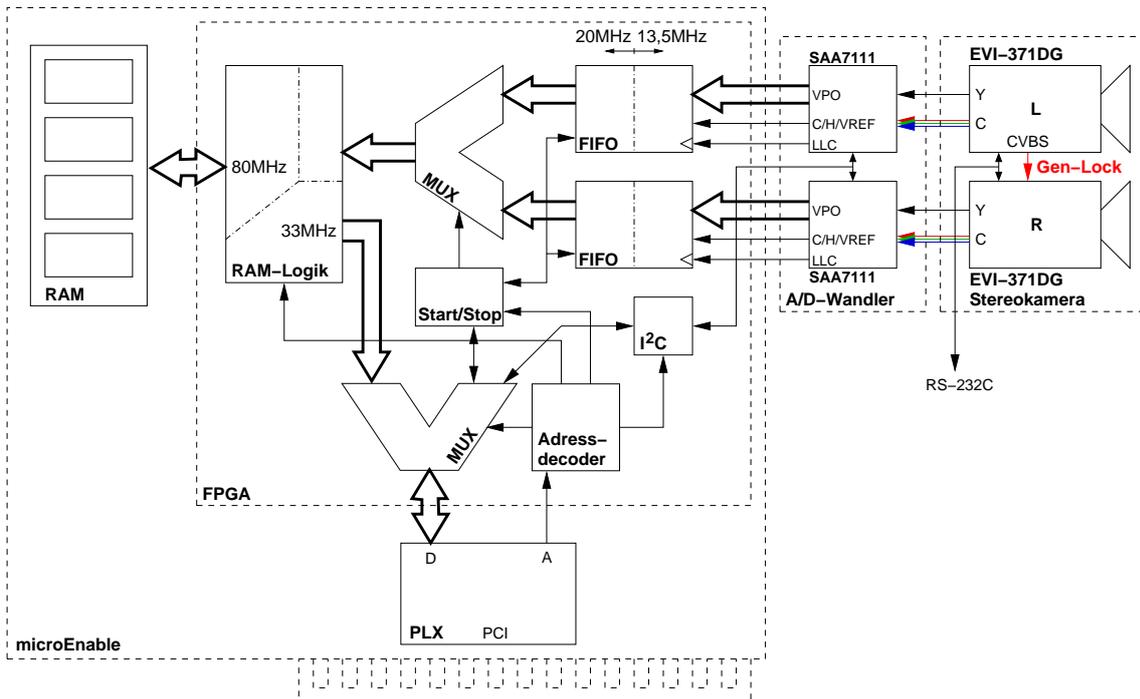


Abbildung 5.8: Hardwarekomponenten und Aufbau des Framegrabbers.

Beim Schreiben der Daten in das RAM wird ein FIFO immer ganz geleert, dann schaltet der Multiplexer auf den anderen FIFO um. Der Schreibvorgang beginnt erst bei einem gewissen Mindestfüllstand, dadurch wird erreicht, daß das RAM mit einem zusammenhängenden Block von aufeinanderfolgenden Daten („burst“) beschrieben wird.



# **Teil III**

## **Software**



# 6 Kalibrierung

Die Kalibrierung dient dazu, die Abbildungseigenschaften der Kameras zu bestimmen. Sie stellt die Verbindung zwischen den dreidimensionalen Punkten und ihren Projektionen in den beiden Bildern her. Für die 3D-Rekonstruktion ist wichtig, daß man bei einer kalibrierten Kamera jedem Bildpunkt einen Strahl zuordnen kann, auf dem der abgebildete Weltpunkt liegen muß (s. Abbildung 6.1). Dann nämlich können die Strahlen zweier korrespondierender Bildpunkte miteinander geschnitten und die Position des Weltpunktes bestimmt werden.

## 6.1 Kameramodell

Zur Beschreibung der Kamera wird das Lochkameramodell verwendet. Mit ihm lassen sich alle Kameras erfassen, die ein punktförmiges Projektionszentrum besitzen — das durchaus im Unendlichen liegen kann, was in einer Parallelprojektion resultiert — und das Bild in eine Ebene projizieren. Nicht von diesem Typ sind zum Beispiel sogenannte „pushbroom“ Kameras, deren Projektionszentrum aus einer Linie besteht [22], oder Kameras zur Aufnahme von 360°-Panoramen (z.B. [51]).

Die Parameter des Kameramodells lassen sich aufteilen in interne und externe. Letztere geben die Lage der Kamera bezüglich eines globalen Koordinatensystems an. Die internen Parameter beschreiben die Abbildung der Weltpunkte vom lokalen Koordinatensystem der Kamera in das Bild.

### 6.1.1 Interne Parameter

Jeder (sichtbare) Punkt im Raum wird entlang der Verbindungsgeraden zum Zentrum der Kamera in die Bildebene projiziert, das Bild des Punktes ist der Schnitt-

punkt der Geraden mit der Bildebene, wie in Abbildung 6.1 skizziert.

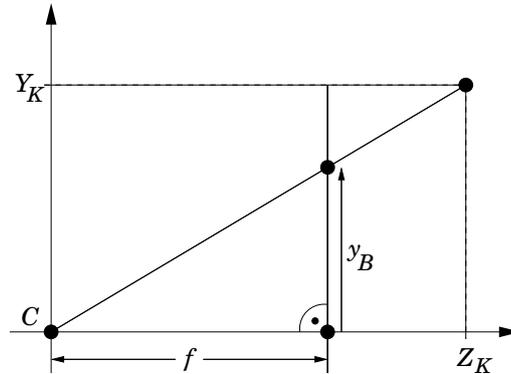


Abbildung 6.1: Abbildung eines Punktes durch Zentralprojektion.

Die Brennweite  $f$  ist der Abstand der Bildebene zum Zentrum  $C$ . Der 3D-Punkt hat im Koordinatensystem der Kamera die Koordinaten  $(X_K, Y_K, Z_K)^T$ , die Koordinaten des Bildpunktes  $(x_B, y_B)^T$  ergeben sich aus dem Strahlensatz:

$$\frac{Y_K}{Z_K} = \frac{y_B}{f} \quad \Rightarrow \quad y_B = f \frac{Y_K}{Z_K}, \text{ analog } x_B = f \frac{X_K}{Z_K}. \quad (6.1)$$

Die Gerade durch das Zentrum, die senkrecht auf der Bildebene steht, wird optische Achse genannt. Ihr Schnittpunkt mit der Bildebene ist der sogenannte Hauptpunkt.

Unter Zuhilfenahme homogener Koordinaten läßt sich Gleichung (6.1) in ein lineares Gleichungssystem umschreiben. In homogenen Koordinaten repräsentieren alle Vektoren  $(wx, wy, w)^T$  mit  $w \neq 0$  den selben Punkt  $(x, y)^T$ . Die inhomogenen Koordinaten ergeben sich durch Normierung des Vektors auf  $w = 1$ . Ein Vorteil homogener Koordinaten ist, daß sich damit auch Punkte im Unendlichen darstellen lassen. Der Vektor  $(x, y, 0)^T$  steht für denjenigen Punkt, der in Richtung  $(x, y)^T$  im Unendlichen liegt. Die Division findet beim Übergang von homogenen zu inhomogenen Koordinaten statt

$$\begin{pmatrix} \tilde{x} \\ \tilde{y} \\ \tilde{w} \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X_K \\ Y_K \\ Z_K \\ 1 \end{pmatrix}, \quad x_B = \frac{\tilde{x}}{\tilde{w}}, \quad y_B = \frac{\tilde{y}}{\tilde{w}}. \quad (6.2)$$

Es ist sinnvoll, im Bild nicht in den Einheiten der Weltkoordinaten (z.B. mm) zu arbeiten, sondern in dem durch die Pixel vorgegebenen Koordinatensystem (Abbildung 6.2). Da die Pixelkoordinaten von einer Ecke des Bildes aus gemessen werden,

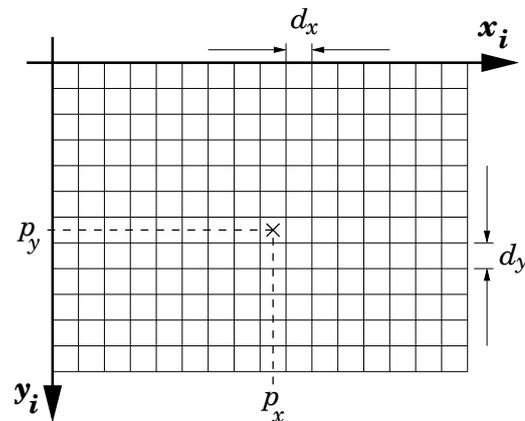


Abbildung 6.2: CCD-Koordinatensystem (Pixel).

fällt der Hauptpunkt nicht mehr mit dem Ursprung des Bildkoordinatensystems zusammen sondern befindet sich an der Stelle  $(p_x, p_y)^T$ .

Sind der horizontale und vertikale Abstand benachbarter Pixel auf dem Chip,  $d_x$  und  $d_y$ , nicht identisch, muß die Brennweite in  $x$  und  $y$  getrennt betrachtet werden.

$$[d_x] = [d_y] = \frac{[X_K]}{\text{Pixel}}, \quad x_i = p_x + \frac{x_B}{d_x}, \quad y_i = p_y - \frac{y_B}{d_y}, \quad f_x = \frac{f}{d_x}, \quad f_y = \frac{f}{d_y} \quad (6.3)$$

Die Brennweiten werden ab jetzt, wie die Bildkoordinaten, in Pixeln angegeben. Gleichung (6.2) wird dann zu

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} X_K \\ Y_K \\ Z_K \\ 1 \end{pmatrix}, \quad x_i = \frac{x}{w}, \quad y_i = \frac{y}{w}. \quad (6.4)$$

Bis hierhin hat eine Kamera vier Parameter, die bei der Kalibrierung bestimmt werden müssen:  $f_x$ ,  $f_y$ ,  $p_x$  und  $p_y$ . Als Alternative zu den zwei Brennweiten können eine Brennweite und ein Seitenverhältnis verwendet werden.

## 6.1.2 Externe Parameter

Im vorangegangenen Abschnitt wurde für die 3D-Punkte immer das lokale Koordinatensystem der Kamera verwendet. Zur Kalibrierung werden Punkte benötigt, von denen man sowohl die 3D- als auch die 2D-Koordinaten weiß, worauf in den Abschnitten 6.2, 6.3 und 6.4 näher eingegangen wird. Zur vollständigen Kalibrierung

gehört neben den internen Parametern ebenfalls, die Transformation zwischen der Kamera und dem Koordinatensystem des Kalibrierkörpers (oder einem globalen) zu bestimmen, in dem die 3D-Punkte gegeben sind.

Die Translation der Kamera hat drei Freiheitsgrade, sie kann zum Beispiel durch die Position ihres Zentrums beschrieben werden. Die Rotationsmatrix  $\mathbf{R} \in SO_3$  besitzt ebenfalls drei Freiheitsgrade, ist orthogonal und ihre Determinante beträgt eins. In inhomogenen Koordinaten lautet die Beziehung zwischen den Weltkoordinaten  $(X, Y, Z)$  und den Koordinaten im Kamerasystem

$$\begin{pmatrix} X_K \\ Y_K \\ Z_K \end{pmatrix} = \mathbf{R} \begin{pmatrix} X - X_C \\ Y - Y_C \\ Z - Z_C \end{pmatrix}, \quad \mathbf{C} = \begin{pmatrix} X_C \\ Y_C \\ Z_C \end{pmatrix}, \quad (6.5)$$

Gleichung (6.4) erweitert sich zu

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \begin{bmatrix} f_x & 0 & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix} \mathbf{R} [\mathbf{I} | -\mathbf{C}] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \quad (6.6)$$

$\mathbf{I}$  steht für die  $3 \times 3$  Einheitsmatrix. Abbildung 6.3 zeigt noch einmal alle bei der Abbildung beteiligten Koordinatensysteme.

### 6.1.3 Projektionsmatrix und ihre Zerlegung

In der allgemeinsten Form läßt sich die lineare Abbildung der Kamera durch eine homogene  $3 \times 4$  Matrix  $\mathbf{P}$  beschreiben

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \mathbf{P} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}, \quad x_i = \frac{x}{w}, \quad y_i = \frac{y}{w}. \quad (6.7)$$

Sie ist bis auf einen Skalierungsfaktor bestimmt, besitzt also 11 Freiheitsgrade. Die vier internen und sechs externen Parameter ergeben zusammen aber nur zehn, woraus besteht der elfte Freiheitsgrad?

Zerlegt man die linke  $3 \times 3$  Matrix von  $\mathbf{P}$  mit einer RQ-Zerlegung [20, 21] in das Produkt aus einer oberen Dreiecksmatrix  $\mathbf{K}$  und einer orthogonalen Matrix  $\mathbf{R}$ , ergibt sich (vgl. Gleichung (6.6))

$$\mathbf{P} = \mathbf{K}[\mathbf{R} | \mathbf{t}] \quad \text{mit} \quad \mathbf{t} = -\mathbf{R}\mathbf{C}. \quad (6.8)$$

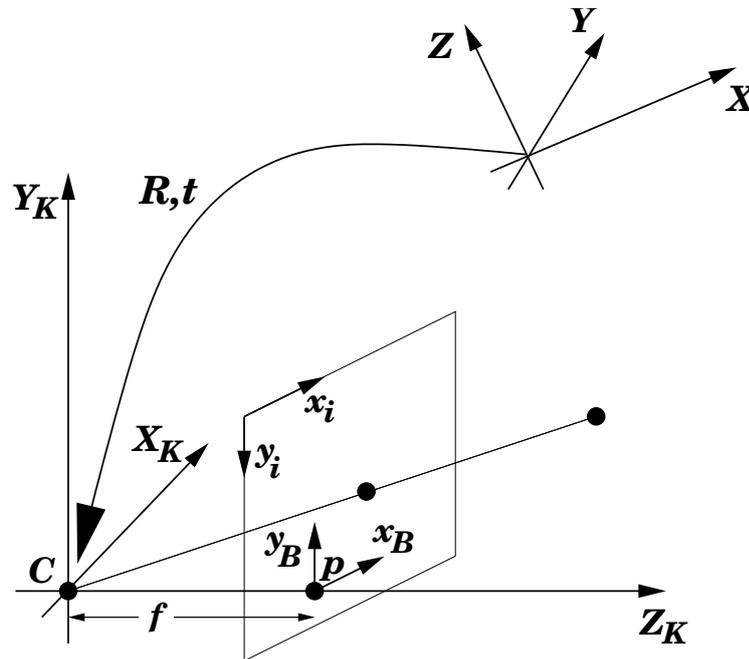


Abbildung 6.3: Koordinatensysteme bei der Abbildung durch eine Lochkamera.

Nach Wahl des Skalierungsfaktors, so daß  $k_{33} = 1$  wird, erhält man

$$\mathbf{K} = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (6.9)$$

Der elfte Freiheitsgrad ist die Schiefe  $s$ , die eine Verkippung der Achsen der Bildkoordinaten gegeneinander beschreibt. Bei allen industriell gefertigten CCD-Chips stehen die Achsen praktisch senkrecht aufeinander, was gleichbedeutend ist mit  $s = 0$ .

## 6.2 Direkte Lineare Kalibrierung

Eine lineare Methode, die Projektionsmatrix einer Kamera aus mit ihr aufgenommenen Bildern zu bestimmen, wurde in [1] vorgestellt. Obwohl sie später um eine nichtlineare Optimierung (Abschnitt 6.2.4) erweitert wurde, ist der Name „Direct Linear Transformation“ (DLT) geblieben [37].

Gleichung (6.7) läßt sich wie folgt umschreiben,  $p_{ij}$  bezeichnet dabei das Matrixelement in Zeile  $i$  und Spalte  $j$  der Matrix  $\mathbf{P}$

$$x_i = \frac{p_{11}X + p_{12}Y + p_{13}Z + p_{14}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}, \quad (6.10)$$

$$y_i = \frac{p_{21}X + p_{22}Y + p_{23}Z + p_{24}}{p_{31}X + p_{32}Y + p_{33}Z + p_{34}}. \quad (6.11)$$

Jeder 3D-Punkt, dessen Welt- und Bildkoordinaten man kennt, ergibt damit zwei lineare Bedingungen für die Elemente von  $\mathbf{P}$ .

$$\begin{bmatrix} -X & -Y & -Z & -1 & 0 & 0 & 0 & 0 & x_iX & x_iY & x_iZ & x_i \\ 0 & 0 & 0 & 0 & -X & -Y & -Z & -1 & y_iX & y_iY & y_iZ & y_i \end{bmatrix} \begin{pmatrix} p_{11} \\ \vdots \\ p_{34} \end{pmatrix} = \mathbf{0} \quad (6.12)$$

### 6.2.1 Exakte Lösung

Durch mehrere solcher Zwangsbedingungen ergibt sich ein lineares Gleichungssystem  $\mathbf{A}\mathbf{p} = \mathbf{0}$ . Um es exakt lösen zu können, sind  $n = 5^{1/2}$  Punkte nötig, vom sechsten wird nur eine der beiden Zeilen benötigt. Die Matrix  $\mathbf{A}$  hat dann die Dimension  $11 \times 12$  und einen Rang von 11, natürlich vorausgesetzt, die einzelnen Zeilen sind linear unabhängig.

Es gibt einige Sonderfälle, in denen diese Voraussetzung nicht erfüllt ist, eine genaue Analyse findet sich in [64]. Wichtig ist, daß die Punkte ein Volumen aufspannen müssen. Je „weniger koplanar“ sie sind, desto besser konditioniert ist das entstehende Gleichungssystem. Liegen alle Punkte in einer Ebene, hat  $\mathbf{A}$  nur noch den Rang 8 und es gibt keine eindeutige Lösung für  $\mathbf{P}$ . In Abschnitt 6.2.5 wird näher darauf eingegangen, was dies für die Verwendbarkeit in der Praxis bedeutet.

Eine Möglichkeit zur Bestimmung von  $\mathbf{P}$  ist, eines der zwölf Elemente vorzugeben und die restlichen elf aus dem linearen Gleichungssystem zu berechnen, in [1] wird  $p_{34} = 1$  gewählt. Die Methode funktioniert nicht, wenn das Element Null sein sollte. Das im nächsten Abschnitt vorgestellte Verfahren hat diese Einschränkung nicht und läßt sich im Fall minimaler Anzahl von Gleichungen ebenfalls verwenden.

### 6.2.2 Überbestimmtes System

Da die Bildkoordinaten aufgrund von Rauschen nicht exakt gemessen werden können, ist es besser mehr als die absolut nötige Anzahl von Punkten zu benutzen. Ab  $n \geq 6$  ist das Gleichungssystem überbestimmt,  $\mathbf{A}$  hat die Dimension  $2n \times 12$ , jedoch wird der Rang aufgrund der Meßfehler nicht 11 sondern 12 sein. Gesucht ist nun der Vektor  $\mathbf{p}$ , der  $\|\mathbf{A}\mathbf{p}\|$  minimiert. Um die unerwünschte triviale Lösung auszuschließen, muß eine Nebenbedingung gestellt werden.

Ein geeignetes und numerisch günstiges Verfahren zum Lösen derartiger Probleme ist die Singulärwertzerlegung („Singular Value Decomposition“ SVD [21]), die

implizit die Nebenbedingung  $\|\mathbf{p}\| = 1$  garantiert. Die Matrix wird in ein Produkt aus drei Matrizen zerlegt:

$$\mathbf{A} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^T, \quad \mathbf{\Sigma} = \text{diag}(\sigma_1, \dots, \sigma_n), \quad \sigma_1 \geq \dots \geq \sigma_n, \quad (6.13)$$

$\mathbf{U}$  hat die Dimension  $2n \times 12$  und orthogonale Spalten,  $\mathbf{V}$  hat  $12 \times 12$  Elemente und ist orthogonal, und  $\mathbf{\Sigma}$  ist eine  $12 \times 12$  Elemente große Diagonalmatrix, die die Singulärwerte von  $\mathbf{A}$  enthält. Für diese Arbeit wurde die Implementierung der SVD aus [56] verwendet.

Lösung ist der Spaltenvektor von  $\mathbf{V}$ , der zum kleinsten Singulärwert gehört. Sind die Singulärwerte absteigend sortiert, wie in Gleichung (6.13) angedeutet, ist es die letzte Spalte von  $\mathbf{V}$ . Gegenüber den anderen Singulärwerten wird  $\sigma_n$  sehr klein sein, und zwar um so kleiner, je geringer die Meßfehler waren. Setzt man  $\sigma_n$  in  $\mathbf{\Sigma}$  auf Null und bildet das Produkt  $\mathbf{U}\text{diag}(\sigma_1, \dots, \sigma_{n-1}, 0)\mathbf{V}^T$ , erhält man diejenige Matrix vom Rang 11, die bezüglich der Frobeniusnorm am nächsten an  $\mathbf{A}$  liegt.

### 6.2.3 Normalisierung

Die Wahl des Nullpunktes des Koordinatensystems der Bilder sollte keinen Einfluß auf die Kameraparameter haben. Verschiebt er sich ( $\tilde{\mathbf{x}}_i = \mathbf{x}_i + \delta_{\mathbf{x}}$ ), sollten alle Kameraparameter gleich bleiben, nur der Hauptpunkt ändert sich um das selbe  $\delta_{\mathbf{x}}$ . Im allgemeinen wird dies bei der Berechnung mit der oben vorgestellten Methode jedoch nicht der Fall sein. Der Grund dafür ist, daß die Nebenbedingung der Minimierung nicht in jedem Koordinatensystem zum selben Minimum führt [23]. Analoges gilt für die Wahl des Koordinatensystems der 3D-Punkte.

Abhilfe schafft, vor der Minimierung eine Normalisierung der Daten durchzuführen, die dafür sorgt daß der Schwerpunkt der Punkte jeweils in den Ursprung verlegt wird. Die Bildpunkte sollen einen durchschnittlichen Abstand von  $\sqrt{2}$  und die 3D-Punkte einen durchschnittlichen Abstand von  $\sqrt{3}$  vom Ursprung haben. Die „mittleren Koordinaten“ sind dann  $(1, 1)^T$  beziehungsweise  $(1, 1, 1)^T$ . Alle Punktwolken, die sich nur durch eine Skalierung und Translation voneinander unterscheiden, werden dadurch in eine kanonische Position gebracht und das Ergebnis der Minimierung ist unabhängig von diesen Transformationen.

$$\forall i = 1 \dots n : \quad \tilde{\mathbf{x}}_i = \mathbf{T}_{\mathbf{B}}\mathbf{x}_i, \quad \tilde{\mathbf{X}}_i = \mathbf{T}_{\mathbf{W}}\mathbf{X}_i, \quad \tilde{\mathbf{x}}_i = \tilde{\mathbf{P}}\tilde{\tilde{\mathbf{x}}}_i \quad (6.14)$$

Danach muß die Normalisierung rückgängig gemacht werden, um das für die ursprünglichen Koordinaten geltende  $\mathbf{P}$  zu erhalten

$$\mathbf{P} = \mathbf{T}_{\mathbf{B}}^{-1}\tilde{\mathbf{P}}\mathbf{T}_{\mathbf{W}}. \quad (6.15)$$

Die Normalisierung hilft nicht nur, das Ergebnis unabhängig von Skalierung und Translation der Punkte zu machen, sie erhöht auch die numerische Stabilität, weil sie die Kondition der Matrix  $\mathbf{A}$  erheblich verbessert. Zum einen wirken sich Meßfehler um so stärker aus, je schlechter das Gleichungssystem konditioniert ist, zum anderen ist es bei der Berechnung mit Fließkommaarithmetik besser, wenn die Exponenten aller Einträge der Matrix in der selben Größenordnung liegen und bei Additionen und Subtraktionen weniger Stellen ausgelöscht werden.

### 6.2.4 Minimierung des Reprojektionsfehlers

Besser als die Minimierung eines algebraischen Fehlermaßes ist, den Fehler direkt in den gemessenen Bildern zu minimieren

$$\min_{\mathbf{P}} \sum_i \|\mathbf{x}_i - \mathbf{P}\mathbf{X}_i\|_2. \quad (6.16)$$

Das Ergebnis ist zusätzlich unabhängig von Rotationen der Koordinatensysteme, da diese den euklidischen Abstand zwischen zwei Punkten nicht ändern. Die Minimierung ist allerdings nicht mehr linear, sondern muß iterativ durchgeführt werden, wofür ein Startwert benötigt wird, der im Konvergenzbereich des gesuchten Minimums liegt. Sinnvoll ist, das Ergebnis der linearen Methode als Startwert zu verwenden. Zur nichtlinearen Optimierung dieser Art von Problem ist zum Beispiel das Levenberg-Marquardt Verfahren gut geeignet [56], unter anderem weil ihm die Überparametrisierung (12 Matrixelemente werden geschätzt,  $\mathbf{P}$  hat aber nur 11 Freiheitsgrade) keine Probleme bereitet.

Nimmt man an, daß die Fehler in den gemessenen Bildkoordinaten mittelwertfreien Normalverteilungen mit identischer Standardabweichung unterliegen und statistisch voneinander unabhängig sind, ergibt die Minimierung der Fehlerquadrate genau die Maximum-Likelihood Schätzung der Elemente von  $\mathbf{P}$  [8]. Hat man dagegen eine Schätzung für die Kovarianzmatrix  $\mathbf{\Sigma}$  der Punkte, kann statt der euklidischen die Mahalanobisdistanz minimiert werden, die die Korrelationen zwischen den Variablen und unterschiedliche Standardabweichungen berücksichtigt

$$\min_{\mathbf{P}} \sum_i \|\mathbf{x}_i - \mathbf{P}\mathbf{X}_i\|_{\mathbf{\Sigma}} = \min_{\mathbf{P}} \sum_i \sqrt{(\mathbf{x}_i - \mathbf{P}\mathbf{X}_i)^T \mathbf{\Sigma}^{-1} (\mathbf{x}_i - \mathbf{P}\mathbf{X}_i)}. \quad (6.17)$$

### 6.2.5 Praxistauglichkeit

In Abschnitt 6.2.1 wurde bereits darauf hingewiesen, daß die Punkte eines Kalibrierkörpers ein Volumen aufspannen müssen, damit die Projektionsmatrix bestimmt werden kann. Für die Verwendung des Verfahrens in der Praxis bedeutet dies eine

große Einschränkung, denn geeignete Kalibrierkörper sind nur aufwendig herzustellen. In [23] findet sich die Daumenregel, daß die Anzahl der Nebenbedingungen fünfmal größer sein sollte als die Anzahl der zu schätzenden Parameter (11 in diesem Fall). Ein Kalibrierkörper sollte also aus mindestens 28 Punkten bestehen, von denen möglichst viele aus allen Richtungen eindeutig in den Bildern identifiziert werden können. Ein weiteres Problem ist, daß die Kalibrierung umso genauer ist, je besser die Punkte das spätere Meßvolumen ausfüllen, was bedeutet daß für verschiedene Meßvolumina auch unterschiedliche Kalibrierkörper benötigt werden.

## 6.3 Verfahren von Tsai

Ebene Kalibrierkörper sind wesentlich einfacher herzustellen als volumetrische. In [66] wurde von R. Tsai eine Methode vorgestellt, die mit der Aufnahme einer oder mehrerer Ebenen zur Kalibrierung auskommt. Ziel der Arbeit war, eine Methode zu entwickeln, die den folgenden Anforderungen genügt:

- Automatisches Verfahren
- Hohe Genauigkeit
- Kurze Meß- und Rechenzeit
- Universalität
- Verwendbarkeit für eine große Bandbreite an Kameras

In der genannten Arbeit wird ein umfassender Überblick über die bis dahin existierenden Kalibriermethoden gegeben und warum sich mit ihnen nicht alle gesteckten Ziele erreichen lassen. Nicht zuletzt aufgrund der Verfügbarkeit einer Implementierung in C [70] wird die Kalibrierung nach Tsai häufig verwendet.

Das verwendete Kameramodell weicht bei den internen Parametern leicht von den in Abschnitt 6.1.1 besprochenen ab. Die Brennweite wird in der Einheit der Weltkoordinaten bestimmt und die Abmessungen  $d_x$  und  $d_y$  sowie die Koordinaten des Hauptpunktes müssen bekannt sein. Während die Bildmitte als Näherung für  $\mathbf{p}$  empfohlen wird, sind die physikalischen Abmessungen der CCD-Pixel vor allem bei nicht für den industriellen Einsatz gedachten Kameras nur schwer mit ausreichender Genauigkeit zu ermitteln.

### 6.3.1 Kalibriermuster

Die Anforderungen an das Kalibriermuster sind einfache Herstellbarkeit und ausreichende Genauigkeit. Es muß sich um eine Ebene handeln, auf der sich mehrere

automatisch eindeutig identifizierbare Punkte befinden. Tsai schlägt in seiner Arbeit vor, ein regelmäßiges Gitter von schwarzen Quadraten auf weißem Grund zu verwenden. Die Eckpunkte der Quadrate müssen möglichst genau gefunden werden. Zunächst binarisiert man das Bild mit einem Schwellwert und verbindet die Kantenpixel miteinander. Die so gefundene Schätzung für die Kantenposition wird auf Subpixelgenauigkeit verbessert, indem nach der Position des größten Grauwertanstiegs senkrecht zur Kante gesucht wird. Der Eckpunkt ist der Schnittpunkt je einer waagerechten und einer senkrechten Regressionsgeraden durch diese Punkte. Andere Arbeiten (z.B. [7]) verwenden den Canny-Kantendetektor [11] anstelle der Binarisierung und senkrechten Suche.

### 6.3.2 Linsenverzerrung

Will man mit der Kamera genaue Messungen durchführen, muß man nichtlineare Effekte berücksichtigen, die von der Abweichung der Linse vom Ideal der Lochkamera herrühren. Abbildung 6.4(a) zeigt das Beispiel einer tonnenförmigen Verzerrung, die bei Kameras mit kurzer Brennweite häufig vorkommt. Geraden werden dabei nicht als Geraden abgebildet. Sind die Eigenschaften der Verzerrung bekannt, kann das Bild mit einer inversen Transformation entzerrt werden, so daß zwischen den Weltkoordinaten und dem entzerrten Bild wieder ein lineares Kameramodell verwendet werden kann (Abb. 6.4(b)).

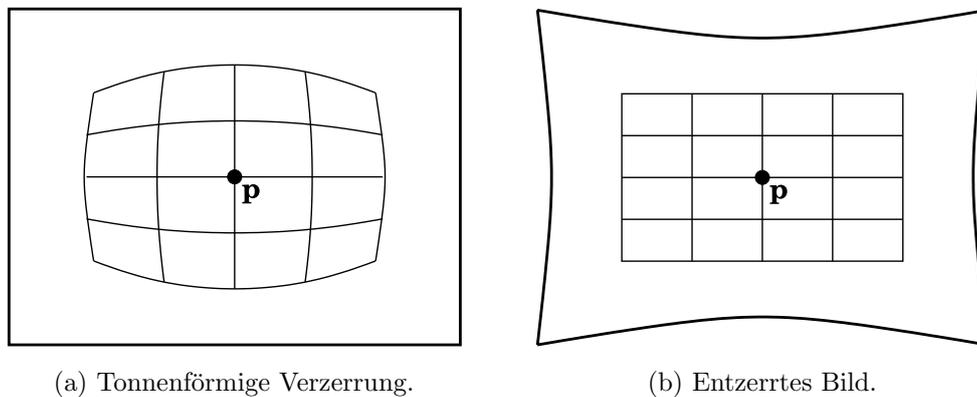


Abbildung 6.4: Linsenverzerrung.

In [66] wird ein rein radiales Modell verwendet, bei dem der Hauptpunkt unverändert bleibt und alle anderen Punkte entlang der Strahlen wandern, die sie mit dem Hauptpunkt verbinden. Die entzerrten Koordinaten  $x_u$  und  $y_u$  ergeben sich aus den

beobachteten Bildkoordinaten  $x_B$  und  $y_B$  durch

$$\begin{aligned}x_u &= x_B(1 + \kappa_1 r^2 + \kappa_2 r^4 + \dots) \\y_u &= y_B(1 + \kappa_1 r^2 + \kappa_2 r^4 + \dots) \\r &= \sqrt{x_B^2 + y_B^2}.\end{aligned}\tag{6.18}$$

Geschätzt wird nur  $\kappa_1$ , da der quadratische Term den bei weitem größten Teil der Verzerrung erklärt und die Anzahl der zu schätzenden Parameter bei der verwendeten nichtlinearen Optimierung so klein wie möglich gehalten werden sollte.

