Diplomarbeit

Automatisierte Bildorientierung mit projektiven Methoden

zum Zwecke der Erlangung des akademischen Grades eines

Diplomingenieurs (Dipl.Ing.)

ausgeführt am

Institut für Photogrammetrie und Fernerkundung
Technische Universität Wien

unter der Anleitung von

o.Univ.Prof. Dr.techn. Karl Kraus

und

Dr.techn. Camillo Ressl

ausgeführt von

Andreas Roncat

Matr.Nr. 0025743

Neuwaldeggerstraße 18a

A-1170 Wien

. . . . . Wien, im Februar 2006

Inhaltsverzeichnis

1 Einleitung **2**

2 Vorbemerkungen zur digitalen Bildverarbeitung **4**
2.1 Digitale Bilder 4
2.2 Ausgewählte Operationen in digitalen Bildern 5
2.2.1 Filterungen 5
2.2.2 Bildpyramiden 7

3 Interest-Operatoren **8**
3.1 Definition von Interest-Operatoren 8
3.2 Der Harris-*Corner-und-Edge-Detector* 9
3.3 Der *Keypoint-Descriptor* von Lowe 10
3.3.1 Extremwertsuche im Scale-Space 10
3.3.2 Lokalisierung der Keypoints 12
3.3.3 Zuweisung der Orientierung. 13
3.3.4 Berechnung des Keypoint-Descriptors 14
3.3.5 Matching der Keypoints in zwei Bildern 17

4 Kurze Einführung in die Projektive Geometrie **18**
4.1 Projektive Ebenen und Projektive Räume 18
4.1.1 Dualität 19
4.1.2 Koordinatendarstellung 19
4.2 Kollineationen und Korrelationen 22
4.3 Kegelschnitte und Quadriken 22
4.4 Perspektive in der Projektiven Geometrie 24

5 Projektive Relative Orientierung und Selbst-Kalibrierung **27**
5.1 Die Fundamentalmatrix $\mathbf{F}_{12}$ 27
5.2 Berechnung der Fundamentalmatrix 29
5.2.1 Linearer Normierter 8-Punkte-Algorithmus 29
5.2.2 Minimaler 7-Punkte-Algorithmus 30
5.3 Robuste Schätzung der Fundamentalmatrix 31
5.3.1 Random Sample Consensus (RANSAC) 32
5.4 Die Essentielle Matrix $\mathbf{E}_{12}$ 33

5.4.1 Anpassung der projektiven Tiefe 35
5.5 Selbstkalibrierung 35
5.5.1 Selbstkalibrierung mit den *Kruppa-Gleichungen* 36
5.5.2 Selbstkalibrierung mit der dualen absoluten Quadrik 37
5.6 Berechnung der Orientierungsparameter 38

6 Experimentelle Ergebnisse **40**
6.1 EO Device 40
6.2 Messkeller 42
6.3 Nahbereichsszene mit eben begrenzten Objekten 51

7 Zusammenfassung und Ausblick **54**

A MATLAB-Sourcecodes **57**
A.1 Hauptroutinen 57
A.2 Hilfsroutinen 74

Zusammenfassung

In dieser Arbeit wird eine Methode zur automatischen Relativen Orientierung von kalibrierten Bildern vorgestellt. Im ersten Schritt werden Interestpoints und ihre Merkmale mit der sogenannten Scale Invariant Feature Transform (SIFT) und dem Keypoint Descriptor, beide entwickelt von David Lowe, aus den Bildern extrahiert. Nach dem Auffinden von korrespondiernden Keypoints über Nearest Neighbourhood wird aus den Punktkorrespondenzen die Fundamentalmatrix mittels eines robusten Schätzverfahrens (RANSAC) berechnet und daraus die relative Orientierung abgeleitet. Diese Werte sind als Näherungswerte für weitergehende Orientierungsprozesse, etwa für eine Bündelblockausgleichung, zu betrachten.

Abstract

In this thesis, a method for an automatic computation of the relative orientation of calibrated images is presented. As a first step, interest points and their features are extracted by the use of the so called Scale Invariant Feature Transform (SIFT) and the keypoint descriptor, both developed by David Lowe. After matching these keypoints with nearest neighbourhood, the fundamental matrix is calculated with a robust parameter estimation technique (RANSAC) and the relative orientation is determined. The so retrieved orientation parameters ought to be considered as approximate values for a further orientation process such as bundle block adjustment.