Die in [66] gemachte Aussage, daß die Genauigkeit nicht wesentlich leide wenn der Bildmittelpunkt als Hauptpunkt fungiert, wird in [46] revidiert. Die Annahme, daß der Hauptpunkt nie mehr als 10 Pixel vom Bildmittelpunkt entfernt sei und damit nicht als Parameter geschätzt werden muß, hat sich in der Praxis als nicht haltbar erwiesen. Eine Möglichkeit ist,  $p_x$  und  $p_y$  in der Optimierung in Schritt 3 des nächsten Abschnittes mitzuschätzen, was allerdings voraussetzt daß die Linsenverzerrung eine gewisse Mindestgröße hat.

### 6.3.3 Ablauf der Kalibrierung

Im folgenden wird nur der prinzipielle Ablauf des Verfahrens skizziert, weil aufgrund der besseren Eigenschaften die in Abschnitt 6.4 ausführlich beschriebene Methode zur Kalibrierung der Stereokamera verwendet wird. Ausgangspunkt ist der sogenannte „Radial Alignment Constraint“ (RAC). Er ist die formale Beschreibung der Tatsache, daß die drei Vektoren  $(x_B, y_B)^T$ ,  $(x_u, y_u)^T$  und  $(X_K, Y_K)^T$  linear abhängig sind, und zwar unabhängig von  $\kappa_1$ ,  $f$  und  $t_z$  (vgl. Abb. 6.1 und Gl. (6.18)).

1. Die Nebenbedingungen, die sich aus dem RAC ergeben, führen auf ein lineares Gleichungssystem für  $t_x$ ,  $t_y$  und vier Elemente der Rotationsmatrix. Es werden mindestens fünf Punkte benötigt, bei mehr als fünf ist das System überbestimmt (vgl. Abschnitt 6.2.2). Die restlichen Elemente von  $\mathbf{R}$  lassen sich aus der Orthogonalität der Matrix herleiten.
2. Aus einem zweiten linearen Gleichungssystem werden Näherungen für  $t_z$  und  $f$  berechnet. Dabei ist wichtig, daß die Kalibrierebene nicht parallel zur Bildebene verläuft, da das System sonst linear abhängig wird und sich die beiden Parameter nicht voneinander trennen lassen. Als optimal hat sich ein Winkel von mindestens  $30^\circ$  erwiesen [66, 7].
3. Eine nichtlineare Optimierung mit  $\kappa_1 = 0$  und den linear berechneten Parametern als Startwert schließt die Berechnung ab.

### 6.3.4 Der unbekannte Skalierungsfaktor

Der kritischste Parameter des Modells ist ein Skalierungsfaktor  $s_x$ , der bekannt sein muß und dessen Herkunft und Rolle bei der Kalibrierung im Folgenden erläutert werden. Die als Ladung gespeicherten Helligkeitsinformationen einer Zeile werden im Auslesetakt durch eine Eimerkettenschaltung in eine Spannung umgewandelt. Unabhängig von der Anzahl der Sensorpixel wird das Ausgangssignal dabei so generiert, daß es in Spannungspegeln und Timing einem Standard entspricht, zum Beispiel RS-170, PAL oder NTSC. Das Videosignal wird dann nach der Übertragung von einem Framegrabber wieder digitalisiert, der für jede Zeile mit einem Analog/Digital-Wandler eine feste Anzahl digitaler Helligkeitswerte generiert. Der Faktor  $s_x$  erfaßt nun das Verhältnis zwischen den physikalischen Pixeln, deren Abmessungen gegeben sind, und den digitalisierten Pixeln, in denen im Bild gemessen wird.

Werden mehrere Ebenen verwendet, kann  $s_x$  bestimmt werden. Leider ist es notwendig, daß sich die Ebenen nur um eine Verschiebung in  $Z$ -Richtung voneinander unterscheiden, was in [66] durch einen Linearvorschub bewerkstelligt wurde. Außer in speziellen industriellen Umgebungen sind solche Vorschübe nicht vorhanden und eine Anschaffung nur zum Zweck der Kalibrierung aus Preisgründen nicht sinnvoll, daher ist die Variante mit mehreren parallelen Ebenen nicht universell einsetzbar.

Bei Verwendung nur einer Kalibrierebene muß  $s_x$  sehr genau bekannt sein, ein Fehler von einem Prozent bewirkt zum Beispiel bei 720 Pixeln Bildbreite schon 7 Pixel Differenz am Rand des Bildes. Mehrere Methoden, diesen Faktor vor der eigentlichen Kalibrierung zu bestimmen, stellt [46] vor. Die einfachste, das Verhältnis zwischen CCD-Pixeln und Bildpixeln, ergab im betrachteten Fall einen Fehler von 4% und ist damit inakzeptabel. Die zweite Möglichkeit, das Verhältnis der Takte des CCD-Auslesens und der A/D-Wandlung zu verwenden, ergibt zwar ein sehr genaues Ergebnis, setzt aber eine intime Kenntnis der Kamera- und Framegrabber Interna voraus, die nicht in allen Fällen vorhanden ist. Die dritte Möglichkeit, das Verhältnis aus dem Interferenzmuster einer Aufnahme bei abgeschaltetem Tiefpaß vor dem A/D-Wandler zu berechnen, erfordert einen Eingriff in den Framegrabber und scheidet daher ebenfalls als nicht praktikabel aus.

### 6.3.5 Eignung für die Kalibrierung der Stereokamera

Da die Kalibrierung mit mehreren Ebenen wie bereits erwähnt praktisch genauso aufwendig ist wie die Verwendung eines echten dreidimensionalen Kalibrierkörpers, kommt nur die Variante mit einer Ebene in Betracht. Damit stellt sich aber das schwierige Problem der Ermittlung von  $s_x$ . Die Kalibrierung ließe sich nur für Ka-

meras anwenden, bei denen die Pixelabmessungen und der Auslesetakts sehr genau bekannt sind, was die Zahl der verwendbaren Modelle drastisch einschränkt. In der Literatur finden sich zudem Hinweise darauf, daß die monoplanare Kalibrierung keine sehr stabilen und genauen Ergebnisse liefert [6], was durch von uns durchgeführte Tests bestätigt wurde.

## 6.4 Verfahren von Zhang

Eine flexiblere und zugleich stabilere Technik zur Kamerakalibrierung wird von Z. Zhang unter anderem in [76] beschrieben. Als Kalibriermuster wird ebenfalls eine Ebene verwendet, die in mehreren beliebigen Orientierungen aufgenommen wird. Die internen Parameter werden in Pixelkoordinaten bestimmt, was das Ermitteln der physikalischen Parameter des Sensors und den Skalierungsfaktor überflüssig macht. Die Brennweite des Lochkammermodells ist ohnehin eine virtuelle Größe und kann nicht direkt mit der reellen Brennweite des Linsensystems verglichen werden.

Mit der Open Computer Vision Library (OpenCV [31]) steht eine Implementierung in C++ zur Verfügung, die Routinen für das Suchen der 2D-Punkte in den Bildern und Kalibrieren einer einzelnen Kamera zur Verfügung stellt.

### 6.4.1 Kalibriermuster

Die OpenCV erwartet ein Schachbrettmuster, dessen Kreuzungspunkte detektiert und in eine bestimmte Reihenfolge gebracht werden, so daß die 3D-Koordinaten sehr einfach zugewiesen werden können. Ein Beispiel ist in Abbildung 6.7 auf Seite 64 zu sehen.

Wird als Kalibrierebene o.B.d.A. die Ebene  $Z = 0$  gewählt, und bezeichnet  $\mathbf{r}_i$  die  $i$ -te Spalte der Rotationsmatrix  $\mathbf{R}$ , so gilt (vgl. Gl. (6.8))

$$\begin{pmatrix} x \\ y \\ w \end{pmatrix} = \mathbf{K} [\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{r}_3 \ t] \begin{pmatrix} X \\ Y \\ 0 \\ 1 \end{pmatrix} = \mathbf{K} [\mathbf{r}_1 \ \mathbf{r}_2 \ t] \begin{pmatrix} X \\ Y \\ 1 \end{pmatrix}. \quad (6.19)$$

### 6.4.2 Aufstellen der Bestimmungsgleichung

Die Beziehung zwischen der Kalibrierebene und ihrem Bild wird durch eine ebene Homographie beschrieben:

$$\lambda \mathbf{K} [\mathbf{r}_1 \ \mathbf{r}_2 \ \mathbf{t}] = \mathbf{H} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}. \quad (6.20)$$

$\mathbf{H}$  ist bis auf einen Skalierungsfaktor bestimmt, besitzt also 8 Freiheitsgrade. Die Homographie kann unter Berücksichtigung von  $Z = 0$  mit Hilfe einer reduzierten Version der Gleichung (6.12) aus den Bild- und Weltkoordinaten bestimmt werden. Es gelten sinngemäß alle Betrachtungen, die zum DLT Algorithmus gemacht wurden, zur Bestimmung von  $\mathbf{H}$  werden mindestens 4 Punkte benötigt.

Als Spalten einer Rotationsmatrix sind  $\mathbf{r}_1$  und  $\mathbf{r}_2$  orthonormal, daher ergeben sich zwei Bedingungen:

$$(h_{11} \ h_{21} \ h_{31}) \mathbf{K}^{-\mathbf{T}} \mathbf{K}^{-1} \begin{pmatrix} h_{12} \\ h_{22} \\ h_{32} \end{pmatrix} = 0 \quad \text{und} \quad (6.21)$$

$$(h_{11} \ h_{21} \ h_{31}) \mathbf{K}^{-\mathbf{T}} \mathbf{K}^{-1} \begin{pmatrix} h_{11} \\ h_{21} \\ h_{31} \end{pmatrix} = (h_{12} \ h_{22} \ h_{32}) \mathbf{K}^{-\mathbf{T}} \mathbf{K}^{-1} \begin{pmatrix} h_{12} \\ h_{22} \\ h_{32} \end{pmatrix}. \quad (6.22)$$

Dabei ist  $\mathbf{K}^{-\mathbf{T}} \mathbf{K}^{-1}$  eine symmetrische  $3 \times 3$  Matrix

$$\mathbf{K}^{-\mathbf{T}} \mathbf{K}^{-1} = \omega = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{12} & \omega_{22} & \omega_{23} \\ \omega_{13} & \omega_{23} & \omega_{33} \end{bmatrix}. \quad (6.23)$$

Gleichung (6.21) und (6.22) ergeben zwei Zeilen eines homogenen linearen Gleichungssystems für die sechs Elemente von  $\omega$  pro Homographie, also aufgenommenem Bild

$$\begin{bmatrix} h_{11}h_{12} & h_{11}h_{22}+h_{21}h_{12} & h_{11}h_{32}+h_{31}h_{12} \\ h_{12}h_{12}-h_{11}h_{11} & h_{12}h_{22}+h_{22}h_{12}-h_{11}h_{21}+h_{21}h_{11} & h_{12}h_{32}+h_{32}h_{12}-h_{11}h_{31}+h_{31}h_{11} \\ h_{21}h_{22} & h_{21}h_{32}+h_{31}h_{22} & h_{31}h_{32} \\ h_{22}h_{22}-h_{21}h_{21} & h_{22}h_{32}+h_{32}h_{22}-h_{21}h_{31}+h_{31}h_{21} & h_{32}h_{32}-h_{31}h_{31} \end{bmatrix} \begin{pmatrix} \omega_{11} \\ \vdots \\ \omega_{33} \end{pmatrix} = \mathbf{0}. \quad (6.24)$$

Für  $n \geq 3$  Bilder ist das System genau beziehungsweise überbestimmt. Lösung ist jeder nichttriviale Vektor aus dem rechten Nullraum der  $2n \times 6$  Elemente großen

Matrix, der sich zum Beispiel wie in Abschnitt 6.2.2 beschrieben durch Singulärwertzerlegung der Matrix bestimmen läßt.

Für den Fall  $n = 2$  können nicht alle fünf internen Parameter bestimmt werden, da nur vier Bedingungen gegeben sind. Durch Hinzufügen der Zeile  $[0 \ 1 \ 0 \ 0 \ 0 \ 0]$ , die der zusätzlichen Bedingung  $s = 0$  entspricht (vgl. Abschn. 6.1.3, ist die Berechnung der restlichen vier dennoch möglich.

### 6.4.3 Bestimmung der internen Parameter

Die interne Kalibrierung  $\mathbf{K}$  kann entweder über eine Cholesky Zerlegung von  $\omega$ , oder wie im Folgenden beschrieben, durch direktes Ausrechnen der einzelnen Parameter bestimmt werden. Ausmultiplizieren von Gleichung (6.23) unter Berücksichtigung von Gleichung (6.9) ergibt [23]

$$\begin{aligned} \omega &= \lambda \mathbf{K}^{-\mathbf{T}} \mathbf{K}^{-1} \\ &= \lambda \begin{bmatrix} \frac{1}{f_x^2} & -\frac{s}{f_x^2 f_y} & \frac{p_y s - p_x f_y}{f_x^2 f_y} \\ -\frac{s}{f_x^2 f_y} & \frac{s^2}{f_x^2 f_y^2} + \frac{1}{f_y^2} & -\frac{s(p_y s - p_x f_y)}{f_x^2 f_y^2} - \frac{p_y}{f_y^2} \\ \frac{p_y s - p_x f_y}{f_x^2 f_y} & -\frac{s(p_y s - p_x f_y)}{f_x^2 f_y^2} - \frac{p_y}{f_y^2} & \frac{(p_y s - p_x f_y)^2}{f_x^2 f_y^2} + \frac{p_y^2}{f_y^2} + 1 \end{bmatrix}, \end{aligned} \quad (6.25)$$

woraus sich die Parameter berechnen lassen [76]:

$$\begin{aligned} p_y &= \frac{\omega_{12}\omega_{13} - \omega_{11}\omega_{23}}{\omega_{11}\omega_{22} - \omega_{12}^2} \\ \lambda &= \omega_{33} - \frac{\omega_{13}^2 + p_y(\omega_{12}\omega_{13} - \omega_{11}\omega_{23})}{\omega_{11}} \\ f_x &= \sqrt{\frac{\lambda}{\omega_{11}}} \\ f_y &= \sqrt{\frac{\lambda\omega_{11}}{\omega_{11}\omega_{22} - \omega_{12}^2}} \\ s &= -\frac{1}{\lambda}\omega_{12}f_x^2 f_y \\ p_x &= \frac{sp_y}{f_x} - \frac{\omega_{13}f_x^2}{\lambda}. \end{aligned} \quad (6.26)$$

### 6.4.4 Bestimmung der externen Parameter

Ist  $\mathbf{K}$  bekannt, können die externen Kameraparameter  $\mathbf{R}$  und  $\mathbf{t}$  aus Gleichung (6.20) berechnet werden, indem korrespondierende Spalten der linken und rechten Seite

gleichgesetzt werden:

$$\mathbf{r}_1 = \frac{1}{\lambda} \mathbf{K}^{-1} \begin{pmatrix} h_{11} \\ h_{21} \\ h_{31} \end{pmatrix} \quad \text{und} \quad \mathbf{r}_2 = \frac{1}{\lambda} \mathbf{K}^{-1} \begin{pmatrix} h_{12} \\ h_{22} \\ h_{32} \end{pmatrix}. \quad (6.27)$$

$\lambda$  ergibt sich aus  $\|\mathbf{r}_1\| = \|\mathbf{r}_2\| = 1$ . Schließlich fehlen noch

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2 \quad \text{und} \quad \mathbf{t} = \frac{1}{\lambda} \mathbf{K}^{-1} \begin{pmatrix} h_{13} \\ h_{23} \\ h_{33} \end{pmatrix}. \quad (6.28)$$

### 6.4.5 Linsenverzerrung

Während [76] nur die beiden radialen Koeffizienten  $\kappa_1$  und  $\kappa_2$  aus Gleichung (6.18) verwendet, bezieht die Implementierung der OpenCV auch nicht rein radiale Komponenten mit ein.

$$\begin{aligned} r^2 &= (x_i - p_x)^2 + (y_i - p_y)^2 \\ \tilde{x}_i &= x_i + (x_i - p_x) \left[ \kappa_1 r^2 + \kappa_2 r^4 + 2\rho_1 y_i + \rho_2 \left( \frac{r^2}{x_i} + 2x_i \right) \right] \\ \tilde{y}_i &= y_i + (y_i - p_y) \left[ \kappa_1 r^2 + \kappa_2 r^4 + 2\rho_2 x_i + \rho_1 \left( \frac{r^2}{y_i} + 2y_i \right) \right] \end{aligned} \quad (6.29)$$

Die Koeffizienten  $\rho_1$  und  $\rho_2$  modellieren Verzerrungen, die auftreten wenn Linse und CCD-Chip nicht parallel zueinander angeordnet sind [5]. Die durchgeführten Messungen belegen allerdings, daß zumindest die verwendeten Kameras damit überparametrisiert sind, da  $\rho_1$  und  $\rho_2$  fast Null und ihre Standardabweichungen sehr groß im Verhältnis zu den Werten sind (Abb. 10.4(b), S. 103). Wie vorhergesagt ist ebenfalls  $\kappa_2$  schwieriger zu schätzen als  $\kappa_1$  (Abb. 10.4(a)).

### 6.4.6 Maximum Likelihood Schätzung

Wie bei den beiden anderen Kalibriermethoden auch, ist die Minimierung des Reprojektionsfehlers aller Punkte in allen Bildern mit einem Optimierungsverfahren der letzte Schritt.

$$\operatorname{argmin} \sum_{i=1}^n \sum_{j=1}^m \|\mathbf{x}_{i,j} - \mathbf{F}(\mathbf{X}_i, \mathbf{K}, \mathbf{R}_j, \mathbf{t}_j, \kappa_1, \kappa_2, \rho_1, \rho_2)\|_2 \quad (6.30)$$

Die Funktion  $\mathbf{F}$  faßt Projektion und Linsenverzerrung zusammen, Index  $i$  läuft über alle 3D-Punkte,  $j$  über die Aufnahmen.

## 6.5 Stereokalibrierung

Das globale Koordinatensystem liegt für jedes aufgenommene Bild in einer Ecke des Kalibrierungsmusters (Abb. 6.5 links, Gleichung (6.19)). Für die Stereokamera ist dies ungünstig, da zu ihrer Beschreibung nur die Transformation zwischen den beiden Kameras interessant ist. Sie läßt sich bestimmen, indem man die globalen Koordinaten für jedes Stereobildpaar so transformiert, daß das globale mit dem lokalen Koordinatensystem einer der beiden Kameras zusammenfällt. Dann entsprechen die externen Parameter der anderen Kamera genau der Transformation zwischen den Kameras.

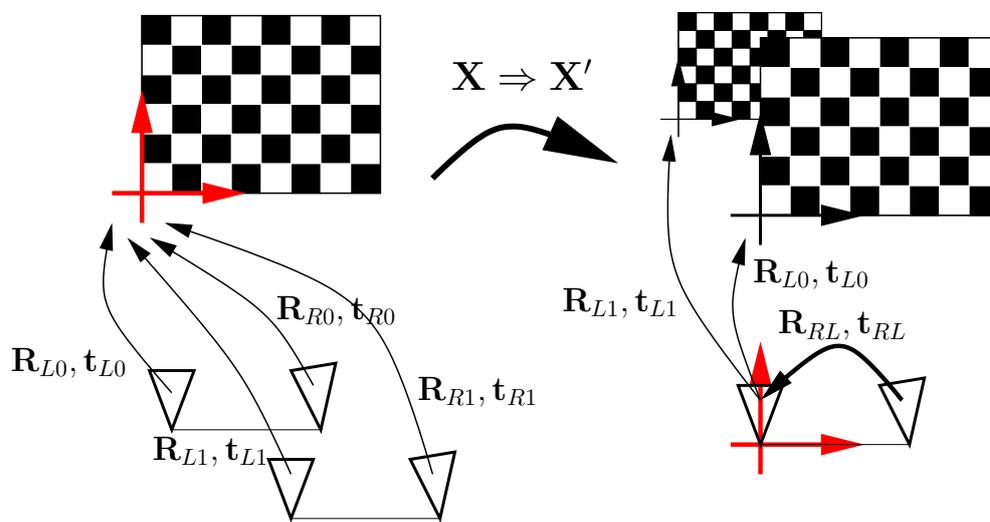


Abbildung 6.5: Übergang des globalen Koordinatensystems vom Kalibrierungsmuster zur linken Kamera.

Anschaulich entspricht dies einem Bezugssystemwechsel, es werden nicht verschiedene Kamerapositionen zur Aufnahme eines festen Musters eingenommen, sondern man nimmt das Muster an mehreren Positionen im Raum mit einer festen Kamera auf.

Mit der Wahl der linken Kamera als Bezugssystem und der daraus folgenden Transformation der globalen Koordinaten  $\mathbf{X}' = \mathbf{R}_L \mathbf{X} + \mathbf{t}_L$  ergibt sich

$$\begin{aligned}
 \mathbf{x}_L &= \mathbf{K}_L(\mathbf{R}_L \mathbf{X} + \mathbf{t}_L) & \mathbf{x}_R &= \mathbf{K}_R(\mathbf{R}_R \mathbf{X} + \mathbf{t}_R) & (6.31) \\
 &= \mathbf{K}_L \mathbf{X}' & &= \mathbf{K}_R(\mathbf{R}_R \mathbf{R}_L^{-1}(\mathbf{X}' - \mathbf{t}_L) + \mathbf{t}_R) \\
 & & &= \mathbf{K}_R(\mathbf{R}_R \mathbf{R}_L^{-1} \mathbf{X}' + \mathbf{t}_R - \mathbf{R}_R \mathbf{R}_L^{-1} \mathbf{t}_L).
 \end{aligned}$$

Daraus lassen sich die Rotation und Translation zwischen den Kameras ablesen:

$$\mathbf{R}_{RL} = \mathbf{R}_R \mathbf{R}_L^{-1} \quad \text{und} \quad \mathbf{t}_{RL} = \mathbf{t}_R - \mathbf{R}_R \mathbf{R}_L^{-1} \mathbf{t}_L. \quad (6.32)$$

Da es sich um gemessene Größen handelt, werden sie sich von Bildpaar zu Bildpaar leicht unterscheiden, sie sollten daher aus allen Paaren gemittelt werden. Die Translationsvektoren werden einfach komponentenweise gemittelt, für die Rotationsmatrizen dagegen geht dies nicht. Eine Möglichkeit, Rotationen zu mitteln, ist sie durch die minimale Anzahl von Parametern darzustellen (in diesem Fall drei), die Parameter zu mitteln und das Ergebnis wieder in Matrixform zu bringen. Geeignete Parametrisierungen sind Eulerwinkel oder die Rodrigues Formel [15].

Es empfiehlt sich, an die Mittelung anschließend eine nichtlineare Optimierung über alle Parameter durchzuführen, zum Beispiel wieder mit dem Levenberg-Marquardt Algorithmus. Abbildung 6.6 zeigt den kompletten Ablauf der Stereokalibrierung:

- Aufnahme von  $n$  Stereobildern.
- Getrennte Kalibrierung der linken und rechten Kamera mit dem Algorithmus aus Abschnitt 6.4.
- Bestimmung der Transformation zwischen den Kameras aus den externen Parametern von je zwei zusammengehörenden Bildern.
- Nichtlineare Optimierung über alle Parameter.
- Ausgabe der Linsenverzerrung und Kameramatrizen  $\mathbf{P}_L$  und  $\mathbf{P}_R$  in eine Datei.

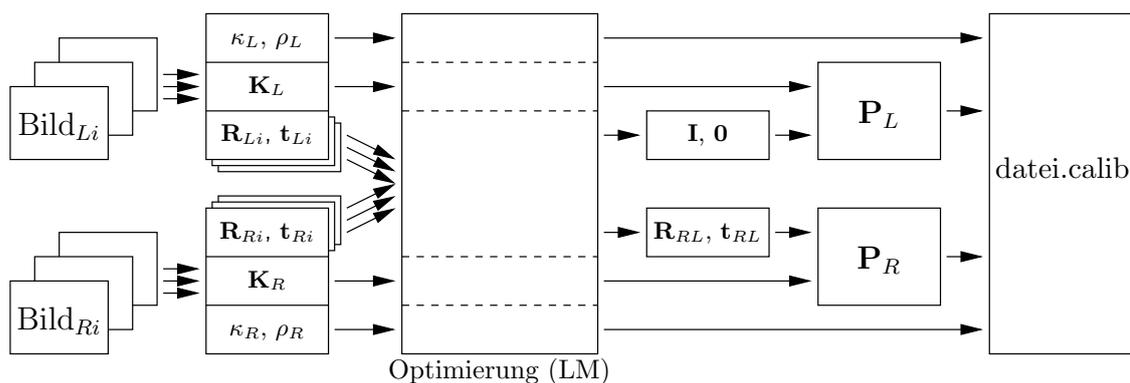
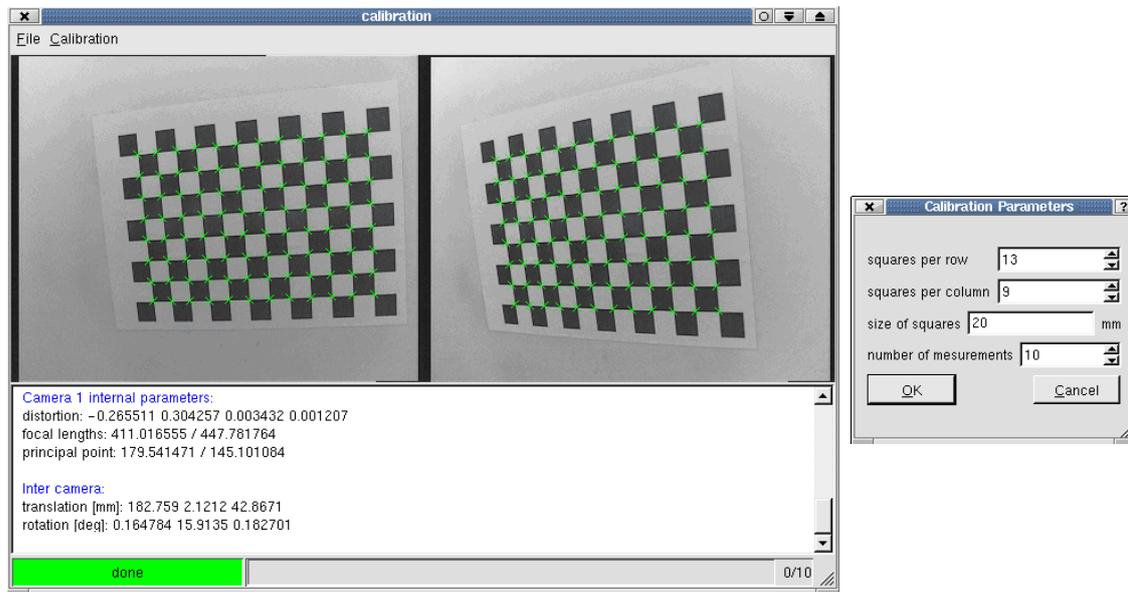


Abbildung 6.6: Schematischer Ablauf der Stereokalibrierung

Abbildung 6.7 zeigt die Oberfläche des Kalibrierprogrammes. Nach Einstellen der Parameter Anzahl und Größe der Quadrate sowie Anzahl der aufzunehmenden Bilder, werden kontinuierlich Bilder aufgenommen. Gefundene Kreuzungspunkte sind in den Bildern eingezeichnet, grün wenn alle gefunden wurden, rot wenn nur einige gefunden wurden. Sind beide Bilder in Ordnung, werden die Koordinaten abgespeichert, frühestens jedoch zwei Sekunden nach der letzten Speicherung, um zu ähnliche Aufnahmen zu vermeiden.

In Abbildung 6.8 sind die Ergebnisse der einzelnen Kalibrierungen, der Fehler pro Punkt vor der Optimierung und nach zehn Schritten des Levenberg-Marquardt Algorithmus, und das Ergebnis der Optimierung zu sehen. Die virtuellen Kamerazentren des Lochkameramodells sind etwa die 20 cm in  $x$ -Richtung auseinander und die Kameras ungefähr  $10^\circ$  aufeinander zu gedreht.



(a) Hauptfenster.

(b) Parameterdialog.

Abbildung 6.7: Oberfläche des Kalibrierprogrammes.

```

Calibration per camera...
  Camera 0 internal parameters:
distortion: -0.310619 -1.234043 -0.000429 -0.002133
focal lengths: 1065.933350 / 1162.243530
principal point: 163.028137 / 128.198883
  Camera 1 internal parameters:
distortion: -0.345074 1.444681 -0.000515 -0.001279
focal lengths: 1047.476685 / 1143.830322
principal point: 180.183411 / 141.366013

Refining inter-camera calibration...
before optimization: residual 0.127441/point
after 10 optimization steps: residual 0.109508/point

  Camera 0 internal parameters:
distortion: -0.327542 -1.046172 0.000108 -0.002132
focal lengths: 1060.642009 / 1156.052861
principal point: 170.749481 / 127.072363
  Camera 1 internal parameters:
distortion: -0.363826 3.818180 -0.001633 -0.001175
focal lengths: 1050.790804 / 1146.090845
principal point: 171.368672 / 139.116898

Inter-Camera:
translation [mm]: 196.064 5.94942 1.89238
rotation [deg]: -0.0591253 9.96944 0.0700541
    
```

Abbildung 6.8: Ergebnis einer Stereokalibrierung.

# 7 Bildvorverarbeitung

Nach der Aufnahme eines Stereobildpaares werden korrespondierende Punkte gesucht, d.h. Punkte bei denen im linken und rechten Bild der gleiche Weltpunkt abgebildet wurde. Müßte nun für jeden Punkt im linken Bild das komplette rechte Bild durchsucht werden, bedeutete das einen sehr großen Rechenaufwand. Glücklicherweise ist dem nicht so, in Abschnitt 7.1 wird gezeigt, daß das Absuchen einer Linie genügt. Durch eine Transformation der Bilder kann zudem erreicht werden, daß diese Linien mit den Zeilen der transformierten Bilder zusammenfallen und nur in horizontaler Richtung gesucht werden muß (Abschnitt 7.2). Eine einmalige Verzerrung der Bilder vor der Suche ist deutlich schneller, als während der Suche interpolieren zu müssen.

## 7.1 Epipolargeometrie und Fundamentalmatrix

Abbildung 7.1 zeigt zwei Kameras mit ihren Zentren  $\mathbf{C}$  und  $\mathbf{C}'$ , die den selben Punkt  $\mathbf{X}$  abbilden. Die beiden Kamerazentren und der Punkt spannen die Ebene  $\pi_e$  auf. Sie schneidet die Bildebenen in jeweils einer Linie, den Epipolarlinien  $\mathbf{l}$  und  $\mathbf{l}'$ . Alle Epipolarebenen bilden ein Bündel, sie haben die Verbindungsgerade der beiden Kamerazentren gemeinsam. Die Schnittpunkte dieser Geraden mit den Bildebenen werden Epipole genannt ( $\mathbf{e}$  und  $\mathbf{e}'$ ), sie sind die Bilder des jeweils anderen Kamerazentrums.

Wenn ein Punkt  $\mathbf{x}$  in einem Bild gegeben ist, wie lautet die Gleichung der Linie  $\mathbf{l}'$ , auf der der Punkt  $\mathbf{x}'$  im anderen liegen muß? Der Sehstrahl, auf dem  $\mathbf{x}$  liegt, kann wie folgt parametrisiert werden [73] (vgl. Gl. (6.8) auf Seite 48):

$$\mathbf{X}(\lambda) = \mathbf{P}^+ \mathbf{x} + \lambda \mathbf{C} \quad \text{mit} \quad \mathbf{P} \mathbf{P}^+ = \mathbf{I} \quad \text{und} \quad \mathbf{P} \mathbf{C} = \mathbf{0}, \quad (7.1)$$

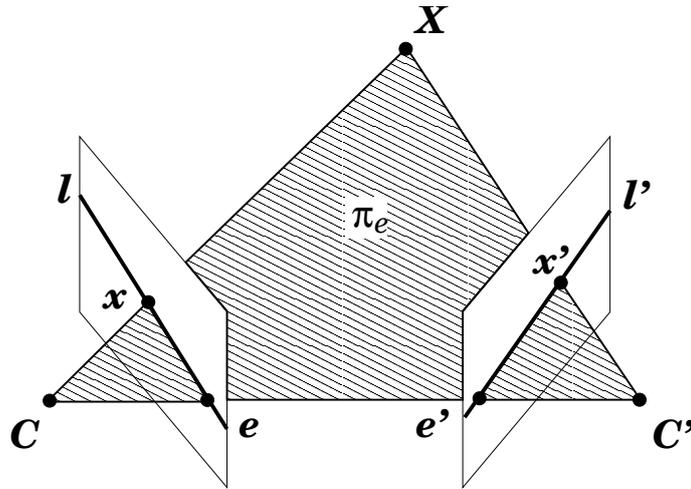


Abbildung 7.1: Epipolare Ebene und Epipolarlinien für einen Punkt.

$\mathbf{P}^+$  bezeichnet die Pseudoinverse von  $\mathbf{P}$ , das Zentrum  $\mathbf{C}$  ist der rechte Nullvektor von  $\mathbf{P}$ . Die Gleichung der Linie  $l'$  ergibt sich aus der Abbildung zweier Punkte der Linie  $l$  durch die rechte Kamera. Der erste Punkt ist das Zentrum der linken Kamera  $\mathbf{C}$ , der zweite der Punkt  $\mathbf{P}^+\mathbf{x}$ , der sich für  $\lambda = 0$  aus Gleichung (7.1) ergibt. In homogenen Koordinaten liefert das Kreuzprodukt zweier Punkte die Gleichung der Geraden, die sie verbindet ( $\mathbf{l} = \mathbf{x}_1 \times \mathbf{x}_2$ ,  $\Rightarrow \mathbf{x}_1^T \mathbf{l} = 0$  und  $\mathbf{x}_2^T \mathbf{l} = 0$ ), in diesem Fall also

$$l' = (\mathbf{P}'\mathbf{C}) \times (\mathbf{P}'\mathbf{P}^+\mathbf{x}). \quad (7.2)$$

Das Kreuzprodukt läßt sich in Matrixform schreiben

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{bmatrix} \mathbf{b} = [\mathbf{a}]_{\times} \mathbf{b}, \quad (7.3)$$

damit kann Gleichung (7.2) unter Berücksichtigung von  $\mathbf{P}'\mathbf{C} = \mathbf{e}'$  umgeformt werden zu

$$l' = [\mathbf{e}']_{\times} \mathbf{P}'\mathbf{P}^+\mathbf{x}. \quad (7.4)$$

Die Tatsache, daß der Punkt im zweiten Bild auf der Epipolarlinie liegen muß, läßt sich ausdrücken durch

$$\mathbf{x}'^T l' = \mathbf{x}'^T [\mathbf{e}']_{\times} \mathbf{P}'\mathbf{P}^+\mathbf{x} = 0. \quad (7.5)$$

Das Produkt  $[\mathbf{e}']_{\times} \mathbf{P}'\mathbf{P}^+$  wird Fundamentalmatrix  $\mathbf{F}$  genannt. Sie ist eine  $3 \times 3$  Matrix von höchstens Rang 2, da die Kreuzproduktmatrix aus Gleichung (7.3) ebenfalls

den Rang 2 besitzt. Die fundamentale Beziehung zwischen zwei Bildern einer Szene nimmt die folgende einfache Form an:

$$\mathbf{x}'^T \mathbf{F} \mathbf{x} = 0. \quad (7.6)$$

Aus Gleichung (7.6) ist ersichtlich, daß die Projektionsmatrizen nicht mehr vorkommen, sondern nur noch die Bildkoordinaten. Die Fundamentalmatrix kann allein aus korrespondierenden Punkten berechnet werden. Da dies im kalibrierten Fall nicht benötigt wird, soll hier nicht näher darauf eingegangen werden, ausführliche Analysen der Epipolargeometrie und ihrer Bestimmung finden sich zum Beispiel in [23] und [15].

Die folgende grundlegende Eigenschaft der Fundamentalmatrix wird im nächsten Abschnitt gebraucht. Alle Epipolarlinien eines Bildes schneiden sich im Epipol, setzt man  $\mathbf{x} = \mathbf{e}$ , gilt Gleichung (7.6) für alle  $\mathbf{x}'$ . Die Epipole sind linker und rechter Nullvektor der Fundamentalmatrix.

$$\mathbf{F} \mathbf{e} = \mathbf{0} \quad \text{und} \quad \mathbf{e}'^T \mathbf{F} = \mathbf{0} \Rightarrow \mathbf{F}^T \mathbf{e}' = \mathbf{0} \quad (7.7)$$

## 7.2 Rektifikation

Wie bereits zu Beginn dieses Kapitels erwähnt, sollen die Bilder so ausgerichtet („rektifiziert“) werden, daß korrespondierende Epipolarlinien horizontal und auf gleicher Höhe verlaufen, um den nachfolgenden Stereoalgorithmus effizient implementieren zu können.

### 7.2.1 Geometrisches Verfahren

Reprojiziert man die Bilder in eine neue, gemeinsame Bildebene (Abbildung 7.2), die parallel zur Verbindungsgeraden der beiden Kamerazentren verläuft und diese nicht enthält, dann wird diese Ebene von allen Epipolarebenen des Büschels in parallelen Linien geschnitten. Die Kamerazentren müssen dabei unverändert bleiben. Durch geeignete Wahl der neuen Bildkoordinaten wird schließlich die Forderung nach identischer  $y$ -Koordinate erfüllt.

### 7.2.2 Algebraisches Verfahren

Die hier beschriebene Methode zur Rektifikation basiert allerdings nicht auf den Projektionsmatrizen und damit der geometrischen Betrachtung (wie z.B. [19]) sondern sie wird algebraisch aus der Fundamentalmatrix hergeleitet. Der Vorteil dieser

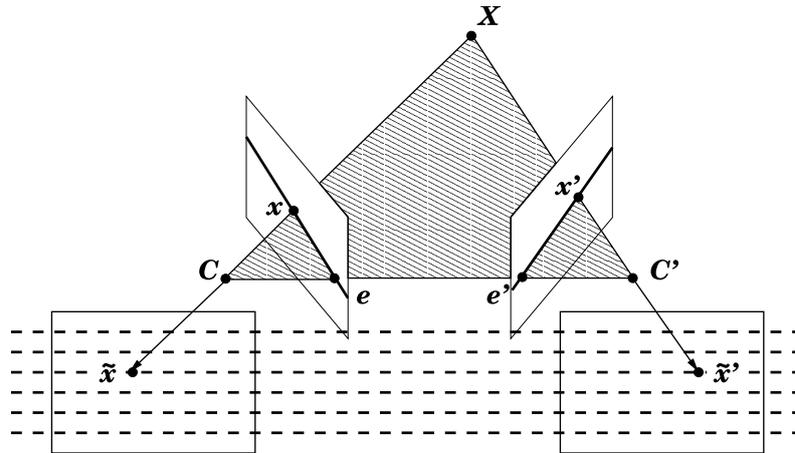


Abbildung 7.2: Rektifikation durch Reprojektion in eine gemeinsame Bildebene.

Herangehensweise ist, daß sie sich gleichermaßen für kalibrierte wie unkalibrierte Bildpaare anwenden läßt.

Zunächst wird also diejenige Fundamentalmatrix benötigt, die aus den beiden Projektionsmatrizen resultiert. Eine Möglichkeit ist, sie aus Gleichung (7.5) zu bestimmen. Eine andere, die ohne die Berechnung der Pseudoinversen auskommt, wird in [24] hergeleitet. Angenommen, die beiden Kameramatrizen liegen in der Form  $\mathbf{P} = [\mathbf{M} | -\mathbf{M}\mathbf{t}]$  und  $\mathbf{P}' = [\mathbf{M}' | -\mathbf{M}'\mathbf{t}']$  vor (vgl. Gl. (6.8) mit  $\mathbf{M} = \mathbf{K}\mathbf{R}$  und  $\mathbf{t} = \mathbf{C}$ ), dann ergibt sich die Fundamentalmatrix

$$\mathbf{F} = [\mathbf{M}'(\mathbf{t}' - \mathbf{t})]_{\times} (\mathbf{K}'\mathbf{K}^{-1}). \quad (7.8)$$

Für Bilder, die die gewünschte Bedingung  $y = y'$  erfüllen, hat die Fundamentalmatrix eine besondere Form

$$\begin{pmatrix} x' & y' & 1 \end{pmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0. \quad (7.9)$$

Da es sich um homogene Größen handelt, ist  $\mathbf{F}$  natürlich nur bis auf einen Skalierungsfaktor ungleich Null bestimmt, die notwendige Bedingung ist  $f_{23} = -f_{32} \neq 0$  und die restlichen Elemente gleich Null.

Das Ziel ist, die Bilder so zu transformieren, daß die Fundamentalmatrix die eben beschriebene Normalform annimmt. Auch für die transformierten Bilder  $\tilde{\mathbf{x}} = \mathbf{H}\mathbf{x}$  und  $\tilde{\mathbf{x}}' = \mathbf{H}'\mathbf{x}'$  muß Gleichung (7.6) erfüllt sein

$$\underbrace{\mathbf{x}'^T \mathbf{H}'^T}_{\tilde{\mathbf{x}}'^T} \underbrace{\mathbf{H}'^{-T} \mathbf{F} \mathbf{H}^{-1}}_{\tilde{\mathbf{F}}} \underbrace{\mathbf{H} \mathbf{x}}_{\tilde{\mathbf{x}}} = 0, \quad (7.10)$$

wobei  $\tilde{\mathbf{F}}$  von der Form sein soll wie in Gleichung (7.9). Die Transformationen der Bilder,  $\mathbf{H}$  und  $\mathbf{H}'$ , müssen invertierbar sein, also den Rang 3 besitzen.

### 7.2.3 Algorithmus

Durch schrittweise Akkumulation geeigneter Transformationen werden  $\mathbf{H}$  und  $\mathbf{H}'$  so bestimmt, daß sie einerseits die Bilder rektifizieren, also auf  $\tilde{\mathbf{F}}$  führen, und andererseits die Verzerrung möglichst gering halten, damit die nachfolgende Korrelation nicht unnötig erschwert wird. Es sei noch angemerkt, daß eine (invertierbare) projektive Verzerrung eines Bildes das Kamerazentrum unverändert läßt, da dann aus  $\mathbf{PC} = \mathbf{0}$  auch  $(\mathbf{HP})\mathbf{C} = \mathbf{0}$  folgt. Die algebraische Rektifikation funktioniert prinzipiell wie die geometrische, ohne allerdings einige geometrische Größen explizit zu benötigen.

Die erste Spalte der Fundamentalmatrix läßt sich auf Null setzen, indem sie von rechts mit einer Matrix multipliziert wird, deren erste Spalte aus ihrem rechten Nullvektor (dem Epipol im linken Bild) besteht

$$\mathbf{F}\mathbf{e} = \mathbf{0} \quad \Rightarrow \quad \mathbf{F} \begin{bmatrix} e_1 & 0 & 0 \\ e_2 & 1 & 0 \\ e_3 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 0 & f_{12} & f_{13} \\ 0 & f_{22} & f_{23} \\ 0 & f_{32} & f_{33} \end{bmatrix}. \quad (7.11)$$

Analog dazu wird von links die erste Zeile auf Null gesetzt

$$\mathbf{v}^T \begin{bmatrix} 0 & f_{12} & f_{13} \\ 0 & f_{22} & f_{23} \\ 0 & f_{32} & f_{33} \end{bmatrix} = \mathbf{0}, \quad \|\mathbf{v}\| = 1 \quad \Rightarrow$$

$$\begin{bmatrix} v_1 & v_2 & v_3 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 & f_{12} & f_{13} \\ 0 & f_{22} & f_{23} \\ 0 & f_{32} & f_{33} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & f_{22} & f_{23} \\ 0 & f_{32} & f_{33} \end{bmatrix}. \quad (7.12)$$

Übrig bleibt eine  $2 \times 2$ -Matrix, die mit Hilfe von vier linearen Gleichungen für die vier gesuchten Elemente auf die gewünschte Form gebracht wird

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & f_{22} & f_{23} \\ 0 & f_{32} & f_{33} \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & a & b \\ 0 & c & d \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & -1 & 0 \end{bmatrix} \quad \Rightarrow \quad (7.13)$$

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & a & b \\ 0 & c & d \end{bmatrix} = \frac{1}{f_{22}f_{33} - f_{23}f_{32}} \begin{bmatrix} 1 & 0 & 0 \\ 0 & f_{23} & f_{33} \\ 0 & -f_{22} & -f_{32} \end{bmatrix}.$$

Von nun an sind die transformierten Bilder rektifiziert. Alle weiteren Transformationen dürfen diese Beziehung nicht mehr verändern, es sind also nur noch Transformationen erlaubt, die sich auf die  $x$ -Koordinate auswirken oder auf beide  $y$ -Koordinaten gleich.

Abbildung 7.3 zeigt die transformierten Eckpunkte eines Bildes in den neuen Koordinaten. Die beiden Vektoren  $\mathbf{v}_1$  und  $\mathbf{v}_2$ , verbinden gegenüberliegende Seitenmittelpunkte.

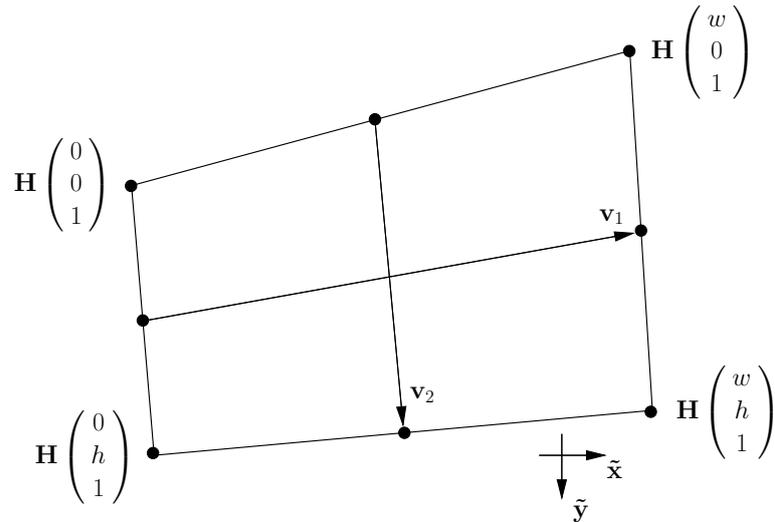


Abbildung 7.3: Transformiertes Bild vor der Korrektur der Scherung.

Die nächste Transformation, die für jedes Bild einzeln durchgeführt wird, soll dafür sorgen, daß die Verzerrungen gegenüber den ursprünglichen Bildern möglichst klein gehalten werden. Dazu werden zwei Parameter  $s_1$  und  $s_2$  so gewählt, daß  $\mathbf{v}_1$  und  $\mathbf{v}_2$  orthogonal sind und ihr Verhältnis dem Verhältnis der Länge zur Breite der Originalbilder entspricht. Die Transformation entspricht einer Skalierung in  $x$ -Richtung kombiniert mit einer Scherung

$$\mathbf{S} = \begin{bmatrix} s_1 & s_2 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}, \text{ so daß } \mathbf{v}_1 \mathbf{v}_2 = 0 \text{ und } \frac{\|\mathbf{v}_1\|}{\|\mathbf{v}_2\|} = \frac{w}{h}. \quad (7.14)$$

Die beiden Bedingungen lassen sich sehr einfach als Optimierungsproblem formulieren. Ausgehend vom Startwert  $s_1 = 1$ ,  $s_2 = 0$ , konvergiert ein Levenberg-Marquardt Verfahren nach wenigen Iterationen.

Zuletzt wird auf beide Bilder eine identische isotrope Skalierung angewandt, die dafür sorgt, daß der Flächeninhalt der transformierten Bilder mit dem der Originalbilder übereinstimmt. Dadurch wird ein grobes Über- oder Unterabtasten vermieden.

## 7.3 Implementierung der Entzerrung

Die Teile der rektifizierten Bilder, auf deren Höhe sich nichts vom anderen Bild befindet, sind für die Korrelation uninteressant. Zuerst wird also durch  $\tilde{y}_{\min} = \max(\min(\tilde{y}), \min(\tilde{y}'))$  und  $\tilde{y}_{\max} = \min(\max(\tilde{y}), \max(\tilde{y}'))$  die vertikale Ausdehnung der rektifizierten Bilder bestimmt, dann die horizontale. Es ist zweckmäßig, sie auch in horizontaler Richtung gleich groß zu machen, auch wenn es dabei am Rand eines Bildes einen Bereich geben kann, der über die gesamte Höhe leer bleibt. Zuletzt werden die Bilder so verschoben, daß der linke obere Punkt im Ursprung zu liegen kommt, damit die Bildkoordinaten von 0 bis  $w = \tilde{x}_{\max} - \tilde{x}_{\min}$  beziehungsweise  $h = \tilde{y}_{\max} - \tilde{y}_{\min}$  laufen.

Während der Aufnahme mit der Stereokamera bleiben die Kameras fest zueinander und ihre internen Parameter unverändert. Faßt man die Rektifikation und den Ausgleich der Linsenverzerrung zusammen und berechnet daraus eine Lookup-Tabelle, können die Videobilder danach sehr schnell entzerrt werden.

In einer Schleife wird für jedes Pixel des Ergebnisbildes ein Eintrag in der Lookup-Tabelle generiert, der vier Koeffizienten zur bilinearen Interpolation und den Offset eines Pixels relativ zum Anfang des Bildpuffers enthält (Algorithmus 7.1). Die Koeffizienten sind aus dem Wertebereich  $[0, 1)$ , sie werden als 8 Bit Festkommazahlen abgelegt. Der Offset ist der Index des linken oberen der vier bei der Interpolation beteiligten Pixel. Weil der Quellpuffer 24 Bit RGB-Daten enthält, muß der Offset noch mit drei multipliziert werden. Pixel, deren Quellpixel außerhalb der Originalbildes lägen, bekommen den Offset 0. Durch Überschreiben des ersten Pixels im Quellpuffer kann die Farbe dieser Pixel im Ergebnisbild bestimmt werden. Wählt man unterschiedliche Farben für das linke und rechte Bild, läßt sich ein „Einrasten“ des Stereoalgorithmus an den Bildrändern verhindern (s. 11.1, S. 105). Natürlich muß die Lookup-Tabelle für jedes Bild getrennt berechnet und gespeichert werden.

Während der laufenden Rekonstruktion können die Bilder dank der vorberechneten Tabellen sehr effizient entzerrt werden. Abbildung 7.4 skizziert die bilineare Interpolation für ein Pixel mit Hilfe von MMX-Befehlen[29]. Zuerst wird der 64 Bit große Eintrag aus der Tabelle gelesen und in 32 Bit Offset und die vier 8 Bit Faktoren entpackt. Aus Offset und dem Anfang des Bildpuffers ergibt sich die Adresse der oberen beiden Pixel, die mit einem `movq`-Befehl gelesen werden. Dieser Zugriff ist nicht auf eine Adresse ausgerichtet, die ein Vielfaches von acht ist, was seine Ausführungsgeschwindigkeit verringern kann. Ein einzelner Zugriff ist jedoch deutlich einfacher und schneller, als die RGB-Daten aus mehreren ausgerichteten Zugriffen zusammensetzen. Das zweite Pixelpaar kommt von einer  $3 \times w_O$  Bytes entfernten Adresse und wird ebenfalls in einem Befehl gelesen ( $w_O$  ist die Breite des Original-

---

**Algorithmus 7.1** Generierung einer Lookup-Tabelle zur Rektifikation

---

```
typedef struct LUTentry {
    unsigned int offset;
    unsigned char f[4];
};
LUTentry LookUpTable[w*h];
LUTentry* LUT = LookUpTable;
for  $\tilde{y} = 0$  to  $h - 1$  do
    for  $\tilde{x} = 0$  to  $w - 1$  do
         $(x \ y \ \tilde{w})^T = \mathbf{H}^{-1}(\tilde{x} \ \tilde{y} \ 1)^T$ 
         $x_i = x/\tilde{w}, y_i = y/\tilde{w}$ 
         $r = (x_i - p_x)^2 + (y_i - p_y)^2$ 
         $\bar{x} = x_i + (x_i - p_x) \left[ \kappa_1 r^2 + \kappa_2 r^4 + 2\rho_1 y_i + \rho_2 \left( r^2/x_i + 2x_i \right) \right]$ 
         $\bar{y} = y_i + (y_i - p_y) \left[ \kappa_1 r^2 + \kappa_2 r^4 + 2\rho_2 x_i + \rho_1 \left( r^2/y_i + 2y_i \right) \right]$ 
        /*  $w_O, h_O =$  Breite und Höhe der Originalbilder */
        if  $\bar{x} \in [0, w_O - 2] \wedge \bar{y} \in [0, h_O - 2]$  then
            LUT->offset =  $3 \cdot (\lfloor \bar{x} \rfloor + w_O \cdot \lfloor \bar{y} \rfloor)$ 
             $u = \bar{x} - \lfloor \bar{x} \rfloor, v = \bar{y} - \lfloor \bar{y} \rfloor$ 
            LUT->f[0] = (int)(255.0*(1.0-u)*(1.0-v));
            LUT->f[1] = (int)(255.0*    u *(1.0-v));
            LUT->f[2] = (int)(255.0*(1.0-u)*    v );
            LUT->f[3] = (int)(255.0*    u *    v );
        else /*  $(\bar{x} \ \bar{y})^T$  außerhalb des Originalbildes */
            LUT->offset = 0;
            LUT->f[0] = 255;
            LUT->f[1] = LUT->f[2] = LUT->f[3] = 0;
        end if
        ++LUT;
    end for
end for
```

---



bildes).

Die RGB-Werte werden dann in je ein MMX-Register pro Pixel von 8 auf 16 Bit ausgepackt, das zweite Quellregister des Packbefehls enthält Null. Die ab dem Pentium III verfügbare `pshufw`-Anweisung erleichtert die Aufteilung der vier Faktoren auf separate Register. Nachdem RGB-Wert und der zugehörige Faktor in eigenen Registern stehen, müssen sie nur noch miteinander multipliziert werden. Das dreimal 16 Bit große Ergebnis der Multiplikationen akkumuliert ein Summenregister, die jeweils acht Nachkommastellen werden durch eine Schiebeoperation und Maskierung entfernt. Der endgültige RGB-Wert des interpolierten Pixels wird schließlich gepackt und abgespeichert.

# 8 Disparitätsberechnung

Die Bestimmung der Disparitäten der einzelnen Bildpunkte ist der zentrale Teil der 3D-Rekonstruktion. Für sie wird der größte Teil der Rechenzeit benötigt, weshalb sich die in diesem Kapitel vorgestellte Implementierung auf die Optimierung der Ausführungsgeschwindigkeit konzentriert. Der Algorithmus soll jedoch nicht nur für die schnelle 3D-Rekonstruktion eingesetzt werden, sondern auch für Anwendungen bei denen wesentlich größere Bilder verarbeitet werden, muß also flexibel bleiben.

Optimierungen, die eine Geschwindigkeitssteigerung erzielen, indem sie etwa die Bildgröße oder den Disparitätssuchbereich von vorneherein festlegen, wie es zum Beispiel die Systeme aus Abschnitt 3.2 machen, sollen nicht zur Anwendung kommen, ebensowenig wie Verfahren, die zeitintensive globale Optimierungen verwenden.

Die vorgestellten Optimierungen betreffen zum einen die Algorithmen, es wird gezeigt wie eine effiziente Berechnung Rechenoperationen sparen kann [47, 50]. Zum anderen wird auf die Implementierung auf Intel kompatiblen 32 Bit Prozessoren eingegangen, insbesondere wird die Eignung der MMX-Befehle [29, 30] für die Beschleunigung der Algorithmen untersucht.

## 8.1 Disparität

Betrachtet man ein rektifiziertes Stereobildpaar, unterscheiden sich die Koordinaten korrespondierender Bildpunkte nur in der  $x$ -Komponente, dieser Unterschied wird Disparität genannt.

Abbildung 8.1 zeigt zwei Kameras in Standardgeometrie, d.h. alle Achsen sind parallel und die Brennweiten und -ebenen identisch. Der Abstand zwischen den Kameras sei die Stereobasis  $b$ . In Standardgeometrie sind die Bilder automatisch rektifiziert.

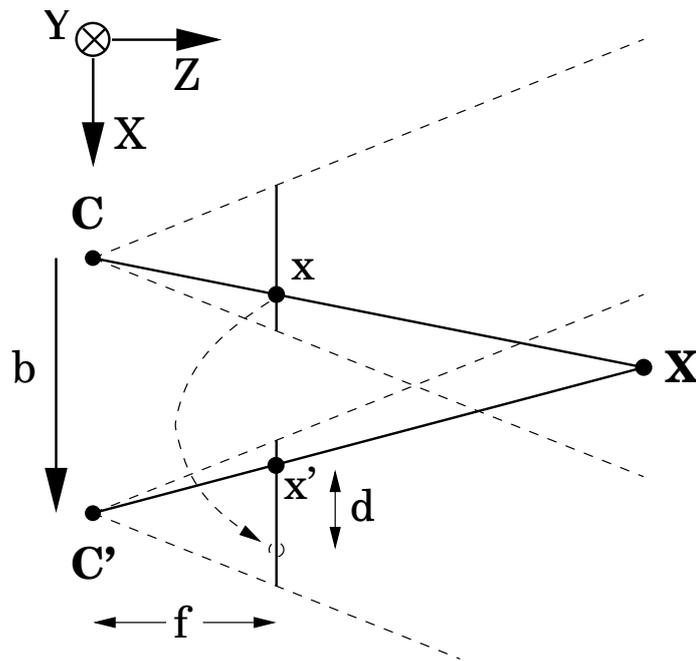


Abbildung 8.1: 3D-Punkt und Disparität zwischen seinen Bildpunkten.

Aus Gleichung (6.1) (S. 46) und der Tatsache, daß sich die Koordinatensysteme der beiden Kameras nur um die Translation  $(b \ 0 \ 0)^T$  unterscheiden, folgt für die horizontalen Bildkoordinaten eines Punktes  $\mathbf{X} = (X, Y, Z)^T$

$$x = f \frac{X}{Z}, \quad x' = f \frac{X - b}{Z} = x - f \frac{b}{Z} = x - d. \quad (8.1)$$

Bei Standardgeometrie ist die Disparität  $d = bf/z$  umgekehrt proportional zur Tiefe, für Punkte im Unendlichen wird sie Null.

## 8.2 Disparitätsvolumen

Die Aufgabe des Stereoalgorithmus ist, die Disparität für jedes Pixel in einem Bild zu bestimmen, also den Punkt im anderen Bild, für den die Wahrscheinlichkeit am größten ist, daß er den selben Weltpunkt abbildet. Weil die Kameras der Stereokamera aus Kapitel 4 nebeneinander angeordnet sind, wird im folgenden anschaulich von linkem und rechtem Bild gesprochen, das Gesagte gilt natürlich für alle rektifizierten Bildpaare unabhängig von der ursprünglichen Aufnahmeconfiguration. Durch die Abmessungen der Bilder  $w \times h$  und dem Suchbereich von  $d_{\min}$  bis  $d_{\max}$  ergibt

sich ein Volumen (Abb. 8.2), in dem an der Stelle  $(x, y, d)$  ein Maß für die Ähnlichkeit zwischen dem Fenster um das Pixel  $(x, y)$  im linken und dem Fenster um  $(x + d, y)$  im rechten Bild abgespeichert ist [65, 32]. In Anlehnung an das Wort Pixel werden die Einträge im Disparitätsvolumen im folgenden als „Voxel“ bezeichnet.

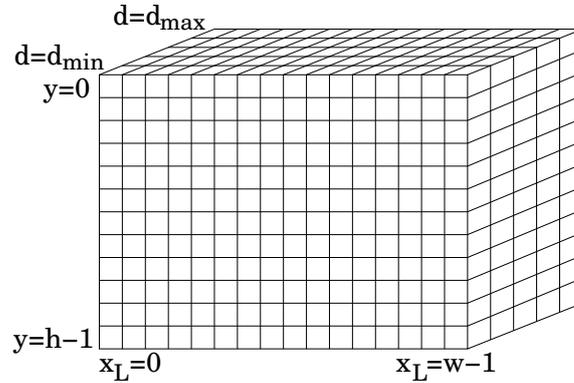


Abbildung 8.2: Disparitätsvolumen.

## 8.3 Ähnlichkeitsmaß

Die Qualität des Ergebnisses wird davon beeinflusst, wie gut das Ähnlichkeitsmaß geeignet ist, das richtige Pixel von den falschen zu unterscheiden, wie eindeutig das gesuchte Extremum ist und ob es an der richtigen Stelle auftritt. Weil das Ähnlichkeitsmaß für jede Position im Volumen berechnet werden muß, hat es außerdem einen großen Einfluß auf die Geschwindigkeit.

Von uns gewählt wurde die Summe der absoluten Differenzen (SAD), aufsummiert über ein rechteckiges Fenster der Größe  $win_x \times win_y$

$$\begin{aligned}
 \text{SAD}(x, y, d) = & \sum_{i=-\frac{1}{2}(win_x-1)}^{\frac{1}{2}(win_x-1)} \sum_{j=-\frac{1}{2}(win_y-1)}^{\frac{1}{2}(win_y-1)} \\
 & [ |R_L(x+i, y+j) - R_R(x+i+d, y+j)| \\
 & + |G_L(x+i, y+j) - G_R(x+i+d, y+j)| \\
 & + |B_L(x+i, y+j) - B_R(x+i+d, y+j)| ].
 \end{aligned} \tag{8.2}$$

Die folgende Aufzählung stellt ihre Vor- und Nachteile gegenüber anderen möglichen Korrelationsmaßen vor:

- Die Summe der quadrierten Differenzen SSD verhält sich sehr ähnlich, die SAD ist jedoch robuster gegenüber einzelnen falschen Pixeln.
- Die normierte Kreuzkorrelation ist etwas robuster bei einem konstanten Helligkeitsunterschied der beiden Bilder. Sie benötigt aber zusätzlichen Rechenaufwand und ist nicht besser als die SAD oder SSD, wenn die Bilder den selben Weißabgleich und die gleiche Helligkeit haben, was sich mit dem verwendeten Kameras erreichen läßt (s. 4.1).
- Bei der Census Transformation, die zum Beispiel im System aus Abschnitt 3.1.2 und 3.2.3 verwendet wird [71], werden Bitvektoren gebildet, die für jedes Pixel angeben, ob sein Grauwert größer oder kleiner ist als der des mittleren Pixels. Diese Vektoren werden dann verglichen. Sie eignet sich gut für eine Hardwarelösung, ist aber in Software nicht effizient zu berechnen.
- Variable Fenstergrößen versprechen exaktere Ergebnisse vor allem an Kanten von Objekten, da dort Pixel von Vorder- und Hintergrund im Fenster gemischt sind [36]. Für eine Echtzeitanwendung sind diese Verfahren jedoch deutlich zu langsam.

Soll die Korrelation mit Hilfe von MMX-Befehlen beschleunigt werden, ist es von großem Vorteil, 16 Bit pro Voxel im Disparitätsvolumen zu verwenden. Um Überläufe zu vermeiden, muß dann die SAD der einzelnen Pixel nach oben begrenzt werden, so daß die Fenstersumme sich mit 16 Bit darstellen läßt

$$\begin{aligned}
 \text{SAD}(x, y, d) = & \sum_{i=-\frac{1}{2}(win_x-1)}^{\frac{1}{2}(win_x-1)} \sum_{j=-\frac{1}{2}(win_y-1)}^{\frac{1}{2}(win_y-1)} \\
 & \min \left( \begin{aligned} & |R_L(x+i, y+j) - R_R(x+i+d, y+j)| \\ & + |G_L(x+i, y+j) - G_R(x+i+d, y+j)| \\ & + |B_L(x+i, y+j) - B_R(x+i+d, y+j)| \end{aligned} \right), \left\lfloor \frac{2^{16} - 1}{win_x win_y} \right\rfloor.
 \end{aligned} \tag{8.3}$$

Nebeneffekt ist, daß die SAD dadurch sogar etwas robuster wird [57]. Ein weiterer wichtiger Vorteil der Verwendung von 16 statt 32 Bit ist die Halbierung des Speicherverbrauchs und der benötigten Speicherbandbreite.

Seit dem Pentium III steht der `psadbw`-Befehl zur Verfügung, der die Berechnung für ein Pixel in einem Maschinenbefehl ermöglicht, die Begrenzung des Korrelationswertes erfolgt mit dem `pminsw`-Befehl.

Es ist offensichtlich, daß die Differenz zweier einzelner Pixel in  $win_x \times win_y$  Fenstern wiederholt als Teilsumme vorkommt. Durch geschicktes Aufsummieren kann erreicht werden, daß sie nur einmal berechnet werden muß (s. 8.5).

## 8.4 Speicherorganisation

Bei der mathematischen Formulierung des Korrelationsverfahrens ist es völlig unerheblich wie die Daten angeordnet sind, bei der realen Implementierung hat die Wahl des Speicherlayouts jedoch erheblichen Einfluß auf die Ausführungsgeschwindigkeit. Durch die effiziente Berechnung ist der begrenzende Faktor bei der Korrelation nicht mehr die Rechenzeit sondern die Speicherbandbreite, es ist also von Vorteil, so wenig Speicher wie möglich zu benutzen und die Häufigkeit der Zugriffe so gering wie möglich zu halten.