Danksagung

Bedanken möchte ich mich zuerst beim Begutachter dieser Arbeit, Herrn Prof. Dr. Karl Kraus. Nicht nur, aber auch für das Wecken meines Interesses am Thema dieser Diplomarbeit (wie nicht anders zu erwarten, wusste er bereits vor mir, dass es mich interessieren würde).

Weiters bedankt seien die Mitarbeiter des Instituts für Photogrammetrie und Fernerkundung für ihre hilfreiche Unterstützung sowie Herr Prof. Dr. Hans Havlicek für seine gleichermaßen niveau- wie humorvollen Vorlesungen über Projektive Geometrie, ohne die ich wohl den Einstieg in diese Thematik nicht geschafft hätte. Meiner Familie und meinen Freunden spreche ich hiermit den Dank für ihr Vertrauen in das Gelingen dieser Arbeit aus.

Last, but not least gebührt meinem Betreuer Dr. Camillo Ressl die aufrichtigste Anerkennung für seine umfassend kompetente, gewissenhafte, geduldige und hilfreiche Betreuung. Leider konnte ich seinen Ausführungen nicht immer gleich im ersten Moment folgen...

Kapitel 1

Einleitung

Der Orientierungsaufgabe kommt in der Photogrammetrie besondere Bedeutung zu, steht sie doch am Beginn jedes photogrammetrischen Auswerteprozesses, ihre Qualität hat entscheidenden Einfluss auf die Qualität des Endergebnisses.

Im Luftbildfall wird die Bestimmung der – Relativen wie Absoluten – Orientierung durch den verstärkten Einsatz von GPS und IMU zunehmend erleichtert [Kraus, 2004]. Anders sieht es hingegen in der terrestrischen Photogrammetrie aus, wo unterschiedlichste Aufnahmeentfernungen und Drehwinkel autreten können. Hier gestaltet sich vor allem die Ermittlung von Näherungswerten für die Orientierungsparameter recht schwierig, besonders in passpunktlosen Bereichen. Das Messen von Verknüpfungspunkten in den Bildern geschieht nachwievor meist manuell und stellt einen gleichermaßen mühsamen wie zeitraubenden Arbeitsschritt dar.

Seit Jahrzehnten existiert die Vision der *Real-Time Photogrammetry* [Förstner, 2005]. Um diesem Traum ein wenig näher zu kommen, wird in dieser Arbeit ein Verfahren präsentiert, um ausgehend von digitalen Bildern und ihren Kalibrierungsdaten die Relativen Orientierungen automatisch zu bestimmen.

Zuerst werden aus den Bildern *Interest Points* und ihre Merkmale (die in einem 128-dimensionalen Vektor, dem sogenannten *Keypoint Descriptor*, verspeichert werden) mittels der Methode der *Scale Invariant Feature Transform* von David Lowe extrahiert. Diese Merkmale sind sowohl maßstabs- als auch rotationsinvariant und auch zum Teil invariant gegenüber Änderungen der Beleuchtungsverhältnisse (siehe Abschnitt 3.3.)

Nach der Extraktion der Keypoint Descriptors wurden mittels *Nearest Neighbourhood* korrespondierende Keypoints in den jeweiligen Bildpaaren ermittelt (siehe Abschnitt 3.3.5).

Aus den nun vorliegenden homologen Punktepaaren können mit den Algorithmen der Projektiven Geometrie die Parameter der Relativen Orientierung ermittelt werden (siehe Kapitel 5). Zuerst wurden mit dem soge-

nannten *8-Punkte-Algorithmus* die *Fundamentalmatrizen* $\mathbf{F}_{12}$ berechnet. Da eine größere Anzahl von Fehlkorrespondenzen zu erwarten war, wurde das robuste Schätzverfahren RANSAC [Hartley und Zisserman, 2001] eingesetzt, um diese zu eliminieren (siehe Abschnitt 5.3.1).

Mit den bekannten Inneren Orientierungen der Aufnahmen wurden aus den $\mathbf{F}_{12}$-Matrizen die sogenannten *Essentiellen Matrizen* $\mathbf{E}_{12}$ berechnet, aus denen eindeutig die Parameter der Relativen Orientierung hervorgehen.