Abschnitt 8.4.1 und 8.4.3 zeigen, daß der Speicherverbrauch deutlich geringer als das komplette Volumen ausfallen kann, was die Korrelation sehr großer Bilder erst möglich macht. Das Disparitätsvolumen zweier Bilder einer 3 Megapixel Kamera mit einem Suchbereich von 500 Pixeln würde zum Beispiel etwa 3 Gigabyte benötigen (bei 16 Bit pro Voxel), mit dem verwendeten Speicherlayout kann dies auf handhabbare 45 Megabyte reduziert werden.

Die nächsten drei Abschnitte stellen je eine Möglichkeit vor, das Disparitätsvolumen im Speicher anzuordnen, und diskutieren ihre Vor- und Nachteile. Unterschieden werden sie nach der Reihenfolge, in der die Variablen im Speicher laufen, zuerst die schnellste und zuletzt die sich am langsamsten ändernde.

### 8.4.1 x, y, d

Legt man das Volumen so an, daß es zusammenhängende Schichten konstanter Disparität enthält, kann die pixelweise SAD sehr einfach berechnet werden, indem mit drei Pointern linear durch die Bilder und die Schicht im Volumen gelaufen wird. Sie läßt sich weiter beschleunigen, wenn die Anzahl der Speicherzugriffe verringert wird indem immer zwei aufeinanderfolgende Pixel in ein MMX-Register gelesen werden. Weil die Voxel in x-Richtung hintereinander liegen, kann die Summierung in diese Richtung (Abb. 8.4) nicht mit MMX beschleunigt werden, nur die in y-Richtung (Abb. 8.5). Die pixelweise SAD und die Summen können in drei getrennten Schleifen berechnet werden, die sehr einfach aufgebaut sind und vom Compiler gut optimiert werden können. Dem Vorteil der einfachen Implementierung steht allerdings der Nachteil gegenüber, daß dreimal (mit der Suche viermal) auf den Speicher zugegriffen werden muß.

Der größte Nachteil ist, daß erst mit der Minimumsuche (s. 8.6) begonnen werden kann, wenn alle Schichten abgearbeitet sind, es muß also das gesamte Volumen im Speicher gehalten werden. Die Speicherzugriffe auf aufeinanderfolgende Disparitäten eines Pixels erfolgen auf weit auseinanderliegende Adressen, was sich negativ auf ihre Geschwindigkeit auswirkt.

### 8.4.2 $x, d, y$

Es werden nur  $win_y+2$  Schichten des Disparitätsvolumens als Ringpuffer im Speicher gehalten (vgl. Abb. 8.5), die pixelweise SAD und die Summierung finden simultan statt (s. 8.5), so daß nur ein Durchlauf durch das Volumen erforderlich ist. Auch wenn  $x$  am schnellsten läuft, empfiehlt es sich, alle Voxel einer Disparität zusammen abzuarbeiten. Dabei muß das Pixel des linken Bildes nur einmal gelesen und in einem dedizierten Register gehalten werden, die Zahl der Zugriffe auf die Bilder sinkt von  $2(d_{\max}-d_{\min}+1)$  auf  $d_{\max}-d_{\min}+2$ . Die Suche nach dem Minimum kann ebenfalls in der innersten Schleife erfolgen und benötigt keinen separaten Durchlauf.

Wenn die innerste Schleife über  $d$  läuft, liegt es allerdings nahe, Voxel aufeinanderfolgender Disparität auch an aufeinanderfolgenden Adressen zu speichern, was im nächsten Abschnitt besprochen wird.

### 8.4.3 $d, x, y$

Eine Cacheline im Pentium III besteht aus 32 Bytes, nimmt also bei 32 Bit pro SAD-Wert 8, bei 16 Bit 16 Voxel auf. Läuft  $x$  am schnellsten, sind darin in  $x$  benachbarte Voxel enthalten (Abb. 8.3(a)), die erst gebraucht werden, wenn die Suche am aktuellen Pixel beendet ist. Je größer der Disparitätsbereich ist, desto geringer sind die Chancen daß die Werte noch im L1-Cache vorhanden sind.

Dreht man das Layout so, daß  $d$  am schnellsten läuft, sind immer genau diejenigen Werte im Cache, die für die Suche entlang der Disparitätsrichtung benötigt werden (Abb. 8.3(b)).

## 8.5 Aufsummieren der Fenster

Die Akkumulation der SAD über ein Korrelationsfenster kann sehr effizient mit einer gleitenden Summe gelöst werden [48, 16]. Die Berechnung teilt sich in zwei Schritte auf: Bildung der Summen einzelner Zeilen und deren Aufsummierung zu kompletten Fenstern. Abbildung 8.4 verdeutlicht die Summierung in  $x$ -Richtung für eine Fensterbreite von fünf Pixeln.

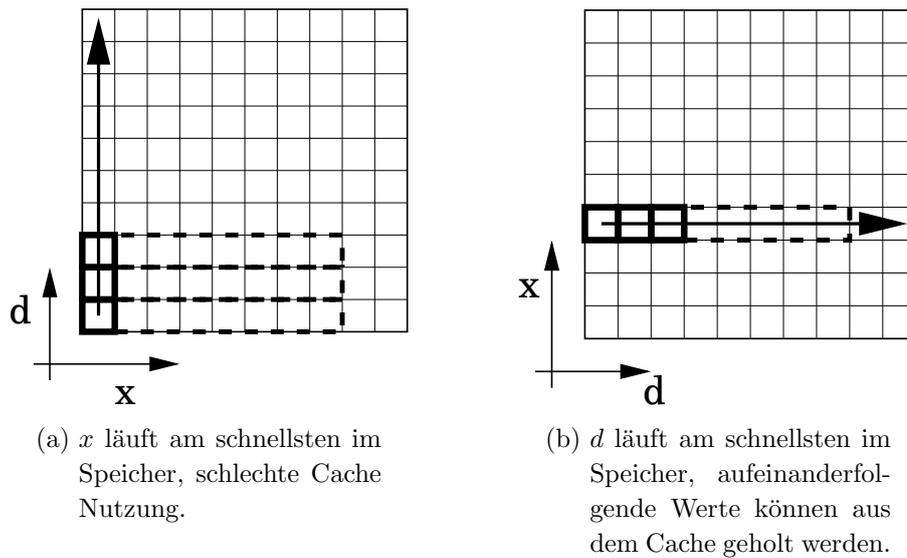


Abbildung 8.3: Unterschiedliche Ausnutzung des Caches beim Durchsuchen der SAD in Disparitätsrichtung. Die Lage der Cachelines ist gestrichelt angedeutet.

Zu Beginn einer Zeile muß einmal die Summe der ersten fünf SAD Werte gebildet werden. Danach kann jede folgende Summe durch je eine Addition und eine Subtraktion berechnet werden. Die Summe eines Fensters wird an der Position in  $x$ -Richtung direkt hinter ihm gespeichert, da die dort stehende pixelweise SAD nicht mehr benötigt wird.

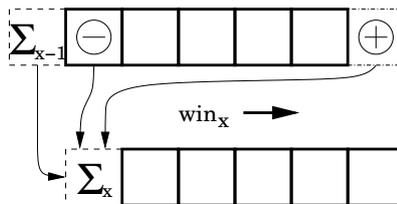
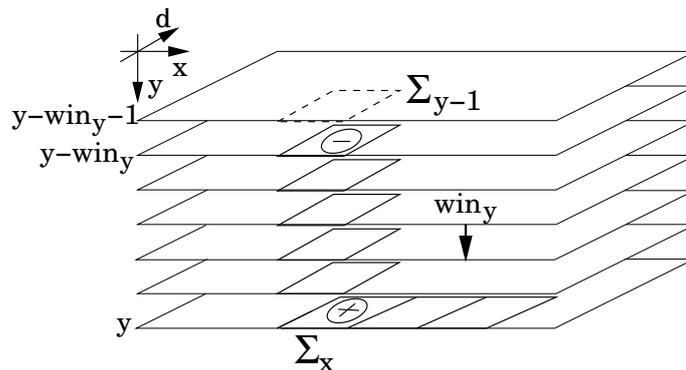


Abbildung 8.4: Gleitende Fenstersumme in  $x$ -Richtung.

Die Summierung in  $y$ -Richtung erfolgt analog (Abb. 8.5). Zu jeder Zeilensumme wird sofort das Fenster eine Zeile oberhalb addiert und dessen oberste Zeile wieder subtrahiert. Die so gebildete Fenstersumme wird an die Stelle der nicht mehr benötigten obersten Zeile gespeichert.

Durch die sofortige Verwendung der Zeilensummen werden unnötige Speicherzugriffe verhindert, das Wiederverwenden des Speichers für die Summen ermöglicht

Abbildung 8.5: Aufsummieren der Fenster in  $y$ -Richtung.

eine gute Ausnutzung der Caches. Desweiteren ist die Rechenzeit nicht abhängig von der Fenstergröße, was sie beim expliziten Aufsummieren wäre.

Nach dem Abarbeiten einer  $x$ - $d$ -Schicht werden die Werte in der obersten Schicht (siehe Abbildung 8.5) nicht mehr benötigt, können also mit neuen Daten überschrieben werden. Auf diese Weise ergibt sich der Ringpuffer aus  $win_y + 2$  solcher Schichten, der von oben nach unten durch das Disparitätsvolumen wandert.

Bei Verwendung des Speicherlayouts aus Abschnitt 8.4.3 kann die Summierung in  $x$  und  $y$  mit MMX-Befehlen beschleunigt werden, indem immer zwei beziehungsweise vier Werte parallel addiert werden.

## 8.6 Minimumsuche

Nach einem Durchlauf der innersten Schleife über die Disparität stehen der kleinste SAD-Wert und die Disparität, an der er vorkam, zur Verfügung. Die so gefundene Disparität steht für das Pixel im rechten Bild, das dem aktuellen Pixel im linken Bild bezüglich des Korrelationskriteriums am ähnlichsten ist. Um den Eindeigkeitstest anwenden zu können, der in Abschnitt 8.7 beschrieben wird, müssen allerdings auch die Disparitäten bekannt sein, die sich ergeben, wenn das rechte Bild als Referenz gewählt wird.

In einer Schicht des Volumens ist das Korrelationsmaß zwischen allen möglichen Fensterpaaren innerhalb des Suchbereichs abgelegt. Lediglich die Suche muß ein zweites mal durchgeführt werden, die Korrelation nicht. Abbildung 8.6 verdeutlicht, an welchen Stellen sich die SAD-Werte befinden, die zu einer festen Position im rechten Bild gehören.

Das Voxel, das zum Beispiel zum Pixel  $(x_R, y_R)$  und der Disparität  $d_{RL} = -2$

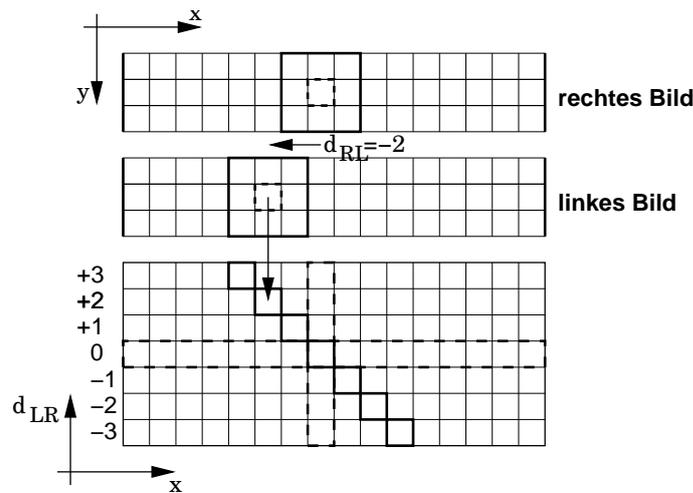


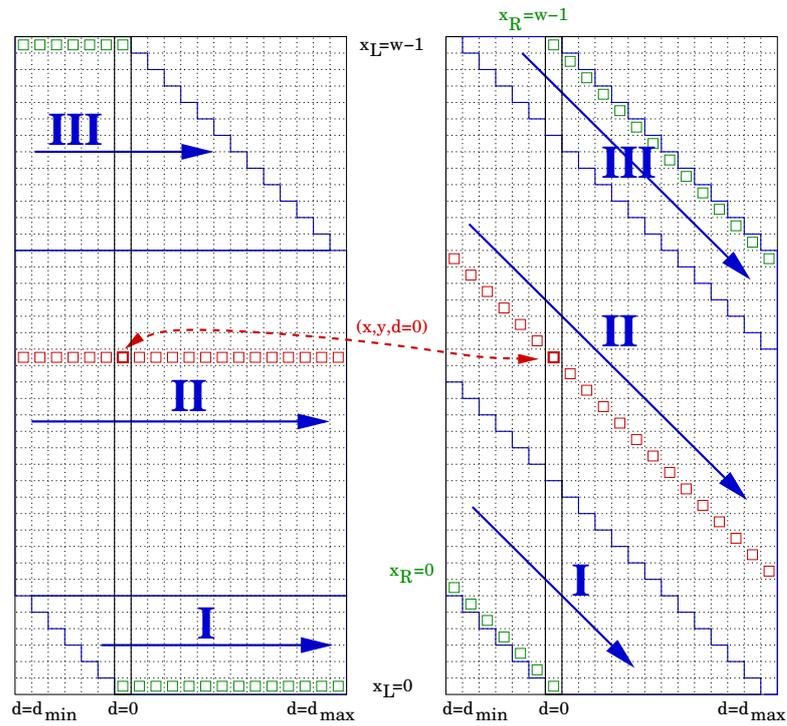
Abbildung 8.6: Minimumsuche von rechts nach links.

gehört, wurde bei der Suche von links nach rechts an der Stelle  $x_L = x_R - 2$  berechnet und zwar für die Disparität  $d_{LR} = +2$ . Alle Voxel, die zu einem Pixel im rechten Bild gehören, liegen auf einer Diagonalen, die sich im Punkt  $(x, y, d = 0)$  mit der Geraden schneidet, auf der von links nach rechts für diese Pixelkoordinaten gesucht wurde.

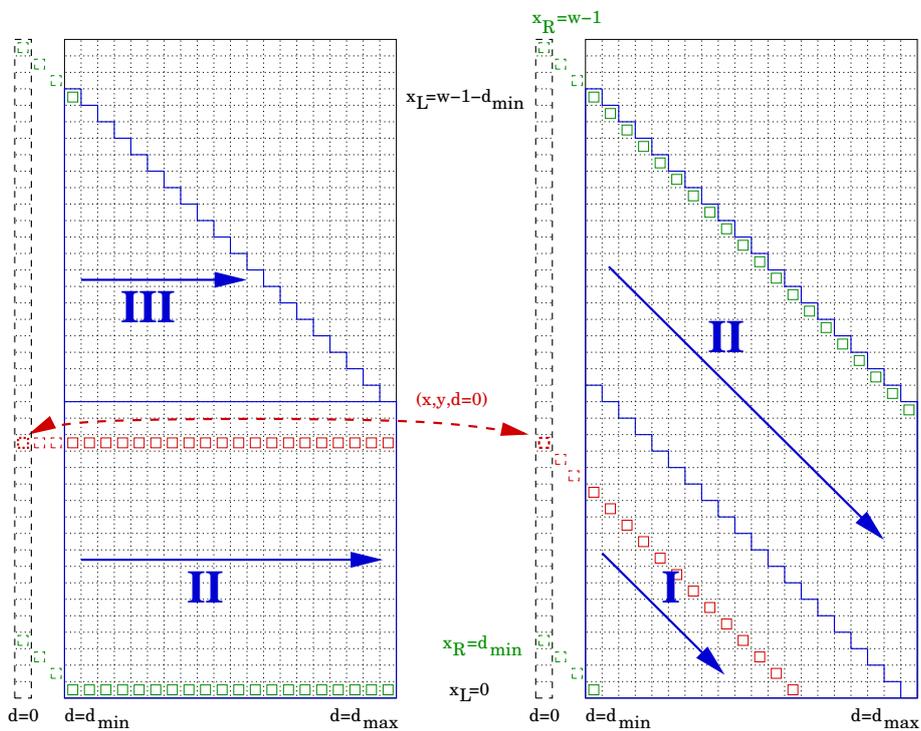
Bei der Implementierung des Stereoalgorithmus ist zu beachten, daß die  $x$ - $d$ -Schichten nicht vollständig mit Korrelationswerten gefüllt sind. Am Rand existieren Bereiche, bei denen sich eines der beiden Pixel außerhalb des Bildes befindet. Es macht keinen Sinn, die Korrelation aus nur einem Pixel zu berechnen, ebenso kann bei großen Bildern und Suchbereichen Rechenzeit gespart werden, wenn diese Bereiche von vorneherein bekannt sind und übersprungen werden.

Abbildung 8.7 zeigt, wie diese Bereiche genau aussehen. Innerhalb der blau umrandeten Flächen liegen die Voxel, bei denen beide Pixel im jeweilige Bild liegen (vgl. Abb. 11.2(b) und 11.2(c) auf Seite 107). In Abbildung 8.7(a) sind für beide Suchrichtungen jeweils drei Bereiche eingezeichnet, die alle (II) oder nur einen Teil der Disparitäten (I und III) beinhalten. Liegt  $d = 0$  nicht im Inneren des Suchbereichs, müssen nur zwei Fälle unterschieden werden (Abb. 8.7(b)). Die mit Quadraten markierten Voxel zeigen in Rot die Suchgeraden eines exemplarischen Pixels im linken und des selben Pixels im rechten Bild, sowie grün die Geraden, die zu den Pixeln am linken und rechten Rand gehören.

Die Suche von rechts nach links wird durchgeführt, nachdem die Korrelation und simultane Suche von links nach rechts einer Schicht abgeschlossen sind. Die sich dabei ergebenden Disparitäten speichert ein Array der Größe  $w$  zwischen, sie werden



(a)  $d_{\min} < 0 < d_{\max}$



(b)  $d_{\max} > d_{\min} \geq 0$

Abbildung 8.7: Suchbereiche und -richtungen. Linkes Bild: Suche von links nach rechts, rechtes Bild: Suche von rechts nach links.

nach dem Eindeutigkeitsstest (8.7) und der Subpixelinterpolation (8.8) nicht mehr gebraucht.

## 8.7 Eindeutigkeitsstest

Wenn die Wahrscheinlichkeit hoch ist, daß eine gefundene Disparität nicht korrekt sein kann, dann sollte dies für das betroffene Pixel erkannt werden. Um später keine Oberflächen zu generieren wo keine waren, werden diese Punkte nicht in das 3D-Modell aufgenommen. Es ist wesentlich einfacher, sie während der Korrelation zu verwerfen, als zu einem späteren Zeitpunkt. Die zwei wichtigsten Ursachen für falsche Zuordnungen sind Verdeckungen und homogene Flächen.

Die Methode, die Verdeckungen am zuverlässigsten erkennt, ist der Test ob die Disparitäten eines Pixelpaares zusammenpassen [16, 14].

$$d_{RL}(x + d_{LR}(x, y), y) \approx -d_{LR}(x, y). \quad (8.4)$$

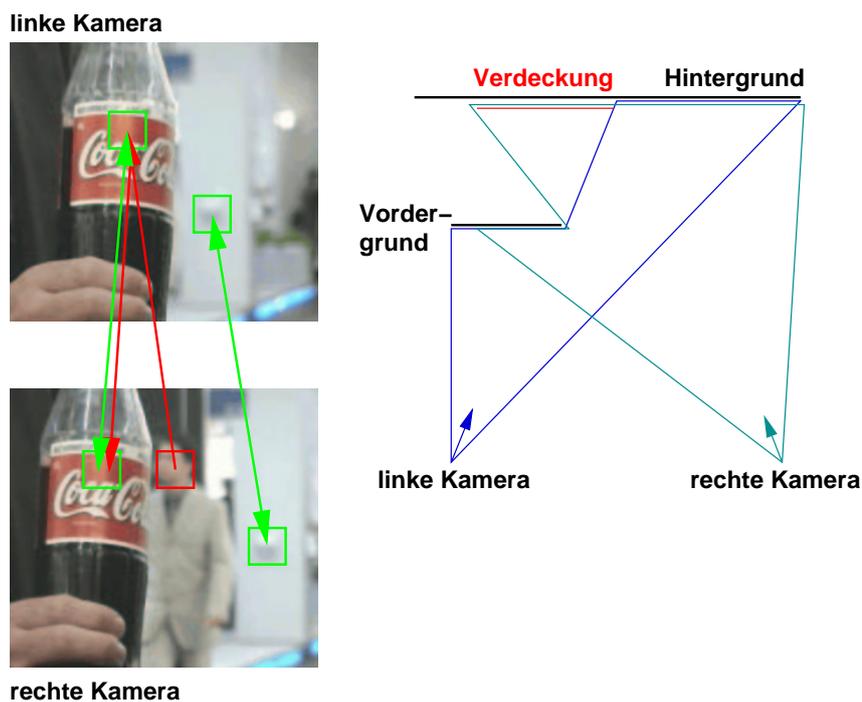


Abbildung 8.8: Vorder-, Hintergrund und in einem Bild verdeckte Fläche.

Der linke Teil von Abbildung 8.8 zeigt die Aufnahmen zweier Kameras von einem Objekt im Vordergrund, das den Hintergrund teilweise verdeckt. Rechts davon

ist die Anordnung von oben gesehen skizziert. Diejenigen Teile des Vorder- beziehungsweise Hintergrunds, die in beiden Bildern zu sehen sind, matchen mit hoher Wahrscheinlichkeit aufeinander, angedeutet durch die grünen Pfeile. Ihre Disparitäten erfüllen Gleichung (8.4). Pixel aus dem Bereich des rechten Bildes, die im linken nicht zu sehen sind, wie am Kopf der Person eingezeichnet, matchen auf das am besten passende sichtbare Objekt. Das zugeordnete Pixel im linken Bild hat aber einen richtigen Partner im rechten, die Disparitäten werden nicht zueinander passen (rote Pfeile). Unterscheiden sich die Disparitäten um mehr als ein Pixel, wird der Match sehr wahrscheinlich falsch sein und verworfen.

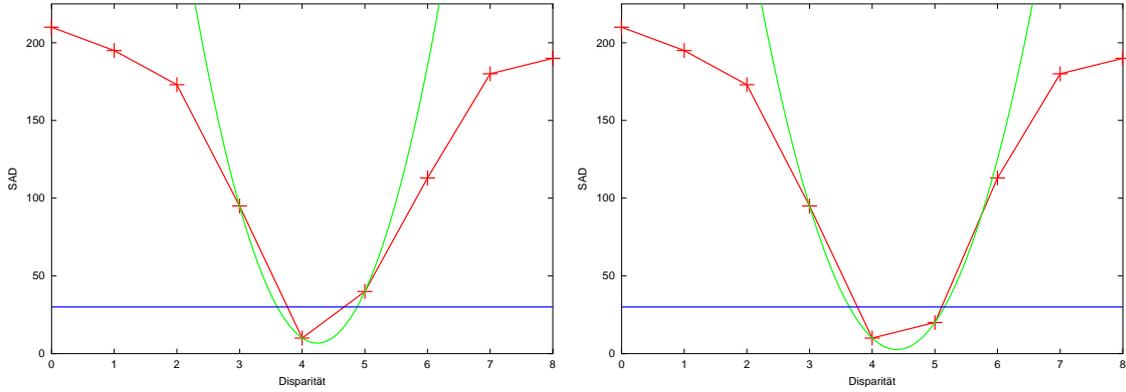
Auf Flächen, die keine oder eine sich wiederholende Textur haben, kann ebenfalls keine zuverlässige Schätzung abgegeben werden. Um diese Fälle zu erkennen, wird die Eindeutigkeit des Minimums einer SAD getestet. Es reicht allerdings nicht aus, zu fordern daß das Minimum der mit gewissem Abstand kleinste Wert sein soll. Zwischen Bereichen, die sich in der Disparität um eins unterscheiden, treten doppelte Minima auf, die einen genauso eindeutigen Match wie ein einzelnes Minimum ergeben. Die Abbildungen 8.9(a) und 8.9(b) verdeutlichen diese beiden Fälle, die eingezeichnete Schwelle wird aus dem Wert des Minimums berechnet  $((1+\mu)SAD_{\min}, \mu \geq 0, \text{ typisch } 0,1-0,2)$ . Liegen mehr als zwei Werte unterhalb der Schwelle, kann der Match als nicht eindeutig verworfen werden (Abb. 8.9(c)).

## 8.8 Subpixelgenauigkeit

Bis jetzt war die Disparität als Differenz der Pixelposition im linken und rechten Bild eine ganzzahlige Größe. Betrachtet man die SAD als abgetastete Werte einer kontinuierlichen Funktion, kann man ihre Werte auch zwischen den Stützstellen durch Interpolation erhalten. Verwendete man die Summe der quadrierten Differenzen anstelle der absoluten, ließe sich die Korrelationsfunktion in der Umgebung des Minimums durch eine Parabel annähern. Die Position des Minimums der Parabel durch das Minimum und seine zwei Nachbarwerte wäre dann eine bessere Schätzung für das wirkliche Minimum als die Integer-Disparität.

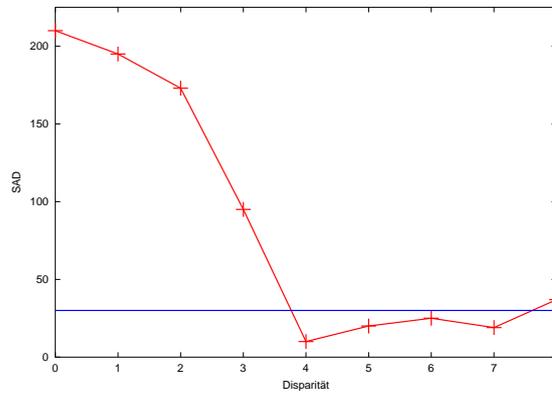
Bei der SAD ist die abgetastete kontinuierliche Funktion die Absolutwertfunktion, deren Parameter leider ungleich schwieriger mit nur drei Werten zu schätzen sind, als die einer Parabel. Weiter entfernte Voxel mit einzubeziehen hilft nicht, da die Approximation nur in einer kleinen Umgebung um das Minimum gültig ist. Auch wenn die Parabel nicht korrekt ist, gibt sie dennoch eine bessere Schätzung für die Position des Minimums als der ganzzahlige Index des Voxels mit der kleinsten SAD. Ihr Scheitel läßt sich direkt aus den drei Werten um das Minimum ablesen (siehe

auch Abb. 8.9(a) und 8.9(b))



(a) Eindeutiges Minimum.

(b) Doppelminimum.



(c) Nicht eindeutiges Minimum.

Abbildung 8.9: Interpolationsparabel und Eindeutigkeits-Schwelle.

$$d_s(x, y) = d_i(x, y) + \frac{\text{SAD}(x, y, d_i - 1) - \text{SAD}(x, y, d_i + 1)}{2(\text{SAD}(x, y, d_i - 1) - 2\text{SAD}(x, y, d_i) + \text{SAD}(x, y, d_i + 1))}. \quad (8.5)$$

Die Subpixeldisparitäten aus den Suchen von links nach rechts und umgekehrt werden gemittelt, die Disparität von rechts nach links ergibt sich durch lineare Interpolation

$$d_{LR}(x) = \lfloor d_{LR}(x) \rfloor + (d_{LR}(x) - \lfloor d_{LR}(x) \rfloor) = i_d + f_d$$

$$d = \frac{1}{2} \left[ d_{LR}(x) + (1 - f_d)d_{RL}(x + i_d) + f_d d_{RL}(x + i_d + 1) \right]. \quad (8.6)$$

## 8.9 Medianfilter

Im Disparitätsbild kommen vereinzelt sowohl isolierte Pixel vor als auch solche, die verworfen wurden während ihre Umgebung sicher geschätzt werden konnte. In Übereinstimmung mit [17] sind diese Pixel mit sehr hoher Wahrscheinlichkeit falsch. Ein Medianfilter ist sehr gut geeignet, diese impulsförmigen Fehler zu entfernen [33]. Er ersetzt jedes Pixel durch das mittlere Element der sortierten Liste seiner Nachbarn. Im verwendeten Fall einer  $3 \times 3$  Pixel Umgebung hat diese Liste neun Einträge. Um das Element in der Mitte zu finden, muß die Liste nicht vollständig sortiert werden, Abbildung 8.10 zeigt, wie die Sortierung mit einer minimalen Anzahl von Vertauschungsoperationen durchgeführt werden kann. Dieses Verfahren ist für kleine Listen schneller, als eine komplett sortierte Liste zu behalten, bei der immer das älteste Element gelöscht und das neue einsortiert wird [13].

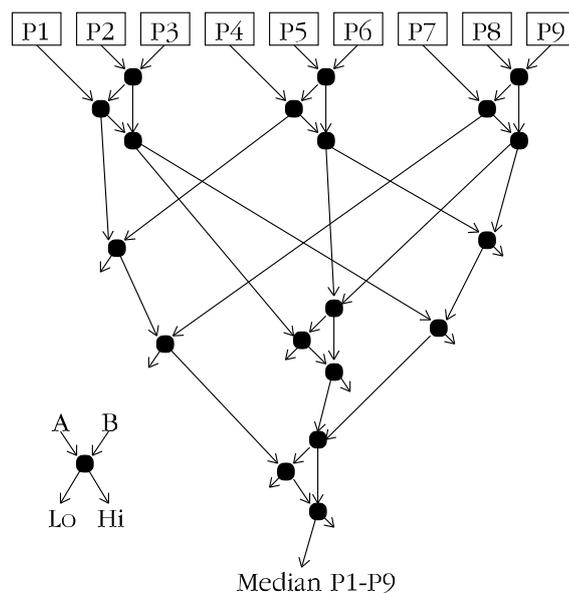


Abbildung 8.10: Minimaler Sortiergraph für den Median von 9 Werten [60].

## 8.10 Algorithmus

Algorithmus 8.1 faßt den Ablauf der Disparitätsberechnung zusammen. Die innerste Schleife ist besonders detailliert ausgeführt, um zu zeigen wie eng die einzelnen Teile

des Algorithmus ineinandergreifen und wie kompakt sie sich formulieren lassen.

Zusätzlich zu den Schichten des Disparitätsvolumens wird ein Akkumulator benötigt, hier mit `sad` bezeichnet. Die Variable `xStride` steht für den Offset zweier Pixel, die sich in  $x$  um eins unterscheiden. Er beträgt mindestens  $d_{\max} - d_{\min} + 1$ , in der MMX-Version des Algorithmus sind die Anfänge der Zeilen auf ein Vielfaches von acht Bytes ausgerichtet, um einen schnellen Zugriff zu ermöglichen [30]. Während sich die Summe in  $x$  einfach mit Hilfe des Offsets berechnen läßt, werden die an der Summe in  $y$  beteiligten Voxel durch eigene Pointer realisiert.

**Algorithmus 8.1** Berechnung der Disparitäten

---

```
for  $y = 0 \dots h - 1$  do
  for  $x = 0 \dots w - 1$  do
    leftPixel laden
    for  $d = d_{\min} \dots d_{\max}$  do
      sad = calcSAD(leftPixel, rightPixel); // Gl.(8.3)
      *sadPtr = sad;
      sad += *(sadPtr - xStride*(winX+1)) // \
             - *(sadPtr - xStride*winX); // Abb. 8.4
      *(sadPtr - xStride*winX) = sad; // /
      sad += *minusWinYminus1slicePtr // \
             - *minusWinYslicePtr; // Abb. 8.5
      *minusWinYslicePtr = sad; // /
      if (sad <= minVal) {
        minVal3 = minVal2;
        minPos = d;
        minVal2 = minVal;
        minVal = sad;
      }
      Pointer inkrementieren
    end for //  $d$ 
    if minVal3 >  $(1 + \mu)$ minVal then
      Subpixelinterpolation (Gl. (8.5))
    else
      Pixel verwerfen
    end if
  end for //  $x$  L→R
  for  $x = 0 \dots w - 1$  do
    for  $d = d_{\min} \dots d_{\max}$  do
      Suche von rechts nach links (Abb. 8.7)
    end for
  end for //  $x$  R→L
  for  $x = 0 \dots w - 1$  do
    if Links-Rechts-Test ok (Gl. (8.4)) then
      Kombination der Subpixeldisparitäten (Gl. (8.6))
    else
      Pixel verwerfen
    end if
  end for //  $x$  Kombination
  Schichtwechsel
end for //  $y$ 
```

---

# 9

## 3D-Modell Erstellung

Die aus der Stereorekonstruktion entstandenen Disparitäten lassen sich beispielsweise mit Falschfarben oder als Graustufenbild darstellen (z.B. Abb. 11.2(d), S. 107). Sie stellen allerdings für die meisten Anwendungen keine direkt verwertbare Information dar. Erst zusammen mit der Kamerakalibrierung wird es möglich, eine dreidimensionale Rekonstruktion der aufgenommenen Szene zu erhalten. Abschnitt 9.1 behandelt die Berechnung der 3D-Punkte aus den korrespondierenden Bildpunkten.

Die Stereokamera erfäßt — abgesehen von transparenten und spiegelnden Flächen, deren Rekonstruktion prinzipbedingt nicht funktionieren kann — die Oberfläche der aufgenommenen Gegenstände. Daher liegt es nahe, auch eine explizite Oberflächendarstellung für das generierte Modell zu wählen. Abschnitt 9.2 beschreibt ein effizientes Verfahren, die 3D-Punkte zu einem Dreiecksnetz zu verbinden.

### 9.1 Berechnung der 3D-Punkte

Ausgangspunkt für die Modellgenerierung sind die Disparitäten, die der Stereoalgorithmus für die Pixel eines der beiden rektifizierten Bilder berechnet hat. Mit Hilfe der Kalibrierung läßt sich jedem Bildpunkt ein Strahl zuordnen, auf dem der abgebildete Raumpunkt gelegen haben muß. Schneidet man diesen Strahl mit dem Strahl, der sich aus dem korrespondierenden Bildpunkt des anderen Bildes und den Parametern der anderen Kamera ergibt, erhält man die Koordinaten des 3D-Punktes.

Das Ergebnis der Disparitätsberechnung von Kapitel 8 sind Punktpaare aus den rektifizierten Bildern, deshalb müssen auch die Kameramatrizen der rektifizierten Bilder verwendet werden. Sie ergeben sich aus denen der ursprünglichen Kalibrierung durch Multiplikation mit der jeweiligen rektifizierenden Transformation (Gl. (7.10)). Die Linsenverzerrung wird vor der Rektifikation aus den Bildern herausgerechnet,

die Projektion der Weltpunkte in die rektifizierten Bilder ist deshalb eine rein lineare Abbildung.

Die Rektifikation nach Abschnitt 7.2.3 besteht aus mehreren Schritten, dadurch können sich in ungünstigen Fällen Rundungsfehler akkumulieren. Betrachtet man die Fundamentalmatrix, die sich aus den Projektionsmatrizen der rektifizierten Bilder nach Gleichung (7.8) ergibt, erfüllen die Bildpunkte die epipolare Bedingung nur annähernd ( $\mathbf{x}'^T \mathbf{F} \mathbf{x} = \epsilon \neq 0$ ), die Strahlen aus dem linken und rechten Bild treffen sich nicht exakt. Es gilt nun, denjenigen 3D-Punkt zu finden, der „am besten“ zu den beiden Strahlen paßt.

Eine anschauliche Möglichkeit ist, denjenigen Punkt zu wählen, der von beiden Strahlen den geringsten Abstand hat (Abb. 9.1).

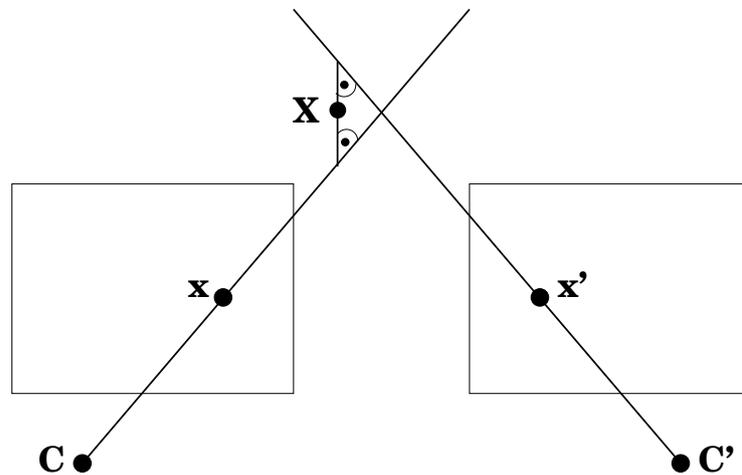


Abbildung 9.1: Vorwärtsschnitt von den Bildpunkten zum 3D-Punkt.

In [25] werden verschiedene Methoden der Triangulierung miteinander verglichen, von der eben beschriebenen „Mittelpunkt-Methode“ wird deutlich abgeraten. Der Grund dafür ist, daß alle Daten der Kalibrierung und der Disparitätsberechnung aus den Bildern gewonnen werden, eine Maximum-Likelihood Schätzung sollte also auch die Fehler in den Bildern minimieren und nicht im Raum. Während bei einer projektiven Rekonstruktion die dort vorgestellte Methode allen anderen klar überlegen ist, können einfachere lineare Methoden im euklidischen Fall, wenn die Kamerakalibrierung bekannt ist, gleich gute oder sogar marginal bessere Genauigkeit der rekonstruierten 3D-Koordinaten liefern.

Bezeichnen  $p_{ij}$  die Elemente der Matrix  $\mathbf{H}\mathbf{P}$  und  $p'_{ij}$  die der Matrix  $\mathbf{H}'\mathbf{P}'$ , so ergeben sich analog zu Gleichung (6.10) der DLT (S. 49) zwei lineare Bedingungen

pro Bildpunkt für die vier unbekanntenen homogenen Koordinaten des 3D-Punktes

$$\begin{bmatrix} xp_{31} - p_{11} & xp_{32} - p_{12} & xp_{33} - p_{13} & xp_{34} - p_{14} \\ yp_{31} - p_{21} & yp_{32} - p_{22} & yp_{33} - p_{23} & yp_{34} - p_{24} \\ x'p'_{31} - p'_{11} & x'p'_{32} - p'_{12} & x'p'_{33} - p'_{13} & x'p'_{34} - p'_{14} \\ y'p'_{31} - p'_{21} & y'p'_{32} - p'_{22} & y'p'_{33} - p'_{23} & y'p'_{34} - p'_{24} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ W \end{pmatrix} = \mathbf{0}. \quad (9.1)$$

Gleichung (9.1) muß für jeden Punkt gelöst werden, für den eine Disparität ermittelt wurde. Bei den verwendeten Bildern der Stereokamera sind das bis zu über 100 000 mal. Als Lösungsmethoden kommen zum Beispiel die Singulärwertzerlegung oder Jacobi-Iteration in Frage [21], die aber verhältnismäßig viele Rechenoperationen benötigen und alle Eigenwerte und -vektoren berechnen, obwohl nur ein Vektor als Lösung benötigt wird.

Verwendet man  $W = 1$  als zusätzliche Bedingung, anstelle der von der SVD implizierten  $\|\mathbf{X}\| = 1$ , und setzt  $x' = x + d$  und  $y' = y$ , ausgehend davon daß die Bilder rektifiziert sind, wird Gleichung (9.1) zu

$$\begin{bmatrix} xp_{31} - p_{11} & xp_{32} - p_{12} & xp_{33} - p_{13} \\ yp_{31} - p_{21} & yp_{32} - p_{22} & yp_{33} - p_{23} \\ (x+d)p'_{31} - p'_{11} & (x+d)p'_{32} - p'_{12} & (x+d)p'_{33} - p'_{13} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} p_{14} - xp_{34} \\ p_{24} - yp_{34} \\ p'_{14} - (x+d)p'_{34} \end{pmatrix}. \quad (9.2)$$

Die Lösung mit einem Gaußschen Algorithmus ist einfach und schnell. Ein zusätzlicher Vorteil ist, daß sich sofort die inhomogenen Koordinaten ergeben.

Punkte im Unendlichen, bei denen  $W$  Null ist, können damit nicht rekonstruiert werden. Diese Punkte sind allerdings für keine der projizierten Anwendungen von Interesse, sie werden deshalb verworfen.

## 9.2 Generierung eines Dreiecksnetzes

Die Verwendung eines Dreiecksnetzes als einfache Approximation der aufgenommenen Oberfläche hat mehrere Vorteile. Das Netz läßt sich einfach und schnell erzeugen, ist zum Beispiel mit OpenGL [54, 53] sofort visualisierbar, und mit VRML [68] existiert ein offenes und standardisiertes Dateiformat zum Austausch mit anderen Programmen.

Die Punkte werden in der Reihenfolge der Pixel des linken Bildes generiert, von links nach rechts und von oben nach unten. Ihre Koordinaten stehen dicht gepackt in einem Feld („vertex array“), eine für die schnelle Darstellung mit OpenGL günstige Anordnung im Speicher. Die Dreiecke werden ebenfalls in einem Feld abgelegt, jedes Dreieck besteht aus drei Indizes in das Feld der Punkte. Punkte und Dreiecke können

so zusammen mit einem einzigen Funktionsaufruf an die Grafikkarte übertragen werden.

Für jedes Pixel des Disparitätsbildes wird versucht, den generierten 3D-Punkt sofort mit seinen linken und oberen Nachbarn oder die Nachbarn untereinander zu verbinden. Abbildung 9.2 zeigt die fünf Fälle, in denen ein oder zwei Dreiecke erzeugt werden (vgl. Alg. 9.1, S. 96).

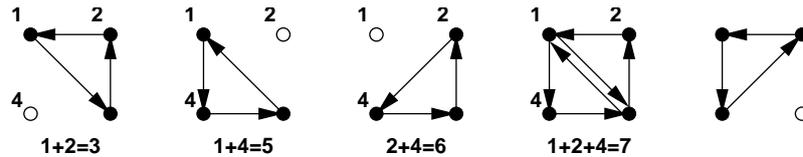


Abbildung 9.2: Alle Fälle, in denen Dreiecke generiert werden.

Die vier Eckpunkte der Quadrate stellen vier benachbarte Positionen des Disparitätsbildes dar. Rechts unten befindet sich jeweils die aktuelle Position  $(x, y)$ . Links daneben liegt  $(x - 1, y)$ , darüber  $(x - 1, y - 1)$  und  $(x, y - 1)$ . Ein ausgefüllter Punkt steht für eine gültige Disparität, vom Stereoalgorithmus verworfene Punkte (Abschnitt 8.7) sind durch leere Kreise dargestellt. Den drei Positionen links von und über der aktuellen wird je eine Zweierpotenz zugeordnet. Die Summe der Zahlen, an deren Position eine gültige Disparität steht, wird benutzt um zu entscheiden, welche Dreiecke erzeugt werden müssen.

Es ist nicht sinnvoll, alle benachbarten Punkte auch zu verbinden, weil bei starken Disparitäts- und damit Tiefenunterschieden Dreiecke entstehen, die sehr langgestreckt sind und in sehr steilem Winkel zur Bildebene stehen. Diese Dreiecke verbinden mit großer Wahrscheinlichkeit Objekte in verschiedenen Tiefen miteinander oder beinhalten Ausreißer. Es ist sowohl für die Visualisierung als auch die Weiterverarbeitung der Daten besser, sie von vornherein nicht zu erzeugen. Ein Dreieck wird nur dann generiert, wenn die paarweisen Disparitätsdifferenzen seiner drei Punkte alle unterhalb einer Schwelle liegen. In der Praxis hat sich ein Schwellwert von 1 bis 1,5 als geeignet erwiesen.

Abbildung 9.3 und Algorithmus 9.1 verdeutlichen, wie die Netzgenerierung mit der der Punkte mitläuft. Abgebildet sind die aktuelle und zwei vorhergehende Zeilen. Durch die gepackte Speicherung im Vertex-Array geht die Nachbarschaftsbeziehung zwischen den Punkten verloren. Sie muß so lange zwischengespeichert werden, wie sie während der Rekonstruktion noch gebraucht wird, dazu dient das Feld `idx[]`. Es enthält die Indizes der Punkte der vorhergehenden Zeile oder Null, wenn an der Stelle kein Punkt rekonstruiert wurde. Die Indizes der aktuellen Zeile überschreiben

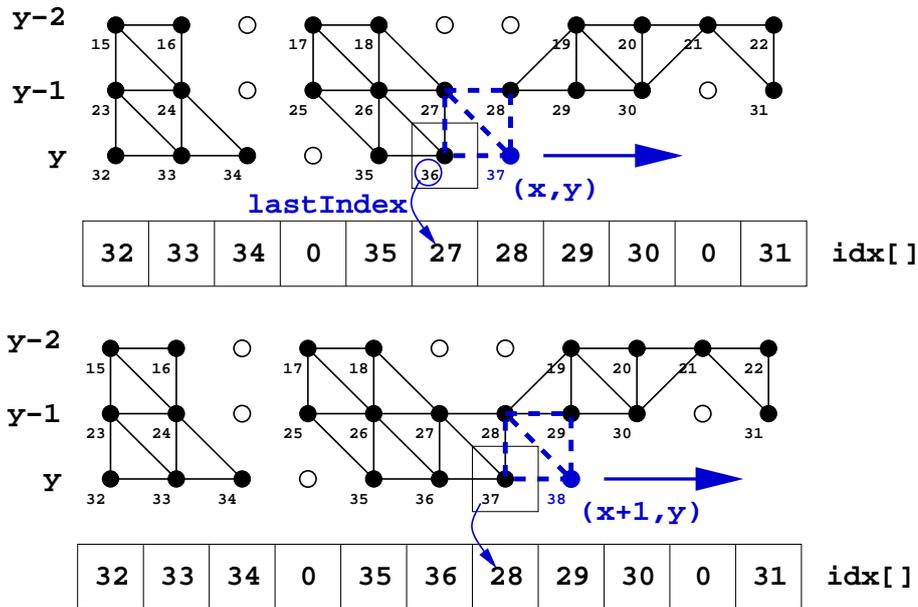


Abbildung 9.3: Fortschreiten der Triangulierung.

diejenigen Positionen, die nicht mehr gebraucht werden. Zusätzlich muß noch der Index des zuletzt generierten Punktes gemerkt werden, weil der alte Index des Punktes  $(x-1, y-1)$  aus dem Array erst nach der Erzeugung der Dreiecke überschrieben werden kann. Um Algorithmus 9.1 übersichtlich zu halten, wurden die Tests weggelassen, die bei zu großem Disparitätsunterschied verhindern, daß ein Dreieck generiert wird.

Als Textur wird das rektifizierte Bild der linken Kamera verwendet. Die Texturkoordinaten ergeben sich durch eine einfache Skalierung aus den Koordinaten des Disparitätsbildes. Es werden keine Normalen für das Dreiecksnetz berechnet und keine künstlichen Lichtquellen verwendet. Für eine künstliche Beleuchtung müßte die reale aus den Bildern herausgerechnet werden, was nicht möglich ist, zumindest nicht in der gewünschten Geschwindigkeit und mit nur einem Bildpaar. Das rekonstruierte Modell erhält die durch die Textur vorhandene natürliche Beleuchtung (s. Abb. 12.3(b), S. 123).

---

**Algorithmus 9.1** Erzeugung der Punkte und des Dreiecksnetzes.

---

```
typedef struct sPNT {float X, Y, Z;};
typedef struct sTRI {unsigned int p0, p1, p2;};
sPNT vertex[];
sTRI triangle[];
int nv=0, nt=0, lastIndex=0, idx[w];
idx[] ← 0
for y = 1 to h - 1 do
  lastIndex = 0;
  for x = 1 to w - 1 do
    if gültige Disparität bei (x, y) then
      vertex[nv] ← Rekonstruktion aus Gl. (9.2)
      int fall = (idx[x-1]?1:0);
      if (idx[x]) fall |= 2;
      if (lastIndex) fall |= 4;
      switch fall
        case 3: triangle[nt++] ← {nv, idx[x], idx[x-1]}; break;
        case 5: triangle[nt++] ← {nv, idx[x-1], nv-1}; break;
        case 6: triangle[nt++] ← {nv, idx[x], nv-1}; break;
        case 7: triangle[nt++] ← {nv, idx[x], idx[x-1]};
              triangle[nt++] ← {nv, idx[x-1], nv-1}; break;
      end switch
      idx[x-1] = lastIndex;
      lastIndex = nv++;
    else /* kein Punkt erzeugt */
      if (lastIndex && idx[x] && idx[x-1]) then
        triangle[nt++] ← {idx[x], idx[x-1], nv-1};
      end if
      idx[x-1] = lastIndex;
      lastIndex = 0;
    end if
  end for
end for
```

---

# **Teil IV**

## **Ergebnisse**



# 10 Kalibrierung

Die Güte der Kalibrierung wird anhand von zwei Kriterien überprüft. Das erste ist das Residuum der nichtlinearen Optimierung aus Abschnitt 6.4.6, das angibt wie gut das Modell auf die gemessenen Daten paßt. Das zweite ist die Wiederholgenauigkeit bei mehreren Messungen mit unveränderten Kameras.

## 10.1 Reprojektionsfehler

Bei allen durchgeführten Messungen hatte das Residuum nach der Optimierung einen Wert von ungefähr 0.1 Pixeln pro detektiertem Kreuzungspunkt (vgl. Abbildung 6.8, S. 64). Bei einer angenommenen Brennweite von 450 Pixeln ergäbe sich daraus eine Unsicherheit der lateralen Koordinaten von etwa 1mm auf 5m Entfernung. Die Genauigkeit des Gesamtsystems ist natürlich geringer, da der Stereoalgorithmus keine vollkommen exakten Disparitäten liefert.

Abbildung 10.1 zeigt den Reprojektionsfehler einer Beispielaufnahme. Die Fehlervektoren stellen die Differenz zwischen der in der Aufnahme gemessenen Position eines Kreuzungspunktes und der vom Kameramodell vorhergesagten Projektion des zugehörigen Punktes der Kalibrierebene dar. Um sie sichtbar zu machen, ist ihre Länge zehnfach vergrößert worden. Die Vektoren sind sowohl in Richtung als auch Länge zufällig verteilt, was darauf schließen läßt, daß bei der Messung kein systematischer Fehler aufgetreten ist.

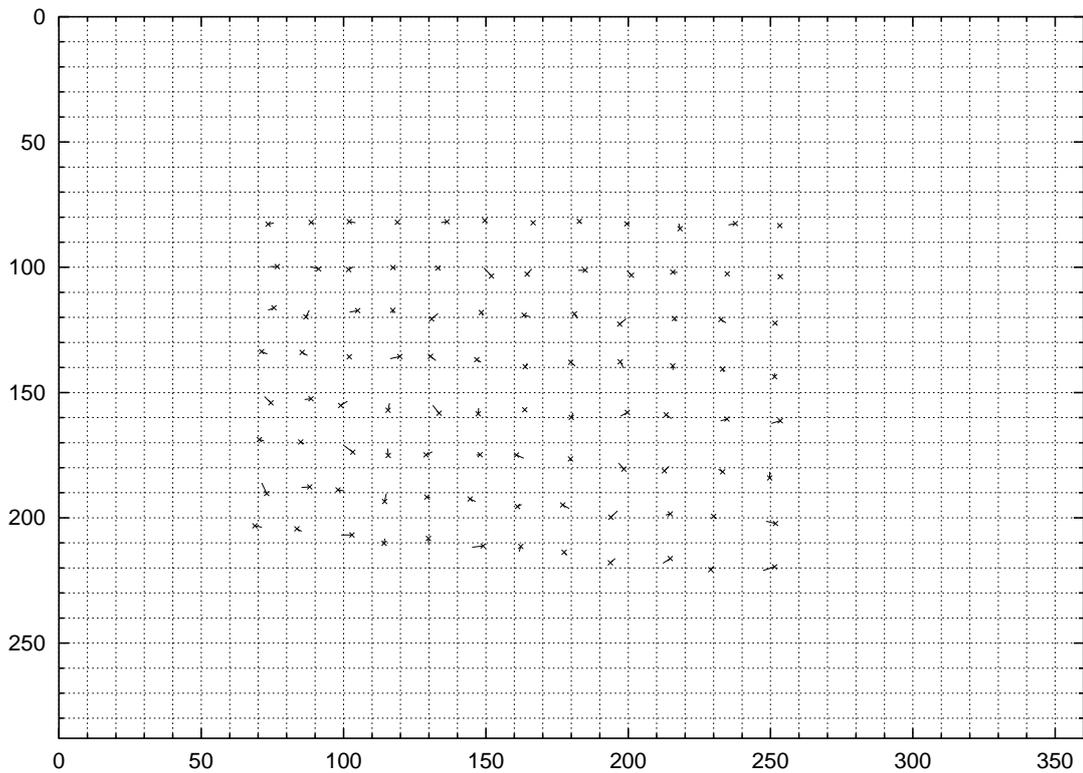


Abbildung 10.1: Reprojektionsfehler einer Aufnahme, zehnfach verstärkt.

## 10.2 Wiederholgenauigkeit

Abbildung 10.3(a) bis 10.5(b) tragen die einzelnen Parameter der Stereo-Kalibrierung über der Anzahl der verwendeten Aufnahmen pro Kalibrierung auf. Es wurden jeweils zehn Kalibrierungen durchgeführt, deren Mittelwert und Standardabweichung eingezeichnet sind.

Übereinstimmend mit [76] läßt sich feststellen, daß bereits mit drei Aufnahmen eine gute Schätzung der Kameraparameter möglich ist, vorausgesetzt der Winkel zwischen den drei Ebenen war hinreichend groß. Die dort gegebene Empfehlung, zwischen fünf und zehn Bilder zu verwenden, gilt ebenso für die Stereokalibrierung. Mehr Bilder verbessern die Schätzung nicht mehr viel, bedeuten aber eine größere Rechenzeit bei der nichtlinearen Optimierung.

Das Verhältnis der in Pixel gemessenen Brennweiten (Abb. 10.3(a)) läßt auf ein Seitenverhältnis der CCD-Elemente von nicht genau 1 sondern etwa 0,92 schließen.

Dieser Wert stimmt gut mit dem aus dem Datenblatt des Sensors ermittelten Verhältnis von  $6.5\mu\text{m}/7\mu\text{m} = 0,928$  überein. Die Standardabweichung der Brennweiten bei 15 verwendeten Ebenen beträgt etwa 0,8 oder 0.2%.

Die Streuung der Koordinaten des Hauptpunktes (Abb. 10.3(b)) ist etwas größer als die der Brennweiten. Er läßt sich schwieriger schätzen, sein Einfluß auf den Fehler der 3D-Koordinaten bei der Triangulation ist jedoch auch geringer [23]. Während er sich bei der rechten Kamera sehr nah an der Bildmitte (180, 144) befindet, liegt er bei der linken Kamera etwa vier Pixel links davon und acht oberhalb.

Der erste Koeffizient der radialen Linsenverzerrung  $\kappa_0$  ist sehr zuverlässig ermittelt worden (Abb. 10.4(a)), der zweite nicht ganz so sicher. Das Ergebnis der tangentialen Verzerrung (Abb. 10.4(b)) ist bei weitem am unsichersten. Wie bereits in Abschnitt 6.4.5 erwähnt, ist das Modell sehr wahrscheinlich überparametrisiert und die radialen Koeffizienten reichen zur Beschreibung aus.

Die Ergebnisse der Parameter bei Verwendung von drei Ebenen unterscheiden sich etwas von den Messungen mit mehr Ebenen. Die Standardabweichungen sind nicht viel größer, aber die Mittelwerte liegen nicht innerhalb der Standardabweichungen der anderen Messungen. Es ist zu vermuten, daß die Anordnung der drei Ebenen bei allen zehn Messungen ähnlich war und sich so ein kleiner systematischer Fehler ergeben hat.

Abbildung 10.5(a) und 10.5(b) zeigen die Translation in Millimetern und die Eulerwinkel der Rotation zwischen den Kameras. Um sich auf verschiedene Objektentfernungen einstellen zu können, ist die linke Kamera drehbar. Im vorliegenden Fall wurde sie um etwa  $16^\circ$  auf die rechte zu gedreht, wie in Abbildung 10.2 skizziert (vgl. Abb 6.5, S. 61). Weil das Koordinatensystem an der linken Kamera festgemacht ist, ergibt sich dadurch nicht nur eine Translation von 180 mm in  $X$ -Richtung, sondern auch eine von 50 mm in  $Y$ .

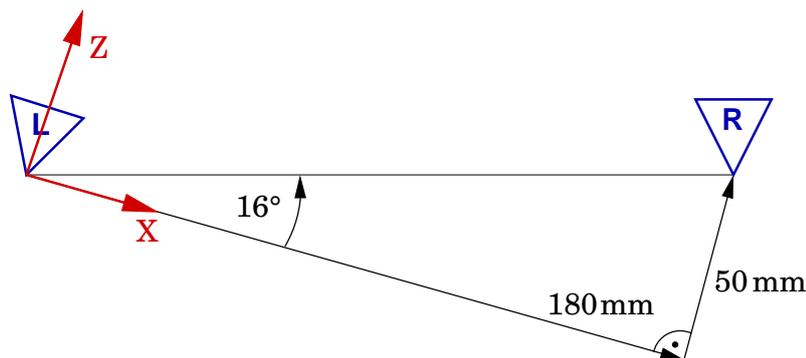
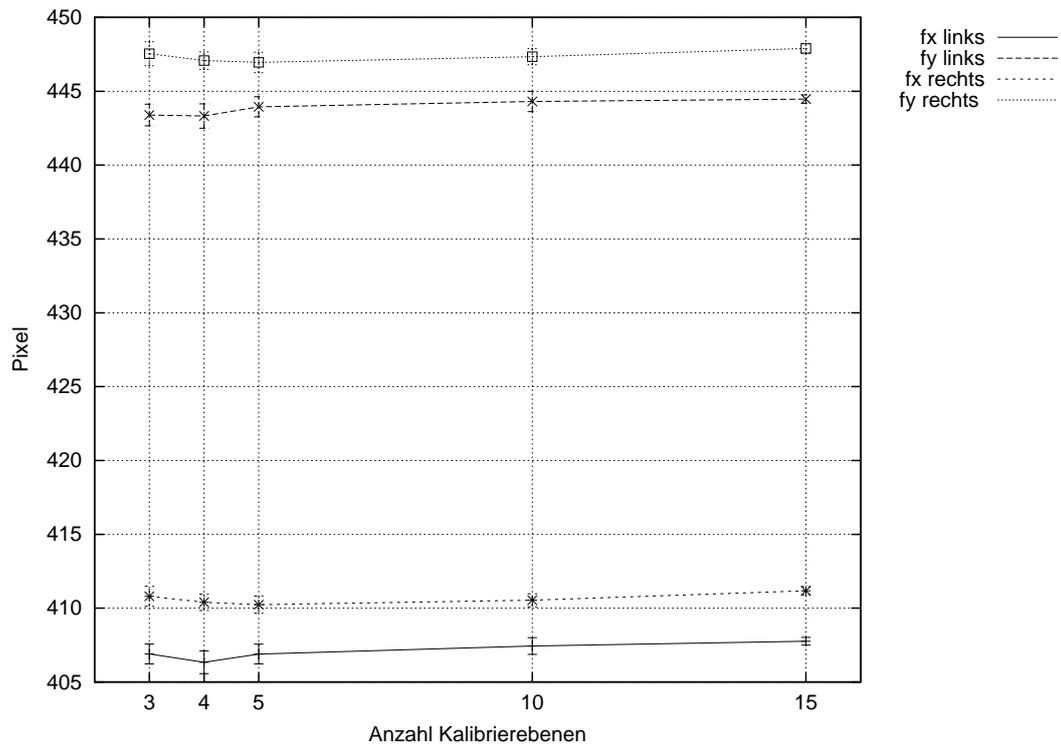
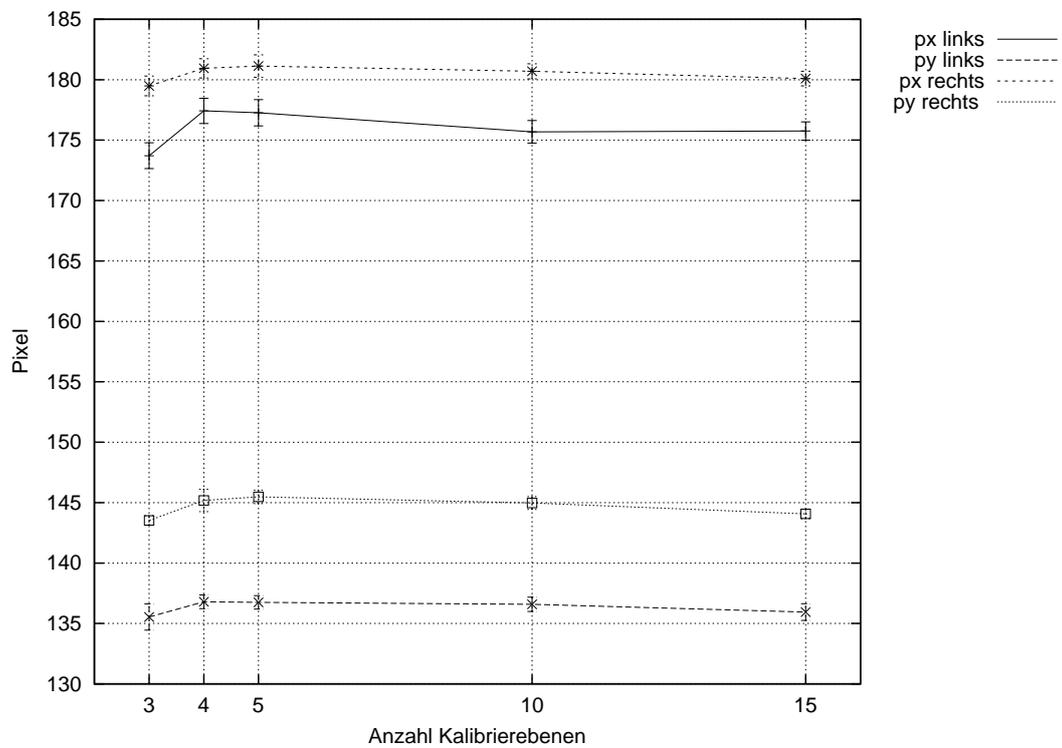


Abbildung 10.2: Anordnung der Kameras bei der Beispielkalibrierung.

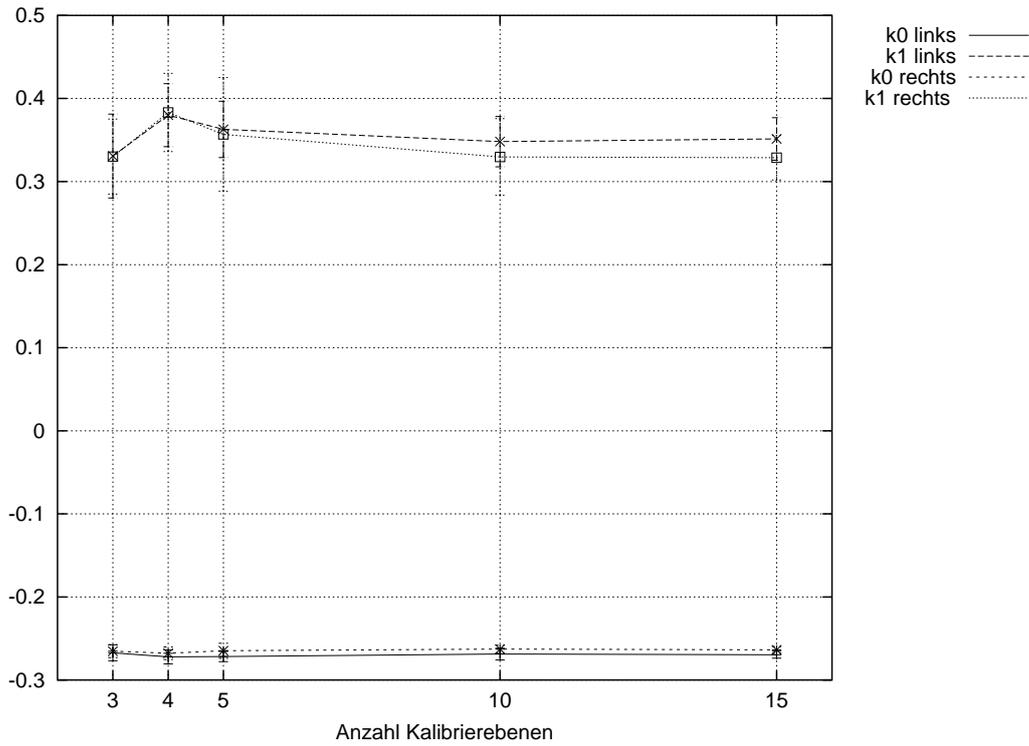


(a) Brennweiten.

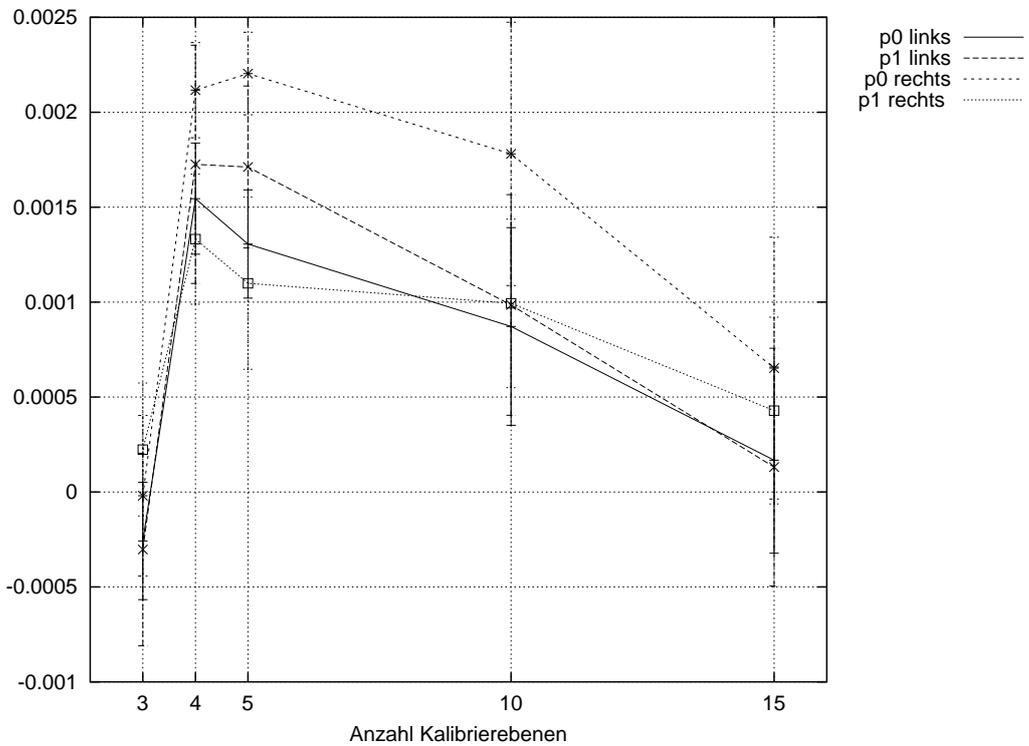


(b) Koordinaten des Hauptpunktes.

Abbildung 10.3: Auswirkung der Anzahl verwendeter Kalibrierebenen auf einzelne Parameter bei der Kamerakalibrierung.

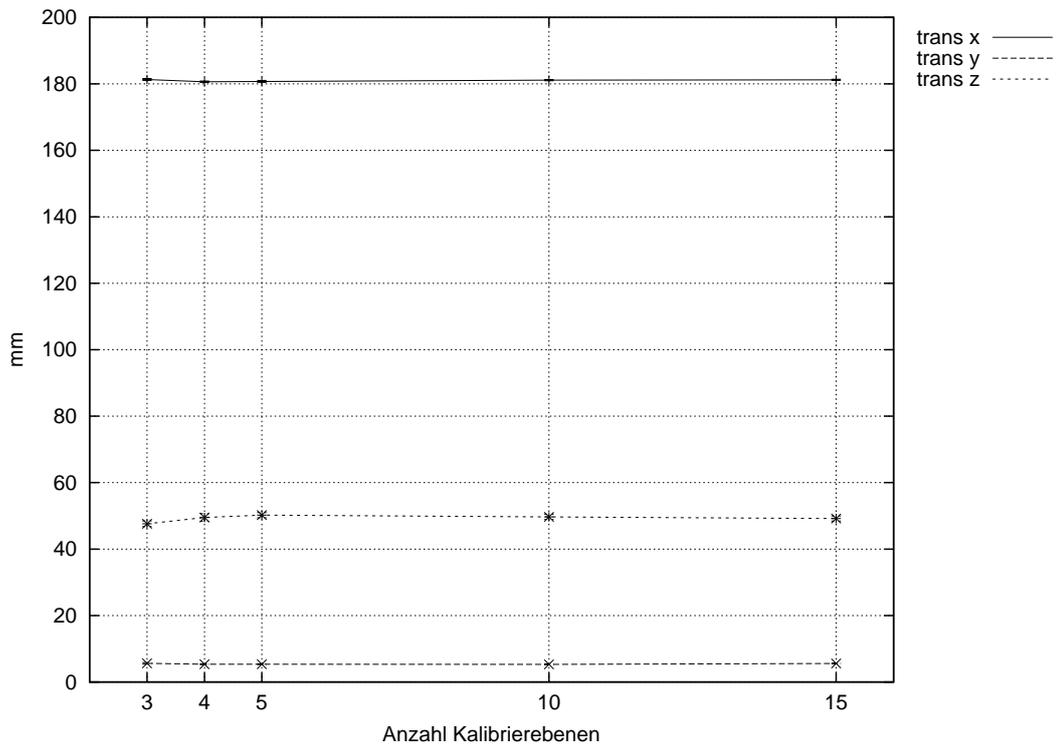


(a) Radiale Linsenverzerrung.

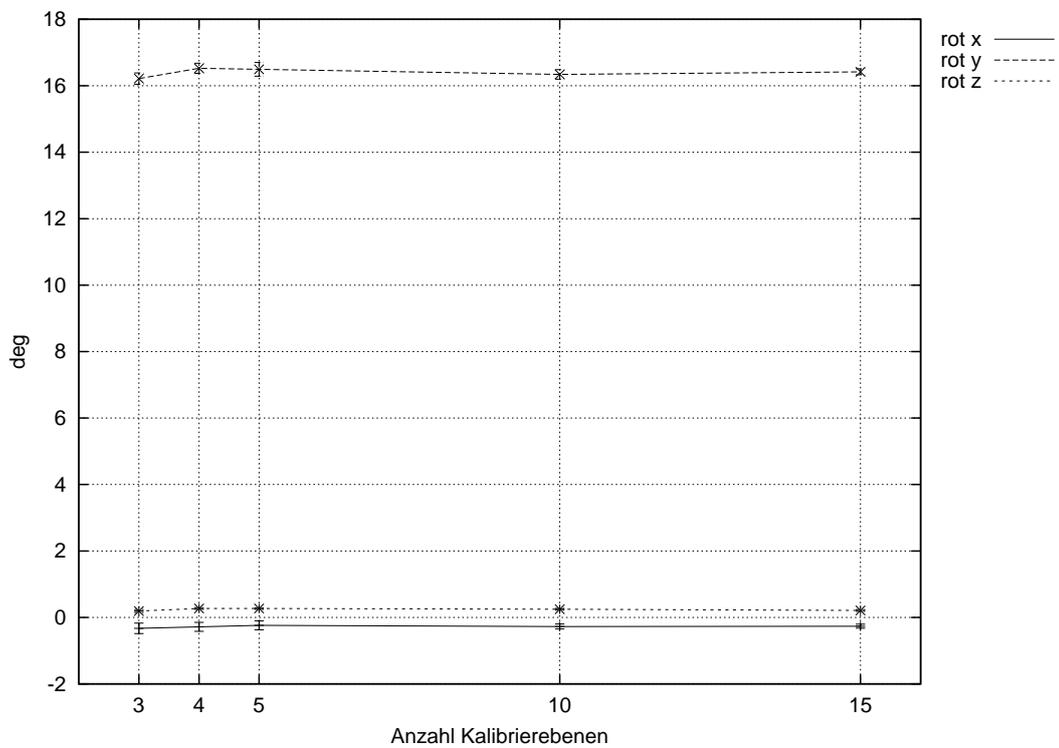


(b) Tangentiale Linsenverzerrung.

Abbildung 10.4: Auswirkung der Anzahl verwendeter Kalibrierebenen auf einzelne Parameter bei der Kamerakalibrierung.



(a) Translation zwischen den Kameras.



(b) Rotation zwischen den Kameras.

Abbildung 10.5: Auswirkung der Anzahl verwendeter Kalibrierebenen auf einzelne Parameter bei der Kamerakalibrierung.

# 11 Stereoalgorithmus

Die Beurteilung der Ergebnisse der Disparitätsberechnung aus Kapitel 8 gliedert sich in vier Teile. Zunächst stellt Abschnitt 11.1 kurz die rektifizierten Bilder vor, auf denen die Disparitätsberechnung arbeitet. Abschnitt 11.2 geht auf die berechneten Disparitäten an sich ein, 11.3 auf das zeitliche Verhalten bei variierenden Bildgrößen und Suchbereichen. Im vierten Teil dieses Kapitels werden das Laufzeitverhalten des Medianfilters und seine Auswirkung auf das Disparitätsbild untersucht.

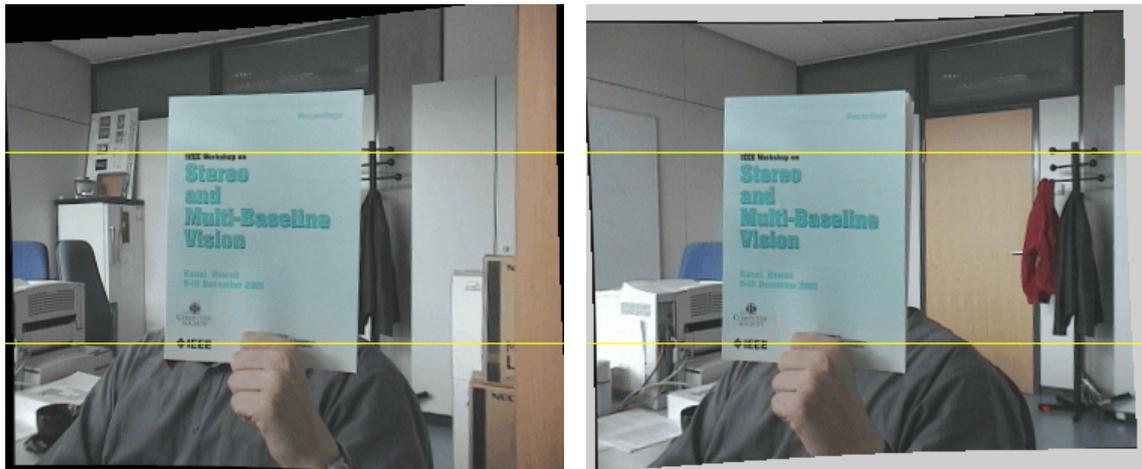
## 11.1 Bildentzerrung

Abbildung 11.1 zeigt das Ergebnis der Entzerrung mit der Lookup-Tabelle nach Algorithmus 7.1 (S. 72). Korrespondierende Bildpunkte liegen auf der selben Höhe in beiden Bildern, wie an den gelben Linien zu sehen ist. Die kissenförmigen Ränder zeigen, daß eine tonnenförmige Verzerrung der Originalbilder ausgeglichen wurde.

Wählte man identische Farben für die Bereiche außerhalb der Originalbilder, würde der anschließende Stereoalgorithmus sie mit hoher Wahrscheinlichkeit einander zuordnen, obwohl ihre Punkte nicht physikalisch miteinander korrespondieren. Der Zeitbedarf für die Entzerrung findet sich in Tabelle 12.1 (S. 121).

## 11.2 Disparitäten

Die Disparitäten, die der Algorithmus liefert, sind mit den Ergebnissen anderer Algorithmen vergleichbar, die die Summe der absoluten Differenzen mit Links-Rechts-Validierung einsetzen. Sie teilen sich damit auch alle Vor- und Nachteile. Vorteile



(a) Linkes Bild.

(b) Rechtes Bild.

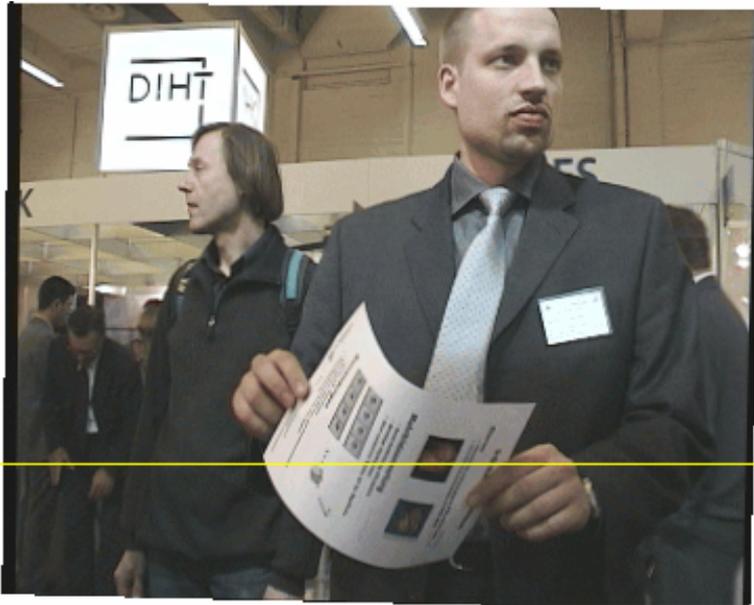
Abbildung 11.1: Entzerrtes Bildpaar.

sind die hohe Dichte der Disparitäten bei genügend guten Ausgangsbildern und die hohe Geschwindigkeit. Erkauft wird die Geschwindigkeit mit einigen Nachteilen:

- Als rein lokaler Ansatz wird keine Beziehung zwischen benachbarten Pixeln hergestellt. Es kommen einzelne Fehler in korrekt zugeordneter Umgebung vor, die zu einem gewissen Teil vom Medianfilter korrigiert werden können.
- Es kann keine Schätzung der Disparität für texturlose Flächen abgegeben werden.
- Die Subpixelinterpolation durch eine Parabel ist nur von begrenzter Genauigkeit [57].
- Objektkanten werden nicht exakt wiedergegeben, ihre Position weicht bis zu der halben Fenstergröße von der wirklichen ab [27, 52].

Der letzte Punkt ist verantwortlich für den größten Teil des Fehlers der Korrelationsmethode. Besser lokalisierte Kanten erhält man, indem man mit mehreren Fenstern pro Pixel arbeitet, deren Referenzpixel sich an unterschiedlichen Positionen innerhalb des Fensters befindet [18]. Leider resultiert diese Methode im Inneren der Objekte in Disparitätssprüngen und deutlich sichtbaren Artefakten [52], weshalb sie hier nicht eingesetzt wird.

Abbildung 11.2(a) zeigt das linke von zwei rektifizierten Bildern, für das die Disparitäten berechnet wurden. Die gelbe Linie markiert die Zeile des Bildes, für die die beiden  $x-d$ -Schnitte durch das Disparitätsvolumen abgebildet sind (vgl. Abb. 8.2,



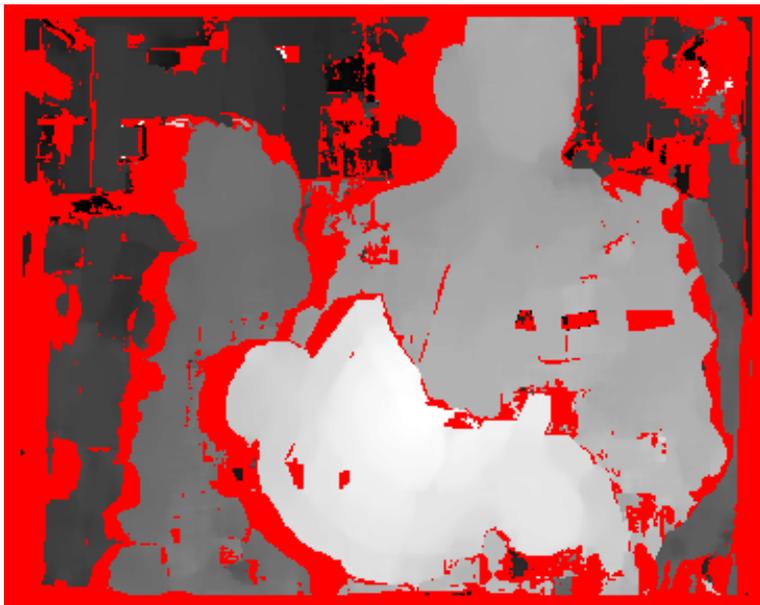
(a) Linkes Bild des Stereopaars.



(b)  $x$ - $d$ -Schicht des Disparitätsvolumens entlang der oben eingezeichneten Linie bei Fenstergröße  $1 \times 1$ .



(c) Dto. bei Fenstergröße  $13 \times 13$ .



(d) Disparität als Grauwert kodiert.

Abbildung 11.2: Ergebnis einer Disparitätsberechnung.

S. 77 und Abb. 8.7(a), S. 84). Je dunkler ein Pixel ist, desto geringer ist die SAD. Die Biegung des Blattes ist in Abbildung 11.2(c) deutlich besser zu erkennen als in 11.2(b). Die Pixel, an denen zum Beispiel die gelbe Linie die Schrift schneidet, wären ohne Aufsummierung zu einem Fenster nicht korrekt zuzuordnen gewesen. In der unteren der beiden Schichten ist zu sehen, daß sich die Kante vom linken Rand des weißen Blattes vor dem dunklen Hintergrund auf die Breite des Korrelationsfensters von 13 Pixeln aufweitet. Die Disparität aller Pixel zeigt Abbildung 11.2(d), sie ist umso größer, je heller das Pixel ist. Rot sind alle Pixel, die aufgrund der Eindeutigkeits-tests verworfen wurden.

### 11.3 Laufzeiten

Kapitel 8 hat sich in der Hauptsache darauf konzentriert, wie die Berechnung der Disparitäten so effizient wie möglich geschehen kann, da sie den größten Anteil an der Gesamtrechenzeit einnimmt. Die Geschwindigkeit verschiedener Implementierungen der Korrelation und das Verhalten bei variierenden Parametern wird hier deshalb ausführlicher untersucht als die Disparitäten im letzten Abschnitt.

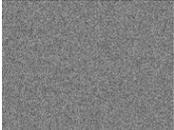
Name	dolphin	3D	pentagon	PAL	image0005 <sup>1</sup>
Größe	160×120	285×206	512×512	752×576	2048×1536
Disp.	±5 ··· ± 20	±5 ··· ± 25	±5 ··· ± 30	±5 ··· ± 50	±10 ··· ± 150
Bild					

Tabelle 11.1: Testbilder und in den Messungen verwendete Disparitätsbereiche.

Tabelle 11.1 enthält die Daten der fünf verwendeten Testbilder. Die Auflösung des kleinsten Bildes ist etwa für Webcam-Anwendungen oder Videokonferenzen typisch, das mit drei Megapixeln größte Bild repräsentiert die Größe heute üblicher Digitalkameraaufnahmen. Die drei anderen liegen in der Größe dazwischen, „pentagon“ ist ein oft benutztes Bildpaar zur Validierung von Stereo-Algorithmen<sup>2</sup>. Für die Messung der Rechenzeiten ist nicht von Bedeutung daß zwei der Bildpaare nur

<sup>1</sup>„Tea pot“-Sequenz <http://www.eecs.lehigh.edu/~tboult/STEREO/DataSets.htm>

<sup>2</sup>Carnegie Mellon Image Database <http://vasc.ri.cmu.edu/idb>

in Graustufen vorliegen, sie werden vor der Berechnung in dreikanalige Farbbilder umgewandelt.

Alle Bilder wurden mit den folgenden sieben Implementierungen durchgerechnet. Die erste Buchstabengruppe kennzeichnet, welches Speicherlayout verwendet wird (vgl. Abschnitt 8.4), 16 oder 32 ist die Anzahl der Bits pro Eintrag im Disparitätsvolumen, danach folgt ein Hinweis auf die Berechnung der SAD zweier einzelner Pixel. **C** bedeutet, daß die Differenzen der einzelnen Farbkanäle in normalem C berechnet und summiert wurden. Bei **psadbw** wird dafür der gleichnamige Befehl verwendet, der ab dem Pentium III und bei neueren Athlons existiert. Die beiden mit **MMX** bezeichneten Varianten nutzen zusätzlich zu **psadbw** noch MMX-Befehle zur Bildung der Fenstersummen.

**xyd 32 C** Speicherlayout wie in Abschnitt 8.4.1 beschrieben, 32 Bit pro Voxel, Berechnung der SAD mit C-Code, keine Begrenzung der Werte.

**xyd 32 MMX** Beschleunigung der SAD mit MMX-Befehlen (als Summe zweier saturierter Differenzen) und der Summierung in  $y$ -Richtung mit paralleler Addition jeweils zweier Werte.

**xdy 32 C** Speicherlayout nach 8.4.2, sonst wie **xyd 32 C**.

**xdy 32 psadbw** SAD-Berechnung mit dem **psadbw** Befehl. Linkes Pixel während der innersten Schleife in einem eigenen MMX-Register.

**dxy 32 psadbw** Wie vorher, aber Speicherlayout aus Abschnitt 8.4.3.

**dxy 16 C** Begrenzung der SAD-Werte, so daß 16 Bit zur Darstellung der Fenstersumme ausreichen. Reine C-Variante, die vom 386er aufwärts läuft, vor allem interessant für Anwendungen in „embedded“-Systemen.

**dxy 16 MMX** Zweimal paralleles Einlesen von je zwei Pixeln des rechten Bildes, parallele Weiterverarbeitung der vier SAD-Werte bei der Berechnung von links nach rechts.

Die Programme wurden unter Linux ausgeführt und mit dem GNU<sup>3</sup> C++-Compiler der Version 2.95.3 erstellt. Die Optimierer-Flags waren: `-O6 -funroll-loops -fschedule-insns2 -march=i686 -mcpu=pentiumpro`. „Loop-unrolling“ bedeutet, den Schleifenrumpf von kleinen Schleifen mehrmals hintereinanderzusetzen, um den relativen Anteil der Zähl- und Sprungbefehle zu verringern. Vor allem kleine Schleifen können davon zum Teil erheblich profitieren. Das „instruction-scheduling“ sorgt dafür, daß die Befehle so umsortiert werden, daß zum Beispiel Wartezeiten beim Laden von Werten aus dem Speicher zur Ausführung von anderen Befehlen genutzt

<sup>3</sup><http://www.gnu.org>

werden. Auf Architekturen mit wenig Registern, wie etwa IA32, ist die zweite Variante gegenüber `-fschedule-insns` vorzuziehen.

Die Laufzeit der einzelnen Programme hängt natürlich nicht nur von der Leistung des Prozessors ab, sondern wird von den Komponenten Cache, Chipsatz und Hauptspeicher beeinflusst. Um zu zeigen, daß sich die Implementierungen auf verschiedenen Rechnern durchaus unterschiedlich verhalten, wurden die Tests auf diesen beiden Systemen durchgeführt:

- Pentium III 800 MHz, 512 MB RDRAM 800, Intel i845 Chipsatz
- Athlon 1,3 GHz, 512 MB PC133 SDRAM (3-2-2), VIA KT-133 Chipsatz

Das Pentiumsystem ist ein Dual-Board, die Benchmarks nutzen aber nur einen Prozessor. Durch einfaches Aufteilen des Bildes in obere und untere Hälfte kann die Berechnung je nach Bildgröße um den Faktor 1,6 bis 1,8 beschleunigt werden. Da Mehrprozessorsysteme vor allem als Arbeitsstation nicht weit verbreitet sind, wurde diese Möglichkeit der Geschwindigkeitssteigerung nicht mit in die Benchmarks aufgenommen. Die limitierende Größe beim Zweiprocessorbetrieb ist die Transferrate zum Hauptspeicher, vor allem bei größeren Bildern wirkt sich die Verwendung von RDRAM positiv aus.

Abbildung 11.3 und 11.4 zeigen die Laufzeit der einzelnen Implementierungen für die fünf Testbilder, aufgetragen über die Anzahl durchsuchter Disparitäten. Die Ergebnisse beider Systeme für ein Bild stehen jeweils nebeneinander, zur besseren Vergleichbarkeit sind die Ordinaten identisch skaliert. Jede Rechnung wurde zehnmal durchgeführt. Die durchgezogenen Kurven verbinden die Mittelwerte, die Standardabweichung wird durch Fehlerbalken gekennzeichnet.

Erwartungsgemäß schneiden die beiden xyd-Varianten am schlechtesten ab, lediglich die stark optimierte MMX-Version liefert beim „PAL“-Bild überraschend konkurrenzfähige Ergebnisse. Für die auffallenden Ausreißer ist die Minimumsuche verantwortlich, der genaue Grund konnte leider nicht ermittelt werden, dazu müßten die prozessorinternen Performanceregister gezielt eingesetzt und ausgewertet werden. Weil diese Implementierungen nur untersucht werden um zu zeigen, daß sie den anderen unterlegen sind, ist dies unterblieben. Das größte Bild kann mit ihnen nicht berechnet werden (Abb. 11.4(g) und 11.4(h)), da das komplette Disparitätsvolumen gespeichert werden muß. Die dazu notwendigen 3,6 GB übersteigen den verfügbaren Hauptspeicher bei weitem.

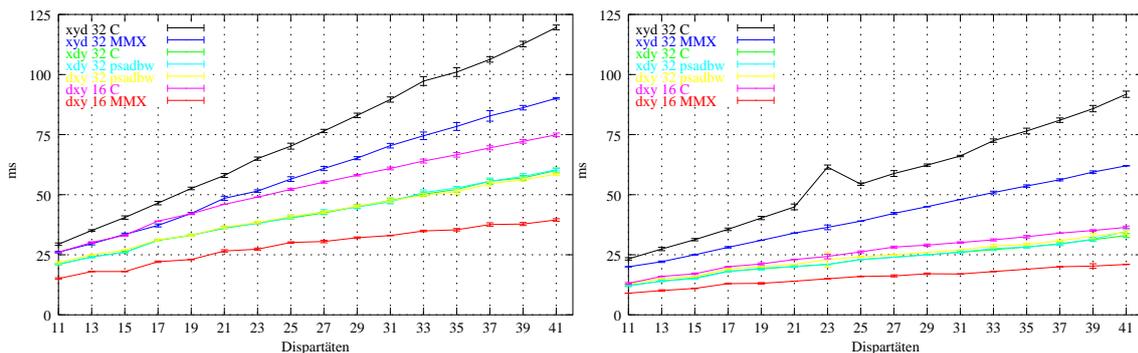
Die beiden Programme, die das xdy-Layout nutzen (Grün und Türkis), liegen bei den Messungen im Mittelfeld. Der `psadbw` Befehl bringt beim Pentium nur einen sehr geringen Vorteil, beim Athlon sind die Kurven fast ununterscheidbar. Offensichtlich stellen die Differenzbildung und Summierung der drei Farbkanäle keinen besonders

großen Anteil an der Rechenzeit, der Algorithmus ist in der Hauptsache durch die Speicherbandbreite begrenzt.

Je größer das Bild ist, desto deutlicher ist der Vorteil des dxy-Speicherlayouts. Besonders deutlich fällt dies in Bild 11.4(h) auf, aber auch beim Pentium zeigt der Vergleich der Bilder 11.3(a) bis 11.4(g), daß sich die gelbe Linie mit zunehmender Bildgröße und größerem Suchbereich immer weiter von der grünen und türkisfarbenen entfernt.

Das Verhalten der Variante „dxy 16 C“ (Magenta) unterscheidet sich zwischen den beiden Rechnern am deutlichsten. Sie schneidet beim Pentium sogar schlechter ab als „xdy 32 C“, obwohl sich die Größe des Disparitätsvolumens halbiert hat. Beim Athlon ist sie dagegen durchgehend schneller. Dies legt den Schluß nahe, daß der Pentium deutlich unwilliger auf kleine (<32 Bit) Speicherzugriffe reagiert als der Athlon.

Am besten werden die Anforderungen für einen hohen Speicherdurchsatz und damit eine hohe Geschwindigkeit von der letzten Implementierung („dxy 16 MMX“) erfüllt. Sie liest die Pixeldaten immer ausgerichtet an einer durch acht teilbaren Adresse, zwei Pixel auf einmal. Die vier Pixel zweier solcher Zugriffe werden zusammen weiterverarbeitet, insbesondere die Summierung der Fenster kann sowohl in  $x$ - als auch in  $y$ -Richtung vierfach parallel erfolgen. Nach jedem Durchlauf der innersten Schleife über die Disparität muß der Prozessor mit einem `emms` Befehl wieder in den Floating-Point Modus zurückgeschaltet werden, um die Subpixelinterpolation durchzuführen. Würden die Fließkommaoperationen in der SSE-Einheit mit einfacher Genauigkeit durchgeführt, könnte der MMX Modus beibehalten werden



(a) „dolphin“, Pentium III 800 MHz

(b) „dolphin“, Athlon 1.3 GHz

Abbildung 11.3: Ergebnis der Laufzeitmessungen am kleinsten Testbild mit zwei verschiedenen Prozessoren.

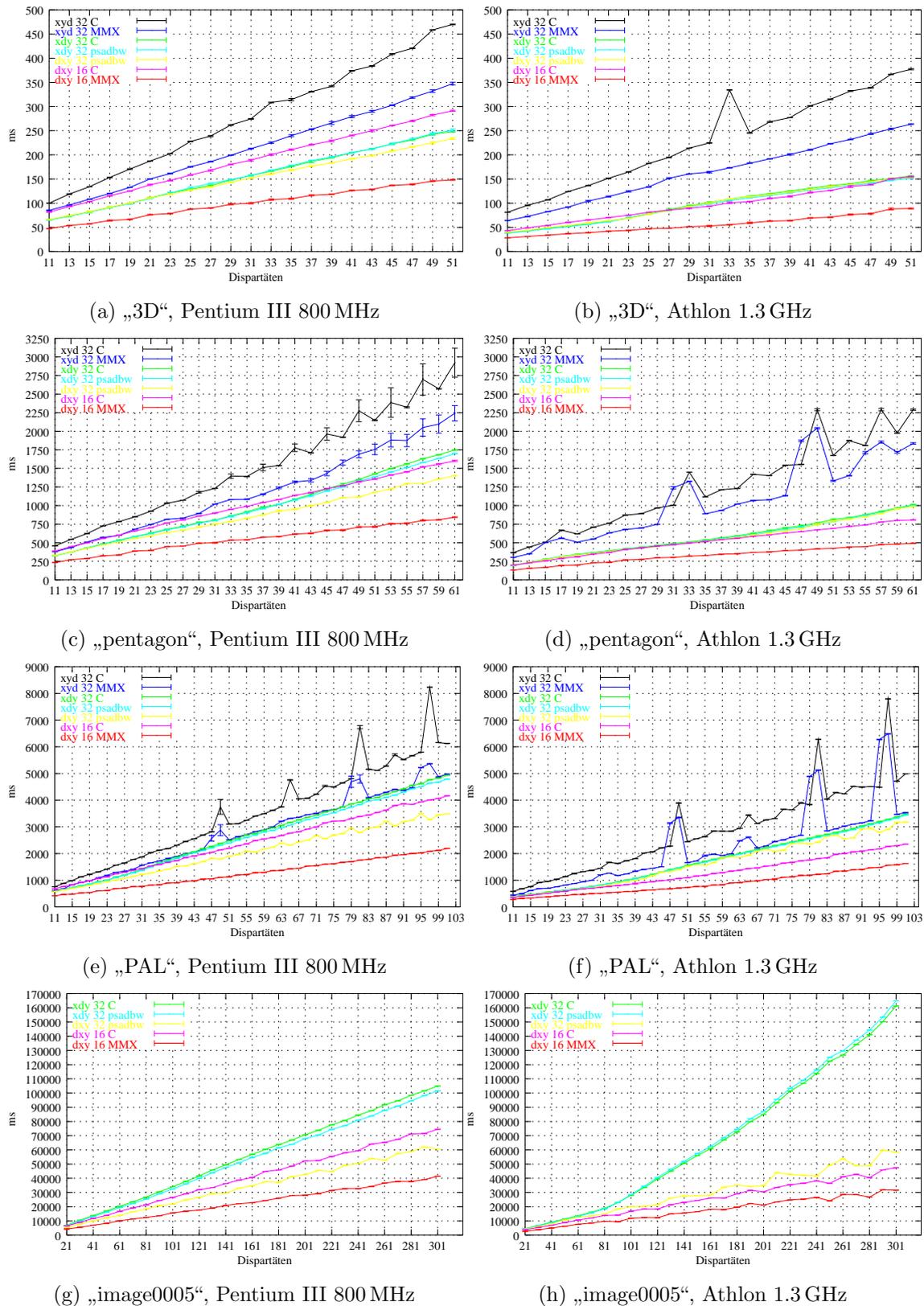


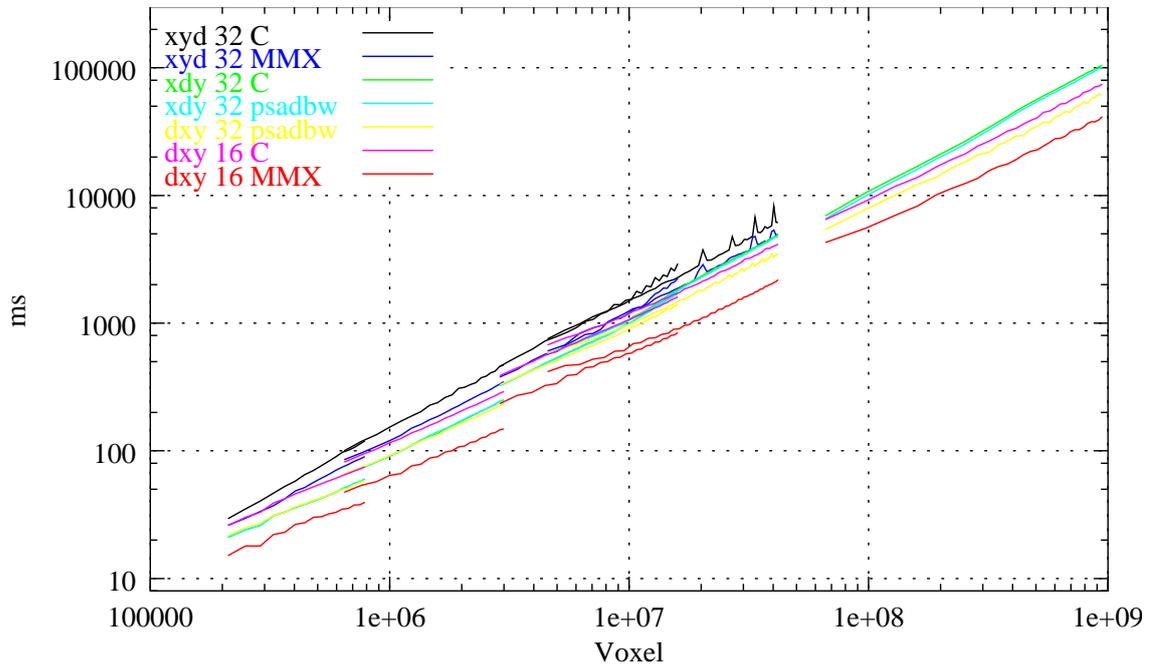
Abbildung 11.4: Laufzeitmessungen für die restlichen vier Bilder.

und die Geschwindigkeit weiter gesteigert werden. Leider konnte diese Verbesserung noch nicht implementiert und getestet werden.

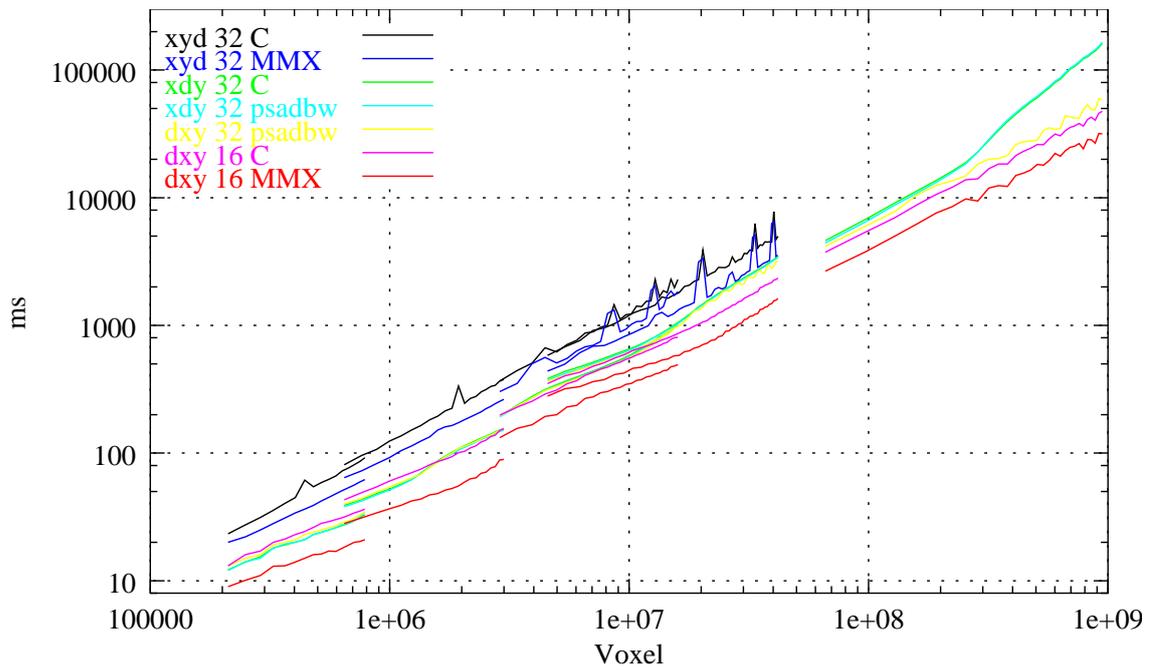
Abbildung 11.5 faßt alle Diagramme der Abbildungen 11.3 und 11.4 zusammen. Die Laufzeit ist hier nicht über der maximalen Disparität aufgetragen, sondern über der Größe des Disparitätsvolumens. Aufgrund des großen Wertebereiches beider Achsen ist ein doppelt logarithmischer Maßstab gewählt worden. Es ist zu erkennen, daß bei identischer Anzahl von Voxeln aber kleinerem Bild (zwei übereinander liegende Kurven gleicher Farbe) weniger Zeit benötigt wird, was darauf schließen läßt, daß die Operationen pro Pixel einen merklichen Anteil an der Gesamtrechnzeit haben. Der Unterschied ist größer bei den schnelleren Implementierungen, weil die Operationen pro Voxel optimiert wurden, die pro Pixel, wie zum Beispiel die Subpixelinterpolation, aber gleich geblieben sind.

Die konstante Steigung der Diagramme zeigt, daß die Algorithmen (mit Ausnahme der beiden Ausreißer beim Athlon im rechten oberen Teil von Abb. 11.5(b)) über den gesamten betrachteten Bereich von fünf Größenordnungen eine konstante Leistung liefern. Die schließlich im Gesamtsystem eingesetzte Version „dxy 16 MMX“ benötigt durchgehend  $50\text{--}70\text{ ns/Voxel}$  auf dem Pentium mit 800 MHz und  $30\text{--}40\text{ ns/Voxel}$  auf dem Athlon 1,3 GHz.

Abbildung 11.6 zeigt das Verhältnis der Laufzeiten von Pentium und Athlon. Die gestrichelt eingezeichnete Linie markiert das Taktverhältnis der beiden CPUs von  $1300/800$ . Bei kleinen Bildern und optimiertem Speicherlayout liegt das Athlonsystem klar vorne, und zwar um mehr als diesen Faktor. Je größer allerdings das Speichertransfervolumen wird, desto mehr schrumpft der Vorteil gegenüber dem Pentium. Sowohl bei den Algorithmen mit nicht optimaler Speicheranordnung als auch beim Verlauf der Kurven der beiden großen Bilder ist dies zu erkennen. Der Prozessorkern des Athlon ist leistungsfähiger, dafür ist das Pentiumsystem insgesamt ausgewogener, was in der Hauptsache dem (deutlich teureren) RDRAM Speicher und dem Chipsatz zuzuschreiben ist.



(a) Pentium III 800 MHz



(b) Athlon 1.3 GHz

Abbildung 11.5: Vergleich der Laufzeiten aller Algorithmen bei allen Bildern.

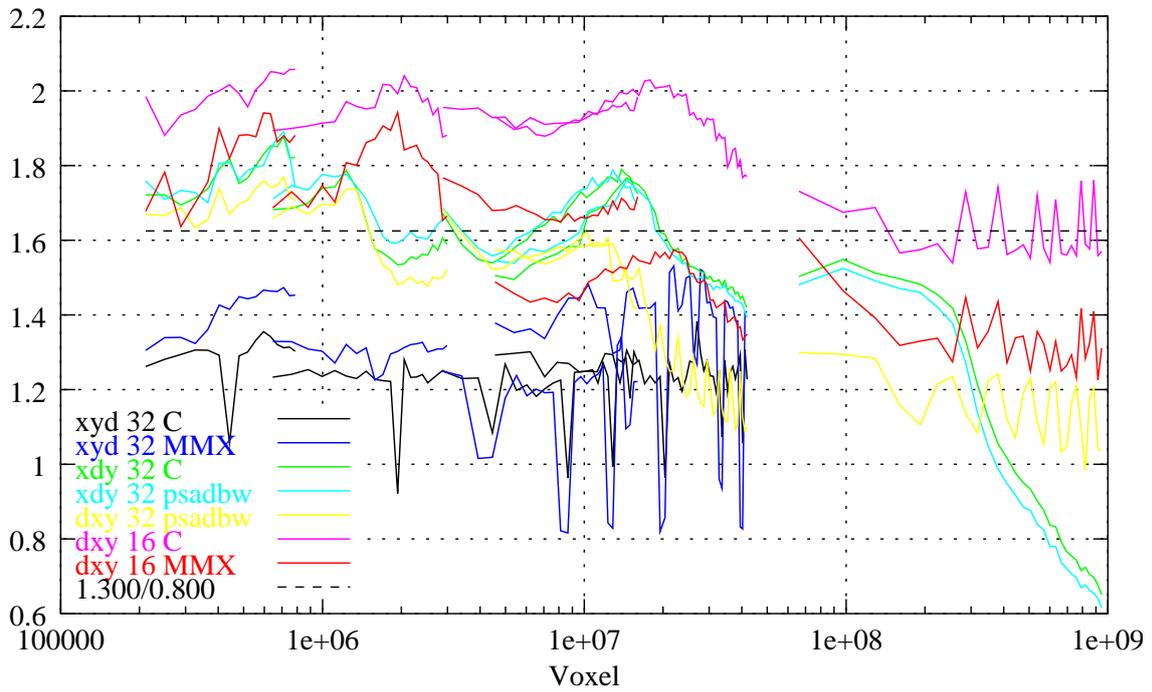


Abbildung 11.6: Quotient der Laufzeit von Pentium und Athlon bei gleichem Test.

## 11.4 Medianfilter

Abbildung 11.7 vergleicht die Rechenzeit dreier Implementierungen des Medianfilters der Größe  $3 \times 3$  Pixel miteinander. Die erste benutzt die Routine `qsort()` aus der Standard-Laufzeitbibliothek, die beiden anderen sortieren die neun Disparitäten nur so weit, bis das Element in der Mitte gefunden ist. Die zweite Methode verwendet die Sortierroutine `select2()` aus [56], die den optimalen Ansatz für flexible Arraygrößen darstellt. Die dritte Methode schließlich implementiert das in Abbildung 8.10 (S. 88) vorgestellte optimale Sortiernetzwerk für eine feste Anzahl von neun Elementen.

Der `qsort()` Algorithmus ist unter anderem deshalb deutlich abgeschlagen, weil er für jeden Vergleich zweier Zahlen einen Funktionsaufruf benötigt. Wird ein Medianfilter mit einer größeren Maske gewünscht, kann er leicht mit `select2()` realisiert werden, bei der kleinen Maske ist die dritte Routine allerdings etwa eineinhalb mal so schnell. Die Messungen wurden auf dem Pentium 800 MHz mit den Testbildern aus dem vorherigen Abschnitt durchgeführt.

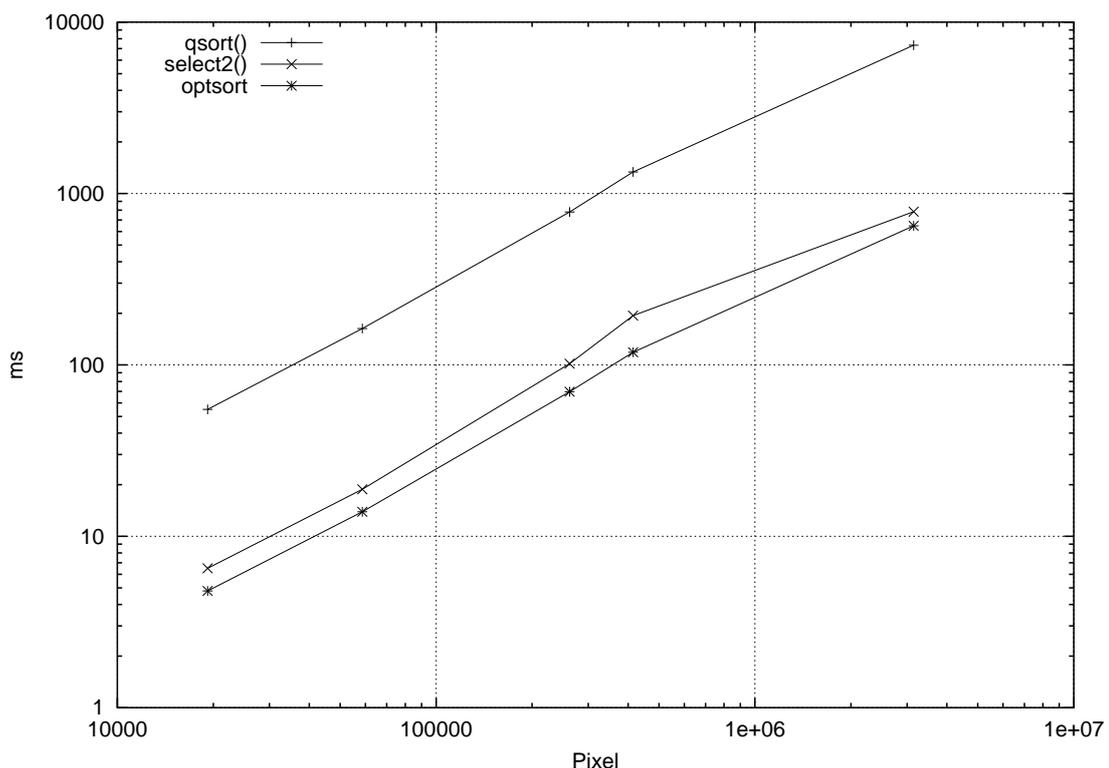
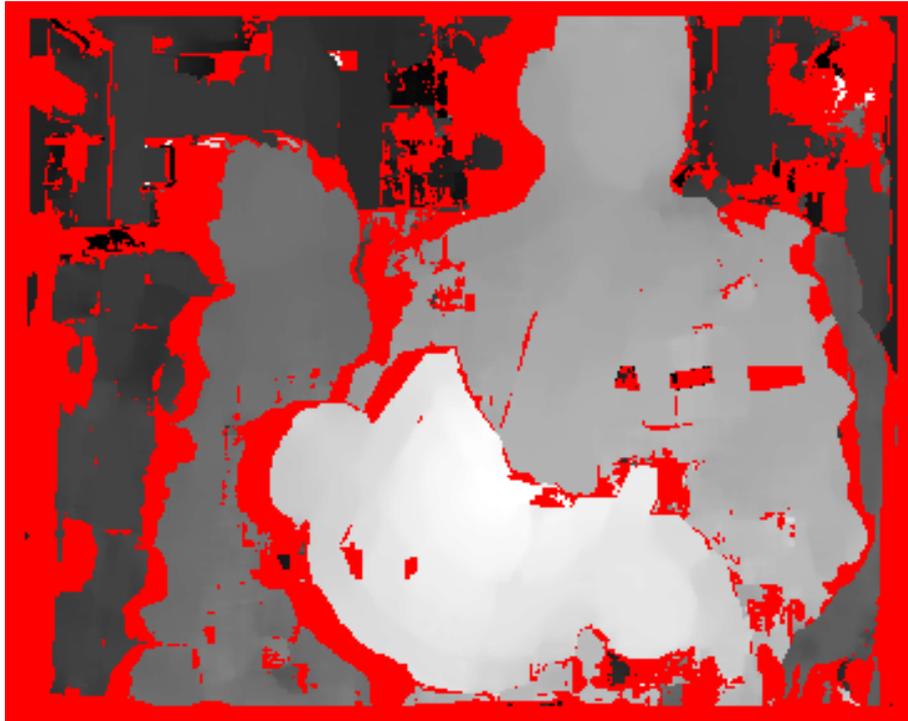


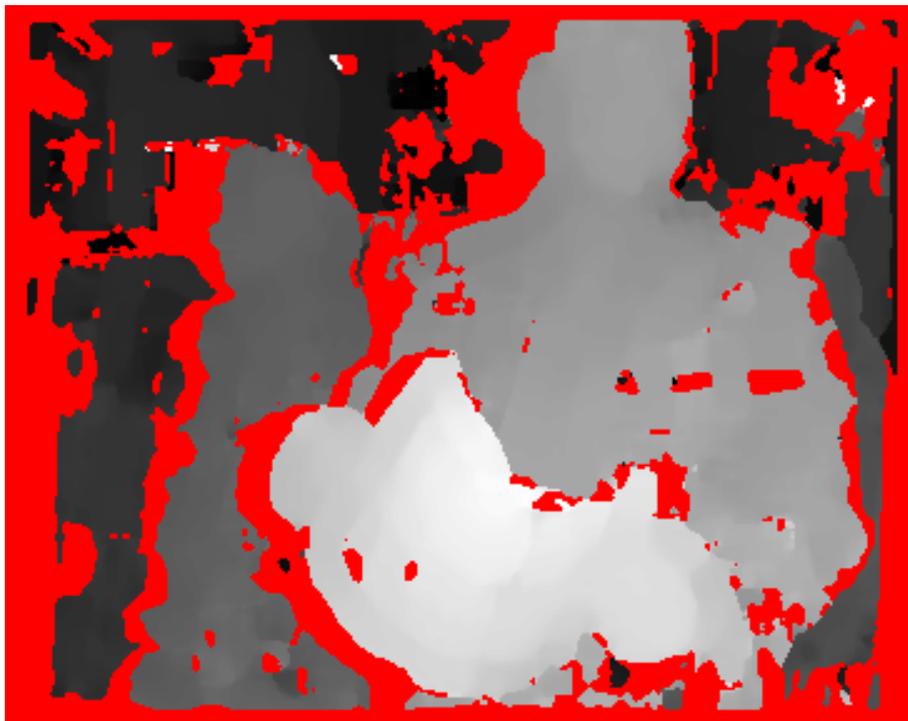
Abbildung 11.7: Vergleich der Ausführungsgeschwindigkeit dreier Implementierungen des Medianfilters.

Abbildung 11.8 zeigt ein Disparitätsbild vor und nach der Anwendung des Medianfilters. Kleine Lücken in Bild 11.8(a), sind geschlossen worden. Fehlerhafte Pixel, etwa im Bereich des Namensschildes (vgl. Abb. 11.2(a)), konnten deutlich reduziert werden. Befinden sich mehr als fünf falsche Pixel innerhalb der 9 Pixel der Filtermaske, können sie natürlich nicht erkannt werden.

Vor allem für die Darstellung als 3D-Modell bringt der Medianfilter Vorteile, da dort sowohl kleine Löcher als auch falsch zugeordnete Pixel viel stärker auffallen als im Disparitätsbild.



(a) Ergebnis direkt nach der Minimumssuche.



(b) Ergebnis nach der Medianfilterung.

Abbildung 11.8: Medianfilterung der Disparitätskarte.

# 12 Gesamtsystem

Nachdem alle Teile des Rekonstruktionsalgorithmus vorgestellt wurden, soll nun ihr Zusammenspiel untersucht werden. Im Rahmen dieser Arbeit wurde ein C++ Programm erstellt, das alle Komponenten vereint und kontinuierlich eine Rekonstruktion der von der Stereokamera aufgenommenen Szene berechnet und darstellt. Durch die Verwendung von Threads werden, anders als bei den Benchmarks, beide CPUs des Pentium-Systems genutzt. Am einfachsten läßt sich die Rektifikation parallelisieren, jedes der beiden Bilder wird von einem eigenen Thread bearbeitet. Bei der Disparitätsberechnung und der Triangulierung teilen sich zwei Threads jeweils die obere und untere Hälfte der Bilder beziehungsweise des Modells. Abbildung 12.1 skizziert den Ablauf des Programmes und das Zusammenspiel der einzelnen Threads.

An den mit Nummern gekennzeichneten Stellen kann eine Zeitmessung erfolgen, die Ergebnisse einer Beispielmessung zeigen Tabelle 12.1 und Abbildung 12.2. Die Übertragung eines Halbbildes signalisiert der PLX-Chip (s. Abb. 5.8, S. 41) mit einem Interrupt. Die oberste Zeile von Abbildung 12.1 zeigt die regelmäßig auftretenden Interrupts der beiden DMA-Übertragungen. Wenn gerade keine Berechnung läuft, wird die Blockierung des entsprechenden Rektifikations-Threads aufgehoben und er läuft los (Punkt ① und ②). Andernfalls wird der Interrupt ignoriert, im Bild gestrichelt angedeutet. Haben beide Threads ihre Arbeit erledigt, blockieren sie wieder und die Disparitätsberechnung beginnt ③. Nach deren Beendigung ④ wird das Ergebnis mediangefiltert ⑤ und in ein 3D-Dreiecksnetz umgewandelt ⑥, das schließlich zur Grafikkarte übertragen wird ⑦.

Sowohl OpenGL als auch die eingesetzte Qt-Bibliothek<sup>1</sup> erfordern, daß Zeichenoperationen im Haupt-Thread erfolgen. Wenn ein anderer Thread eine solche Ope-

---

<sup>1</sup><http://www.trolltech.com>

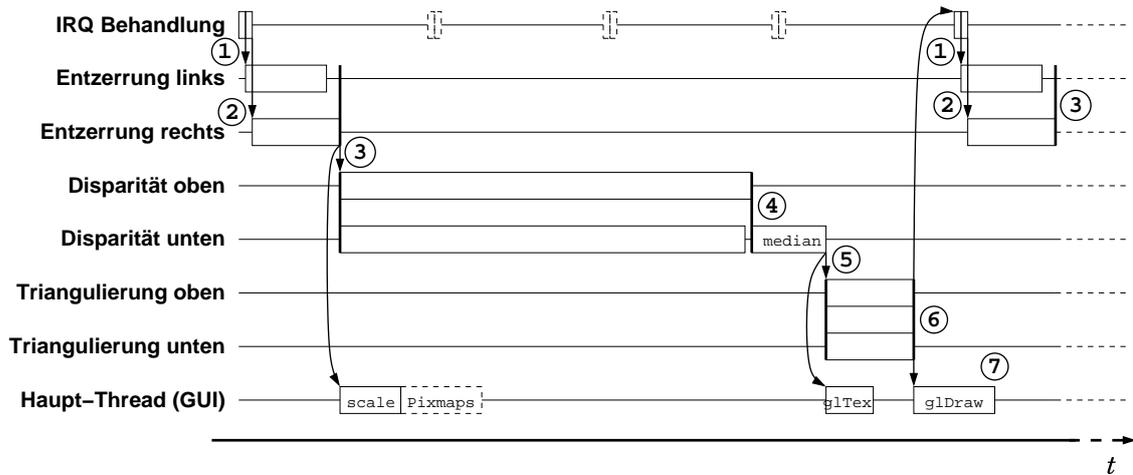


Abbildung 12.1: Threads des Rekonstruktionsprogramms.

ration anstößt, geschieht dies asynchron durch ein Event, das dem GUI-Thread über die Eventloop zugestellt und von ihm ausgeführt wird. Die folgenden drei Fälle nutzen explizit oder intern nicht reentrante Routinen und fallen daher in die gerade beschriebene Kategorie.

- Das rektifizierte linke Bild wird so skaliert, daß seine Seitenlängen eine Potenz von zwei sind, damit es als Textur für OpenGL verwendet werden kann [53] (im Bild angedeutet mit `scale`).
- Ist eine Anzeige der Kamerabilder gewünscht, müssen sie in das Displayformat umgewandelt werden (`Pixmap`).
- Die Übertragung der Textur (`glTex`) findet gleichzeitig mit der Modellgenerierung statt. Wird das Modell mit der Maus bewegt, kann zwischen den Zeitpunkten ⑤ und ⑥ kurzzeitig die neue Textur auf den alten Geometriedaten angezeigt werden. Fänden die Übertragungen von Textur und Geometrie (`glDraw`) zum Zeitpunkt ⑥ nacheinander statt, wäre eine Reaktion auf Mausbewegungen in dieser Zeit nicht möglich und das Modell bewegte sich ruckartig. Die Textur bereits nach der Rektifikation und Skalierung zu übertragen, würde das Zeitfenster unnötig vergrößern, in dem dieser Effekt auftreten kann.

Der Unterschied von  $6 \mu s$  zwischen den Zeitpunkten des linken und rechten Bildes in Tabelle 12.1 liegt am oberen Ende dessen, was gemessen wurde. Die Differenz rührt zum einen daher, daß zuerst der Interrupt des linken Bildes komplett abgearbeitet wird. Dazu gehören Kontextwechsel zwischen Kernel- und Userspace und

Nr.	t(ms)	$\Delta t$ (ms)	Ereignis
①	26.33		Übertragung des linken Bildes
②	26.39	0.06	Übertragung des rechten Bildes
③	38.21	11.82	Bildentzerrung
④	233.25	195.04	Disparitätsberechnung
⑤	267.60	34.35	Medianfilter
⑥	279.00	11.40	Triangulierung
⑦	290.41	11.41	Übertragung an die Grafikarte
①	306.32	279.99	Nächstes Bild ( $\Delta t$ zum letzten Bild)

Tabelle 12.1: Messung der Zeitpunkte aus Abbildung 12.1.

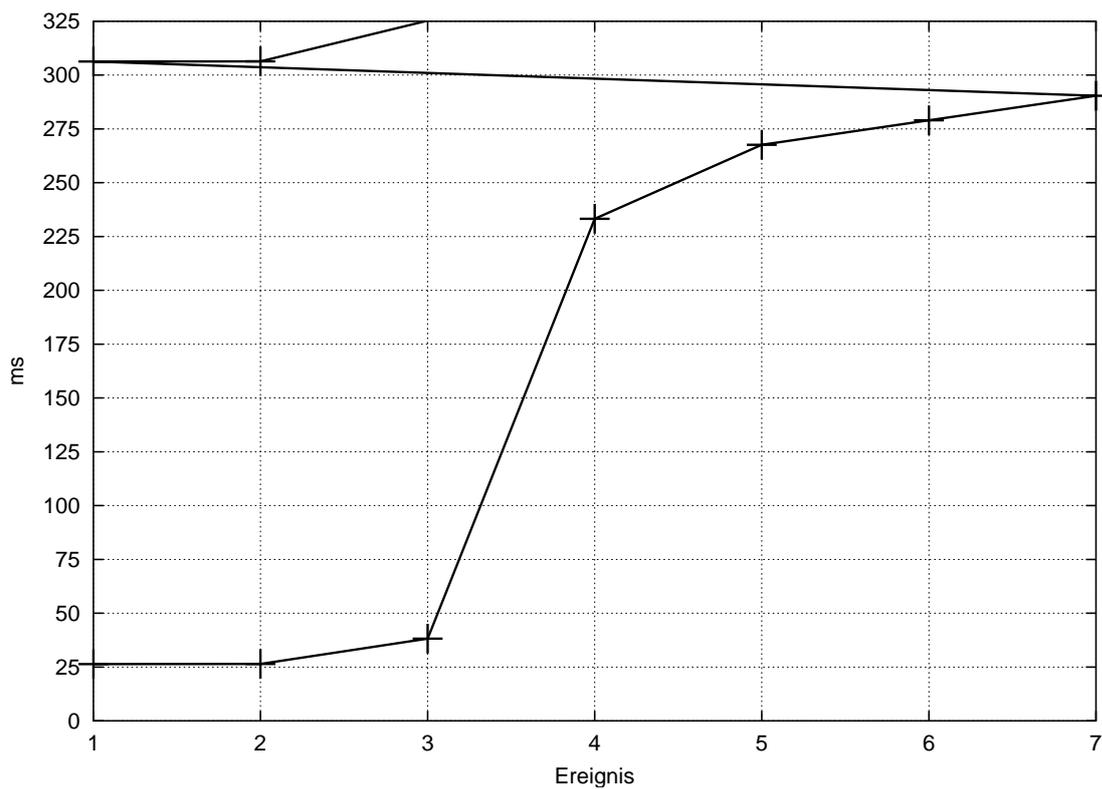


Abbildung 12.2: Zeitlicher Ablauf der Rekonstruktion.

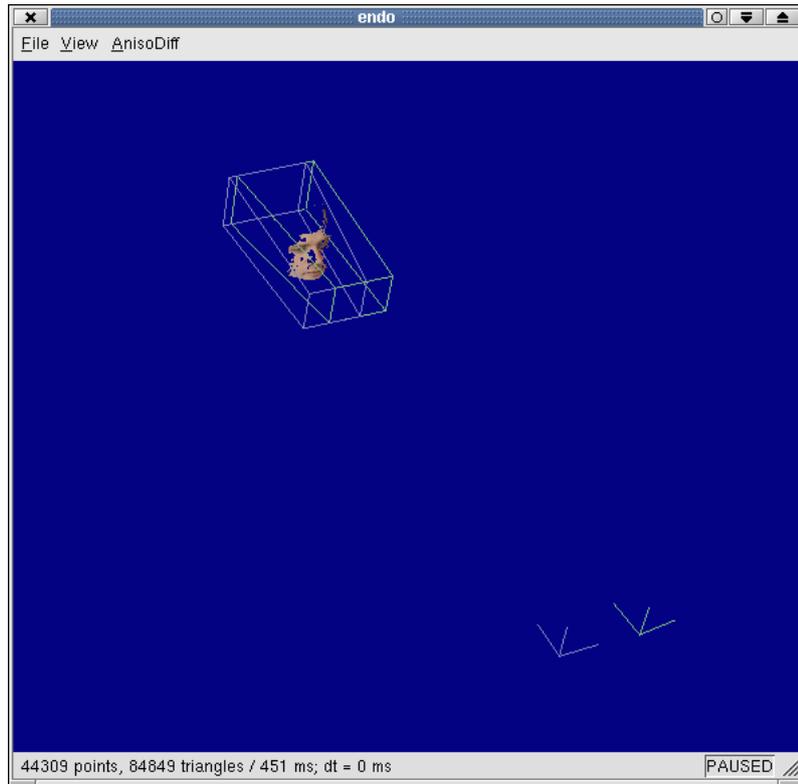
umgekehrt, das Aufwecken des Rektifikations-Threads und das Verwalten des internen Ringpuffers für die Bilddaten. Zum anderen ist die Übertragung auf dem PCI-Bus nicht immer exakt gleich schnell und das linke Bild hat einen geringen Vorteil, da es zuerst gestartet wird.

In diesem Beispiel waren die rektifizierten Bilder etwa  $360 \times 300$  Pixel groß, der Disparitätsbereich betrug 32 Pixel. Die Zeiten für die Triangulation und Übertragung hängen von der Anzahl der gefundenen Disparitäten ab. Mit 95000 Dreiecken gehört das Beispiel zu den dichteren Rekonstruktionen. Die eingesetzte Grafikkarte GeForce2/GTS von NVIDIA<sup>2</sup> hat also über 8 Millionen Dreiecke pro Sekunde verarbeitet. Dem theoretischen Spitzenwert von 25 Millionen Dreiecken pro Sekunde könnte man noch etwas näherkommen, wenn statt einer Liste von Dreiecken sogenannte „triangle-strips“ verwendet würden, deren Verwaltung und Erzeugung aber bei einer dynamischen Geometrie den Vorteil für die Anzeige leicht wieder zunichte macht. Wie Bild 12.2 zeigt, hat die Disparitätsberechnung den bei weitem größten Anteil an der Rechenzeit. Erst wenn sie noch weiter optimiert wurde, lohnt sich eine Betrachtung der anderen Teile wieder.

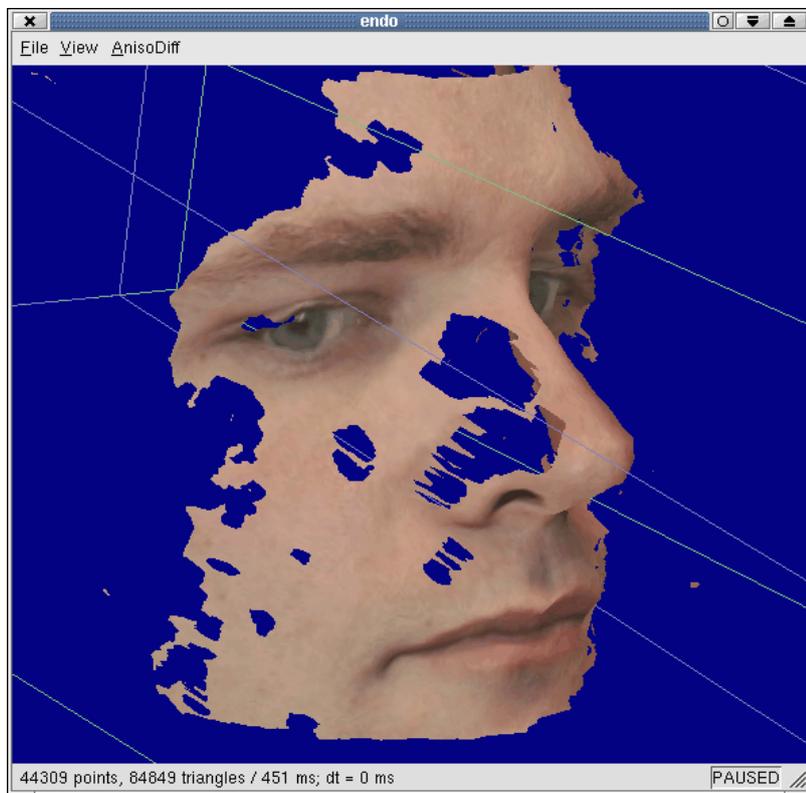
Abbildung 12.3 zeigt das Rekonstruktionsprogramm und zwei Ansichten einer Beispielrekonstruktion eines Gesichtes aus etwa 2,5 m Abstand. In Abbildung 12.3(a) sind rechts unten die lokalen Koordinatensysteme der beiden Kameras zu sehen, links darüber die beiden Sichtpyramiden der Kameras. Die vordere und hintere Begrenzung wird durch den Tiefen- beziehungsweise Disparitätssuchbereich bestimmt. Das rekonstruierte Modell befindet sich im Schnittvolumen der beiden Pyramiden. Die Darstellung der Kamerakoordinatensysteme und der Sichtpyramiden erleichtert die Orientierung und den Umgang mit dem Programm erheblich. Abbildung 12.3(b) zeigt das Modell alleine, es besteht aus 44309 Punkten und 84849 Dreiecken. Aufgrund des großen Disparitätssuchbereichs, in Abbildung 12.3(a) zu erkennen an der Länge der Pyramidenstümpfe und der vorne und hinten stark unterschiedlichen horizontalen Überlappung zwischen ihnen, liegen hier 451 ms zwischen zwei Aufnahmezeitpunkten ①.

---

<sup>2</sup><http://www.nvidia.com>



(a) Sichtpyramiden der Kameras und Kamerapositionen.



(b) Modell.

Abbildung 12.3: Beispielrekonstruktion.



# 13

## Diskussion, Ausblick und Zusammenfassung

### 13.1 Diskussion

Mit der FPGA-Karte microEnable steht die Möglichkeit zur Verfügung, zwei volle PAL-Datenströme in Farbe synchron zu digitalisieren, was über die Möglichkeiten kommerzieller Framegrabber hinausgeht. Die Übertragung der Daten mit Scatter-Gather-DMA direkt in den Speicher des Anwendungsprogramms benötigt praktisch keine Leistung der CPU, erst in jüngster Zeit setzen herkömmliche Framegrabber ebenfalls diese Technik ein. Durch Tausch des Video A/D-Wandlers gegen LVDS-Treiber und eine angepasste Konfiguration des FPGAs können auch Digitalkameras mit diesem inzwischen recht weit verbreiteten Interface angeschlossen werden.

Keines der auf dem Markt befindlichen Stereo-Systeme war mit Kameras ausgestattet, die die Rekonstruktion des Kopfes im MEG-Raum mit ausreichender Genauigkeit erlaubt hätten. Dies und die Tatsache, daß die Kameras eng an die Systeme gekoppelt und nicht dazu gedacht sind, vom Anwender kalibriert zu werden, machen deutlich, daß eine alternative Lösung gesucht werden mußte. Mit den gewählten Kameras läßt sich darüber hinaus ein weit größeres Spektrum von Anwendungen abdecken. In Weitwinkelstellung besitzen sie zur Erfassung eines großen Bereiches einen ebenso großen Öffnungswinkel wie die Fertigsysteme, durch Vergrößerung der Brennweite können sie aber zusätzlich Messungen aus großem Abstand durchführen. Desweiteren macht die Verfügbarkeit eines analogen Eingangs erst den Anschluß eines Stereo-Endoskops möglich.

Aufgrund der variablen Brennweite der Kameras und der Möglichkeit, den Winkel zwischen ihnen zu verändern, ist eine einfach zu handhabende und robuste Kalibrie-

nung notwendig. Genau diese Anforderungen erfüllt das Verfahren aus Abschnitt 6.4, das sich leicht auf den Stereo-Fall erweitern ließ.

Die Geschwindigkeit der Rekonstruktion kann natürlich nicht mit dem von Spezialhardware konkurrieren, kommt aber in die selbe Größenordnung wie aktuelle Softwaresysteme. Wichtig ist dabei, daß die kommerziell erhältlichen Systeme nur Grauwertbilder zur Korrelation verwenden, also nur ein Viertel der Bilddaten verarbeiten (8 statt 32 Bit pro Pixel). Der entwickelte Stereoalgorithmus zeigt eine konstante Leistung über mehrere Größenordnungen der zu verarbeitenden Datenmenge. Er ist nicht nur zur schnellen Rekonstruktion mit beschränkter Bildgröße geeignet, sondern kann alle vorkommenden Bilder verarbeiten. Es wurde gezeigt, daß die Berücksichtigung der Cache-Hierarchie und die Verwendung von SIMD-Befehlen bei der Implementierung der Korrelation eine deutliche Geschwindigkeitssteigerung bewirken.

Bei der Endoskopie stellt die Darstellung des rekonstruierten 3D-Modells einen großen Teil der Anwendung dar. Es war wichtig, nicht mit den Disparitäten oder den 3D-Koordinaten aufzuhören, sondern sie so aufzubereiten, daß sie in möglichst kurzer Zeit von einer hardwarebeschleunigten Grafikkarte dargestellt werden können. Obwohl die Triangulation und die Netzgenerierung das Lösen eines Gleichungssystems pro Punkt und die Ermittlung von Nachbarschaftsbeziehungen beinhalten, benötigen sie nicht mehr Zeit als zum Beispiel die Bildentzerrung.

## 13.2 Ausblick

Das System ist in der Lage, eine Gesichtsoberfläche aus der Entfernung von mehreren Metern mit ausreichender Geschwindigkeit und Dichte zu rekonstruieren. Die zweite Anwendung, das Headtracking beim MEG, kann nun darauf aufbauend implementiert werden. Wie groß die damit erzielbare Genauigkeit sein wird, wird sich erst in praktischen Tests zeigen können, zumal sie maßgeblich vom verwendeten Verfahren zum Oberflächenmatching abhängt.

Als Ergänzung zur Korrelation, deren Modelle (Abbildung 12.3(b)) immer noch Löcher enthalten können, wird das auf anisotroper Diffusion basierende Verfahren nach [2] implementiert. Es berechnet eine geschlossene Minimalenergiefläche, deren Glättungsterm vom Gradient des Bildes abhängt und so an Helligkeitssprüngen große Disparitätsänderungen zuläßt. Die Dauer einer Rechnung beträgt in etwa 1–3 Minuten, je nach Komplexität der Szene. Als Anfangsschätzung wird das Ergebnis der Korrelation benutzt. Eine laufende Diplomarbeit beschäftigt sich damit, dieses Verfahren um eine explizite Modellierung von Verdeckungen zu ergänzen. Die Dis-

paritäten an diesen Stellen werden bis jetzt nur vom Glättungsterm beeinflusst und repräsentieren keinen Teil der echten Objektoberfläche. Vor allem beim Zusammenfügen mehrerer Teilansichten zu einem Gesamtmodell wirken sich diese „Phantomflächen“ störend aus, sie sollten deshalb zum frühest möglichen Zeitpunkt entfernt werden. Das Verfahren eignet sich vor allem dafür, hochqualitative 3D-Modelle zu liefern, wenn die Rechenzeit nicht im Echtzeitbereich liegen muß. Bei der Offline-Vermessung von Organen mit Bildern des Stereo-Endoskops liefert der Algorithmus bereits deutlich bessere Ergebnisse als die Korrelation.

Weiterhin ist denkbar, die Minimumssuche ebenfalls mit MMX-Befehlen zu implementieren und die zur Subpixelinterpolation notwendigen Fließkommaberechnungen in der SSE-Einheit durchzuführen. Dies würde das Umschalten der Fließkommaregister in den MMX-Modus und zurück überflüssig machen und den Speichertransfer durch die Verwendung zusätzlicher Register reduzieren. Ein Teil der gesparten Zeit könnte dann in die Verbesserung der Ergebnisse an Objektkanten investiert werden [27].

Die erzeugten Oberflächen enthalten oft deutlich mehr Dreiecke, als zur Beschreibung der Geometrie nötig sind. Wenn mehrere davon zu größeren Modellen zusammengesetzt werden sollen, ist es sinnvoll sie aus möglichst großen Dreiecken zusammenzusetzen und dadurch Punkte einzusparen. [69] beschäftigt sich damit, die Netze zu filtern, zu reduzieren und zu vereinigen. Für diese Arbeit wurden solche Reduktionsmethoden allerdings nicht benötigt und wären auch nicht schnell genug gewesen.

## 13.3 Zusammenfassung

Im Rahmen dieser Arbeit wurde ein aus Hard- und Software bestehendes System zur schnellen Rekonstruktion dreidimensionaler Oberflächen entwickelt. Ausgehend von einer geplanten Anwendung, die mit existierenden Systemen nicht realisierbar war, wurde zunächst festgestellt wo die Stärken und Schwächen der Systeme liegen, darauf basierend ein geeignetes Verfahren gewählt und die zu lösenden Teilaufgaben identifiziert. Die Arbeit konzentriert sich auf die Entwicklung möglichst allgemein verwendbarer Kernalgorithmen, ohne dabei die geplanten Anwendungen aus den Augen zu verlieren. Insbesondere wurde auf modulares Design geachtet, so daß sich die einzelnen Bausteine leicht für beliebige Anwendungen verwenden lassen, die eine Funktionalität zur 3D-Rekonstruktion benötigen.

Das erste Glied in der Bildverarbeitungskette bildet eine flexible und relativ preisgünstige Farb-Stereokamera aus zwei Standard-Blockkameras. Der auf programmier-

barer Logik aufbauende Stereo-Framegrabber überträgt die Bilder in den Hauptspeicher, ohne die CPU zu belasten.

Der Software-Teil der Arbeit beinhaltet die Entwicklung und Weiterentwicklung von Algorithmen zur Kalibrierung der Kameras und metrischen Rekonstruktion dreidimensionaler Modelle, die dann applikationsspezifisch weiterverarbeitet werden können. Durch Ausnutzen algorithmischer und implementierungstechnischer Optimierungen in allen Stufen der Verarbeitung konnte eine konkurrenzfähige Geschwindigkeit erreicht werden, wobei die durchgehende Verwendung von Farbinformation eine wesentliche Erweiterung gegenüber existierenden Systemen darstellt. Mit Hilfe des Ausdrucks eines Laserdruckers können sowohl die Stereokamera als auch das Stereo-Endoskop in weniger als einer Minute mit hoher Genauigkeit kalibriert werden.

Rektifikation, Disparitätsberechnung und Triangulation lassen sich unverändert auch zur Rekonstruktion aus unkalibrierten Bildsequenzen einsetzen. Als Ergänzung in diese Richtung existieren ein global optimierender Stereoalgorithmus, dessen Stärken und Schwächen komplementär zu denen der Korrelation sind, und ein Verfahren zur Reduktion der sehr schnell sehr groß werdenden Datenmengen.

# Danksagung

Ich danke allen, die mich beim Entstehen dieser Arbeit unterstützt haben, insbesondere danke ich

- Prof. Dr. Reinhard Männer und PD. Dr. Jürgen Hesser für die Möglichkeit, ein interessantes Thema nach meinen Vorstellungen bearbeiten zu können und für jederzeit offene Türen und Ohren,
- Prof. Dr. Bernd Jähne für die Übernahme des Koreferates,
- Dennis Maier, Marc Deutscher, Alexander Dejon, Stefan Weber und Andreas Rößle für ihr Interesse und die gute Arbeit, die sie mit ihren Diplomarbeiten geleistet haben,
- Günther Stahl von der mechanischen Werkstatt des physikalischen Instituts der Universität Heidelberg für die professionelle Fertigung Kameragehäuses,
- und ganz besonders meiner Mutter, die mir das Studium ermöglicht und mich immer voll unterstützt hat.



# Abbildungsverzeichnis

2.1	Funktionsprinzip eines Laserscanners. . . . .	11
2.2	Beispiel einer „Shape From Shading“-Berechnung. . . . .	12
2.3	Rekonstruktion aus Objektkonturen. . . . .	13
4.1	Aufbau der Stereokamera. . . . .	24
4.2	Kamerablock der Serie EVI-370. . . . .	25
4.3	Stereoendoskop. . . . .	26
5.1	FPGA-Karte microEnable und Kamerainterface . . . . .	28
5.2	Logikblöcke und Verbindungen eines XILINX 4000 FPGAs. . . . .	30
5.3	Vertikale und horizontale Synchronisationssignale am Ausgang des Video Analog-Digital Wandlers SAA7111. . . . .	32
5.4	Übertragung eines Pixels im RGB888 Modus. . . . .	33
5.5	Darstellung der Simulation von Algorithmus 5.2. . . . .	38
5.6	Simulation eines Halbbildes mit CHDL. . . . .	39
5.7	Aufnahme einer Zeile in der Simulation. . . . .	40
5.8	Hardwarekomponenten und Aufbau des Framegrabbers. . . . .	41
6.1	Abbildung eines Punktes durch Zentralprojektion. . . . .	46
6.2	CCD-Koordinatensystem (Pixel). . . . .	47
6.3	Koordinatensysteme bei der Abbildung durch eine Lochkamera. . . . .	49
6.4	Linsenverzerrung. . . . .	54
6.5	Übergang des globalen Koordinatensystems vom Kalibriermuster zur linken Kamera. . . . .	61
6.6	Schematischer Ablauf der Stereokalibrierung . . . . .	62
6.7	Oberfläche des Kalibrierprogrammes. . . . .	64
6.8	Ergebnis einer Stereokalibrierung. . . . .	64

7.1	Epipolare Ebene und Epipolarlinien für einen Punkt. . . . .	66
7.2	Rektifikation durch Reprojektion in eine gemeinsame Bildebene. . . . .	68
7.3	Transformiertes Bild vor der Korrektur der Scherung. . . . .	70
7.4	Implementierung der bilinearen Interpolation mit MMX Befehlen. . . . .	73
8.1	3D-Punkt und Disparität zwischen seinen Bildpunkten. . . . .	76
8.2	Disparitätsvolumen. . . . .	77
8.3	Unterschiedliche Ausnutzung des Caches beim Durchsuchen der SAD in Disparitätsrichtung. . . . .	81
8.4	Gleitende Fenstersumme in $x$ -Richtung. . . . .	81
8.5	Aufsummieren der Fenster in $y$ -Richtung. . . . .	82
8.6	Minimumsuche von rechts nach links. . . . .	83
8.7	Suchbereiche und -richtungen. . . . .	84
8.8	Vorder-, Hintergrund und in einem Bild verdeckte Fläche. . . . .	85
8.9	Interpolationsparabel und Eindeutigkeits-Schwelle. . . . .	87
8.10	Minimaler Sortiergraph für den Median von 9 Werten. . . . .	88
9.1	Vorwärtsschnitt von den Bildpunkten zum 3D-Punkt. . . . .	92
9.2	Alle Fälle, in denen Dreiecke generiert werden. . . . .	94
9.3	Fortschreiten der Triangulierung. . . . .	95
10.1	Reprojektionsfehler einer Aufnahme, zehnfach verstärkt. . . . .	100
10.2	Anordnung der Kameras bei der Beispielkalibrierung. . . . .	101
10.3	Auswirkung der Anzahl verwendeter Kalibrierenebenen auf einzelne Pa- rameter bei der Kamerakalibrierung. . . . .	102
10.4	Auswirkung der Anzahl verwendeter Kalibrierenebenen auf einzelne Pa- rameter bei der Kamerakalibrierung. . . . .	103
10.5	Auswirkung der Anzahl verwendeter Kalibrierenebenen auf einzelne Pa- rameter bei der Kamerakalibrierung. . . . .	104
11.1	Entzerrtes Bildpaar. . . . .	106
11.2	Ergebnis einer Disparitätsberechnung. . . . .	107
11.3	Ergebnis der Laufzeitmessungen am kleinsten Testbild mit zwei ver- schiedenen Prozessoren. . . . .	111
11.4	Laufzeitmessungen für die restlichen vier Bilder. . . . .	112
11.5	Vergleich der Laufzeiten aller Algorithmen bei allen Bildern. . . . .	114
11.6	Quotient der Laufzeit von Pentium und Athlon bei gleichem Test. . . . .	115
11.7	Vergleich der Ausführungsgeschwindigkeit dreier Implementierungen des Medianfilters. . . . .	116

11.8 Medianfilterung der Disparitätskarte. . . . .	118
12.1 Threads des Rekonstruktionsprogramms. . . . .	120
12.2 Zeitlicher Ablauf der Rekonstruktion. . . . .	121
12.3 Beispielrekonstruktion. . . . .	123



# Literaturverzeichnis

- [1] ABDEL-AZIZ, Y. I. und H. M. KARARA: *Direct Linear Transformation from Comparator Coordinates into Object Space Coordinates in Close-Range Photogrammetry*. In: *Proceedings of the Symposium on Close-Range Photogrammetry*, Seiten 1–18, Falls Church, VA, 1971. American Society of Photogrammetry.
- [2] ALVAREZ, LUIS, RACHID DERICHE, JAVIER SÁNCHEZ und JOACHIM WEICKERT: *Dense Disparity Map Estimation Respecting Image Discontinuities: A PDE and Scale-Space Based Approach*. Technischer Bericht RR-3874, INRIA, Januar 2000.
- [3] ANSI/IEEE: *1076-1993 VHDL Language Reference Manual*. Institute of Electrical and Electronics Engineers, Inc., 1993. ISBN 1-55937-376-8.
- [4] ANSI/IEEE: *1364-2001 IEEE Standard for Verilog Hardware Description Language*. Institute of Electrical and Electronics Engineers, Inc., 2001. ISBN 0-7381-2827-9.
- [5] ATKINSON, K. B. (Herausgeber): *Close Range Photogrammetry and Machine Vision*. Whittles Publishing, 1996. ISBN 1-870325-46-X.
- [6] BAKSTEIN, HYNEK und RADIM HALÍŘ: *Camera Calibration with a Simulated Three Dimensional Calibration Object*. In: SVOBODA, TOMÁŠ (Herausgeber): *Proceedings of the Czech Pattern Recognition Workshop*, Seiten 99–106, Prague, Czech Republic, Februar 2000. Czech Pattern Recognition Society. ISBN 80-238-5215-9.
- [7] BATISTA, JORGE, JORGE DIAS, HELDER ARAÚJO und ANÍBAL T. ALMEIDA: *Iterative Monoplane Camera Calibration - A Multi-Step Approach*. Technischer Bericht, Institute of Systems and Robotics, University of Coimbra, März 1996.

- [8] BRONŠTEJN, I. N. und KONSTANTIN A. SEMENDJAEV: *Taschenbuch der Mathematik*. Harri Deutsch, Thun; Frankfurt/Main, 1989. ISBN 3-87144-492-8.
- [9] BROSCHE, O., P. DILLINGER, K. KORNMESSER, A. KUGEL, R. MÄNNER, R. RISSMANN, S. RÜHL, H. SIMMLER, H. SINGPIEL, R. LAY und K.-H. NOFFZ: *Simulating FPGA-Coprocessors using FPGA Development System CHDL*. In: *PACT 98 Workshop on Reconfigurable Computing*, Seiten 78–82, Paris, 1998.
- [10] BROSCHE, O., P. DILLINGER, K. KORNMESSER, A. KUGEL, R. MÄNNER, M. SESSLER, H. SIMMLER, H. SINGPIEL, S. RÜHL, R. LAY, K.-H. NOFFZ und L. LEVINSON: *MicroEnable - A Reconfigurable FPGA Coprocessor*. In: *Proc. 4th Workshop on Electronics for LHC Experiments*, Seiten 402–406, Rome, Italy, 1998. CERN/LHCC/98-36.
- [11] CANNY, J.: *A Computational Approach to Edge Detection*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 8(6):679–698, 1986.
- [12] DEUTSCHER, MARC: *Aufbau eines Farb-Stereokamerasystems mit einem programmierbaren Stereoframegrabber auf FPGA-Basis*. Diplomarbeit, Lehrstuhl für Informatik V, Universität Mannheim, 1999.
- [13] DEVILLARD, NICOLAS: *Fast Median Search: An ANSI C Implementation*. <http://ndevilla.free.fr/median/>, 1997.
- [14] EGNAL, GEOFFREY und RICHARD P. WILDES: *Detecting Binocular Half-Occlusions: Empirical Comparisons of Four Approaches*. In: *IEEE Conference on Computer Vision and Pattern Recognition*, 2000.
- [15] FAUGERAS, OLIVIER: *Three-Dimensional Computer Vision: A Geometric Viewpoint*. The MIT Press, Cambridge, Massachusetts, 1993. ISBN 0-262-06158-9.
- [16] FAUGERAS, OLIVIER und OTHERS: *Real Time Correlation-based Stereo: Algorithm, Implementations and Applications*. Technischer Bericht RR-2013, INRIA, 1993.
- [17] FUA, P.: *A Parallel Stereo Algorithm that Produces Dense Depth Maps and Preserves Image Features*. Machine Vision and Applications, 6(1), 1993.
- [18] FUSIELLO, A., V. ROBERTO und E. TRUCCO: *Efficient Stereo with Multiple Windowing*. In: *IEEE Conference on Computer Vision and Pattern Recognition*, Seiten 858–863, San Juan, Puerto Rico, Juni 1997.

- 
- [19] FUSIELLO, A., E. TRUCCO und A. VERRI: *Rectification with Unconstrained Stereo Geometry*. In: CLARK, A. F. (Herausgeber): *Proceedings of the British Machine Vision Conference*, Seiten 400–409. BMVA Press, September 1997.
- [20] GIVENS, W.: *Computation of Plane Unitary Rotations Transforming a General Matrix to Triangular Form*. *SIAM Journal for Applied Mathematics*, 6:26–50, 1958.
- [21] GOLUB, GENE H. und CHARLES F. VAN LOAN: *Matrix Computations*. The Johns Hopkins University Press, 3. Auflage, 1996. ISBN 0-8018-5414-8.
- [22] GUPTA, R. und R. I. HARTLEY: *Linear pushbroom cameras*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(9):963–975, September 1997.
- [23] HARTLEY, RICHARD und ANDREW ZISSERMAN: *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. ISBN 0-5216-2304-9.
- [24] HARTLEY, RICHARD I.: *Theory and Practice of Projective Rectification*. *International Journal of Computer Vision*, 35(2):1–16, November 1999.
- [25] HARTLEY, RICHARD I. und PETER STURM: *Triangulation*. *Computer Vision and Image Understanding CVIU*, 68(2):146–157, 1997.
- [26] HEINOL, HORST G.: *Untersuchung und Entwicklung von modulationslaufzeitbasierten 3D-Sichtsystemen*. Doktorarbeit, Universität Siegen, 2001.
- [27] HIRSCHMÜLLER, HEIKO: *Improvements in Real-Time Correlation-Based Stereo Vision*. In: *IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Stereo and Multi-Baseline Vision*, Dezember 2001.
- [28] HOFFMANN, M., F. BRUGMANNS und M. GÜNTHER: *Endoskopische 3D-Vermessung von Organen*. In: *9. Heiligenstädter Kolloquium, iba e.V.*, Seiten 368–869, Rosenhof Heiligenstadt, 1998.
- [29] INTEL CORPORATION: *IA-32 Intel Architecture Software Developer's Manual, Volume 2: Instruction Set Reference Manual*, 2001. <http://developer.intel.com/design/pentiumiii/manuals>.
- [30] INTEL CORPORATION: *Intel Architecture Software Optimization Reference Manual*, 2001. <http://developer.intel.com/design/pentiumiii/manuals>.

- [31] INTEL CORPORATION: *Open Source Computer Vision Library*. <http://www.intel.com/research/mrl/research/opencv>, 2001.
- [32] INTILLE, S. S. und A. F. BOBICK: *Disparity-space Images and Large Occlusion Stereo*. In: *European Conference on Computer Vision*, Seiten 179–186, 1994.
- [33] JÄHNE, B.: *Digitale Bildverarbeitung*. Springer, Berlin; Heidelberg; New York, 4. Auflage, 1997. ISBN 3-540-61379-X.
- [34] JÄHNE, BERND, HORST HAUSECKER und PETER GEISLER (Herausgeber): *Handbook of Computer Vision and Applications*. Academic Press, 1999. ISBN 0-12-379770-5.
- [35] KANADE, TAKEO: *Development of a Video-Rate Stereo Machine*. In: *Proceedings of 94 ARPA Image Understanding Workshop*, Seiten 549–558, Monttrey Ca., November 1994.
- [36] KANADE, TAKEO und MASATOSHI OKUTOMI: *A Stereo Matching Algorithm with an Adaptive Window: Theory and Experiment*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(9):920–932, 1994.
- [37] KARARA, H.M. (Herausgeber): *Handbook of Non-Topographic Photogrammetry*. American Society of Photogrammetry, 1979.
- [38] KLETTE, R., A. KOSCHAN, K. SCHLÜNS und V. RODEHORST: *Surface Reconstruction based on Visual Information*. Technischer Bericht 95/6, Dep. of Computer Science, The University of Western Australia, Juli 1995.
- [39] KONOLIGE, K.: *Small Vision Systems: Hardware and Implementation*. In: *Eighth International Symposium on Robotics Research*, Hayama, Japan, Oktober 1997.
- [40] KORNMESSER, K.: *The FPGA Development System CHDL*. <http://www-li5.ti.uni-mannheim.de/fpga/chdl>, 1996-2001.
- [41] KORNMESSER, K., A. KUGEL und R. MÄNNER: *The FPGA Development System CHDL*. In: *IEEE Symposium on Field-Programmable Custom Computing Machines*, Rohnert Park, California, April 2001.
- [42] KOSCHAN, ANDREAS, VOLKER RODEHORST und KATHRIN SPILLER: *Color Stereo Vision Using Hierarchical Block Matching and Active Color Illumination*. In: *International Conference on Pattern Recognition*, Band 1, Seiten 835–839, Vienna, Austria, August 1996.

- [43] KUTULAKOS, KIRIAKOS N.: *Shape from the Light Field Boundary*. In: *IEEE Conference on Computer Vision and Pattern Recognition*, Seiten 53–59, San Juan, Puerto Rico, Juni 1997.
- [44] LEE, K. M. und C. C. J. KUO: *Shape from shading with a linear triangular element surface model*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(8):815–822, 1993.
- [45] *Lenna Sjööblom, Miss November 1972*. Playboy Magazine, November 1972. <http://www.lenna.org>, <http://www.playboy.com/playmates/directory/197211.html>.
- [46] LENZ, REIMAR K. und ROGER Y. TSAI: *Techniques for Calibration of the Scale Factor and Image Center for High Accuracy 3-D Machine Vision Metrology*. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):713–720, September 1988.
- [47] MAIER, DENNIS: *Schnelle Rekonstruktion dreidimensionaler Oberflächen aus Stereo-Farbbildaufnahmen*. Diplomarbeit, Lehrstuhl für Informatik V, Universität Mannheim, Februar 1999.
- [48] MCDONNELL, M.J.: *Box-filtering techniques*. *Computer Graphics and Image Processing*, 17:65–70, 1981.
- [49] MEERBERGEN, G. VAN, M. VERGAUWEN, M. POLLEFEYS und L. VAN GOOL: *A hierarchical stereo algorithm using dynamic programming*. In: *IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Stereo and Multi-Baseline Vision*, Kauai, Hawaii, Dezember 2001.
- [50] MÜHLMANN, KARSTEN, DENNIS MAIER, JÜRGEN HESSER und REINHARD MÄNNER: *Calculating Dense Disparity Maps from Color Stereo Images, an Efficient Implementation*. *International Journal of Computer Vision*, 47(1):79–88, Mai 2002.
- [51] NAYAR, S. K.: *Catadioptric omnidirectional camera*. In: *IEEE Conference on Computer Vision and Pattern Recognition*, Seiten 482–488, San Juan, Puerto Rico, Juni 1997.
- [52] OKUTOMI, MASATOSHI und YASUHIRO KATAYAMA: *A Simple Stereo Algorithm to Recover Precise Object Boundaries and Smooth Surfaces*. In: *IEEE Conference on Computer Vision and Pattern Recognition, Workshop on Stereo and Multi-Baseline Vision*, Kauai, Hawaii, Dezember 2001.

- [53] WOO, MASON, JACKIE NEIDER, TOM DAVIS, DAVE SHREINER und OPENGL ARCHITECTURE REVIEW BOARD: *OpenGL Programming Guide: The Official Guide to Learning OpenGL, Version 1.2*. Addison-Wesley, 3. Auflage, August 1999. ISBN 0-201-60458-2.
- [54] SHREINER, DAVE (Herausgeber): *OpenGL Reference Manual: The Official Reference Document to OpenGL, Version 1.2*. Addison-Wesley, 3. Auflage, Dezember 1999. ISBN 0-201-65765-1.
- [55] PHILIPS SEMICONDUCTORS: *SAA7111 Video Input Processor (VIP)*, Mai 1998.
- [56] PRESS, WILLIAM H., BRIAN P. FLANNERY, SAUL A. TEUKOLSKY und WILLIAM T. VETTERLING: *Numerical Recipes: The Art of Scientific Computing*. Cambridge University Press, Cambridge (UK) and New York, 2. Auflage, 1992.
- [57] SCHARSTEIN, D. und R. SZELISKI: *A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms*. International Journal of Computer Vision, 47(1):7–42, Mai 2002.
- [58] SILICON SOFTWARE GMBH: *microEnable Framegrabberhandbuch*, 1999.
- [59] SLABAUGH, G., B. CULBERTSON, T. MALZBENDER und R. SCHAFER: *A Survey of Methods for Volumetric Scene Reconstruction from Photographs*. Technischer Bericht 1, Center for Signal and Image Processing, Georgia Institute of Technology, 2001.
- [60] SMITH, JOHN L.: *Implementing Median Filters in XC4000E FPGAs*. Xcell: The Quarterly Journal for Xilinx Programmable Logic Users, 23, 1996.
- [61] SONY CORPORATION: *EVI-370 Series Instruction Manual*, 1999.
- [62] SONY CORPORATION: *EVI-D30/D31 Command List*, 1999.
- [63] STRASSACKER, G.: *Rotation, Divergenz und das Drumherum : Eine Einführung in die Elektromagnetische Feldtheorie*. B.G. Teubner, Stuttgart, 3. Auflage, 1992.
- [64] STURM, P.: *Critical Motion Sequences for Monocular Self-Calibration and Uncalibrated Euclidean Reconstruction*. Technischer Bericht, INRIA, 1996.
- [65] SUN, C.: *A Fast Stereo Matching Method*. In: *Digital Image Computing: Techniques and Applications*, Seiten 95–100. Massey University, Auckland, New Zealand, Dezember 1997.

- 
- [66] TSAI, ROGER Y.: *A Versatile Camera Calibration Technique for High-Accuracy 3D Machine Vision Metrology Using Off-the-Shelf TV Cameras and Lenses*. IEEE Journal of Robotics and Automation, Seiten 323–344, August 1987.
- [67] VOGEL, HELMUT: *Gerthsen Physik*. Springer-Verlag, Berlin, Heidelberg, 19. Auflage, 1997. ISBN 3-540-62988-2.
- [68] WEB3D CONSORTIUM: *The Virtual Reality Modeling Language Version 1.0 Specification*. <http://www.vrml.org/technicalinfo/specifications/vrml1.0.htm>, November 1995.
- [69] WEBER, STEFAN: *Simplification, Anisotropic Filtering and Merging of Surface Meshes*. Diplomarbeit, Lehrstuhl für Informatik V, Universität Mannheim, Januar 2002.
- [70] WILLSON, REG G.: *Tsai Camera Calibration Software*. <http://www-2.cs.cmu.edu/~rgw/TsaiCode.html>, Oktober 1995.
- [71] WOODFILL, JOHN und BRIAN VON HERZEN: *Real-Time Stereo Vision on the PARTS Reconfigurable Computer*. In: *IEEE Symposium on FPGAs for Custom Computing Machines*, Seiten 201–210, April 1997.
- [72] XILINX INC.: *XC4000E and XC4000X Series Field Programmable Gate Arrays*, Mai 1999.
- [73] XU, G. und Z. ZHANG: *Epipolar Geometry in Stereo, Motion and Object Recognition*. Kluwer Academic Publishers, 1996.
- [74] ZABIH, R. und J. WOODFILL: *Non-parametric Local Transforms for Computing Visual Correspondence*. In: *3rd European Conference on Computer Vision*, Seiten 150–158, Mai 1994.
- [75] ZHANG, RUO, PING-SING TSAI, JAMES CRYER und MUBARAK SHAH: *Shape from Shading: A Survey*. IEEE Transactions on Pattern Analysis and Machine Intelligence, 21(8):690–706, August 1999.
- [76] ZHANG, ZHENGYOU: *A Flexible New Technique for Camera Calibration*. Technischer Bericht MSR-TR-98-71, Microsoft Research, Dezember 1998.
- [77] ZITNICK, C. und T. KANADE: *Cooperative Algorithm for Stereo Matching and Occlusion Detection*. Technischer Bericht CMU-RI-TR-99-35, Robotics Institute, Carnegie Mellon University, Oktober 1999.