Da die Relative Orientierung maßstabsfrei ist, kann noch die sogenannte *projektive Tiefe* [Rodehorst, 2004] zur Maßstabsbestimmung eingeführt werden (siehe Abschnitt 5.4.1).

Die erwähnten Arbeitsschritte wurden in MATLAB implementiert, wobei aufgrund der deutlich kürzeren Rechenzeit für die Ermittlung der Keypoint Descriptors das Demo-Programm `siftDemoV4` von David Lowe verwendet wurde[1]. Die MATLAB-Sourcecodes finden sich im Anhang dieser Arbeit.

Die vorgestellte Methode wurde anhand von drei Nahbereichs-Beispielen unter unterschiedlichen Verhältnissen getestet. Sie sind in Kapitel 6 beschrieben.

Der Vollständigkeit halber werden in Abschnitt 5.5 noch zwei Verfahren der *Selbstkalibrierung* vorgestellt, die bei der Verwendung von unkalibrierten Aufnahmen zum Einsatz kommen.

[1] URL: `http://www.cs.ubc.ca/spider/lowe/keypoints/siftDemoV4.zip`, Zugriff: 7. Juli 2005.

Kapitel 2

Vorbemerkungen zur digitalen Bildverarbeitung

2.1 Digitale Bilder

Ein digitales Bild ist im Wesentlichen eine diskrete zweidimensionale Funktion $f(x, y)$, wobei x und y ebene kartesische Koordinaten darstellen. Digitale Bilder sind matrixartig verspeichert. Der mathematische Zusammenahng zwischen Zeilenindex i und Spaltenindex j mit den Bildkoordinaten (x, y) kann je nach Software unterschiedlich sein. In dieser Arbeit hat die Mitte des linken oberen Elements ($i = 1$, $j = 1$) die Bildkoordinaten $(1, 1)$, die positive x-Achse weist nach rechts, die positive y-Achse nach unten[1]. Es handelt sich hierbei also um ein Linkssystem, wie in Abbildung 2.1 ersichtlich ist.

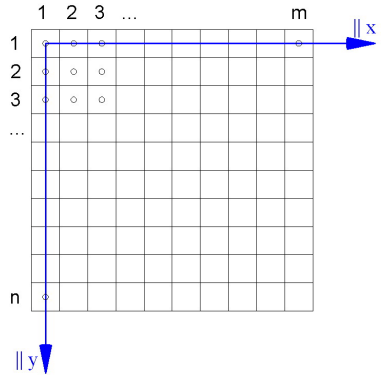


Abbildung 2.1: Digitales Bild-Koordinatensystem.

Die zwei wichtigsten Vertreter digitaler Bilder sind *Grauwert-* und *RGB-*

[1]Zu beachten ist: $(i, j) \in \mathbb{N}^2$, $(x, y) \in \mathbb{R}^2$.

Bilder (Farbbilder). In einem Grauwertbild gibt es pro Wertepaar (i, j) einen zugeordneten Funktionswert, den Grauwert, in RGB-Bildern je einen für die drei Kanäle Rot (R), Grün (G) und Blau (B). Ein solches Bildelement mit den Koordinaten (i, j) mit Funktionswert(en) wird meist als *Pixel* bezeichnet. Die Funktionswerte sind meist natürliche Zahlen im Intervall $[0, 255]$. In Grauwertbildern steht 0 für schwarz, 255 für weiß, in RGB-Bildern sind dies die Tripel $(0, 0, 0)$ bzw. $(255, 255, 255)$. Äquivalent zum RGB-Farbraum gibt es den *IHS*-Farbraum. *I* steht in diesem Zusammenhang für *Intensität (Intensity)*, *H* für *Helligkeit (Hue)* und *S* für *Sättigung (Saturation)*. Beide Farbräume sind umkehrbar eindeutig ineinander transformierbar, wobei die Intensität dem Grauwert des jeweiligen Pixels in einem Grauwertbild entsprechen würde. Die Transformation von RGB in den Grauwertbereich ist ebenfalls eindeutig, aber nicht umkehrbar.

Für weitergehende Informationen zu diesem Bereich sei auf [Kraus, 2004] und [Gonzalez et al., 2004] verwiesen.

2.2 Ausgewählte Operationen in digitalen Bildern

Dieser Abschnitt beschäftigt sich mit Filterungen in digitalen Bilder sowie mit der Erzeugung von Bildpyramiden.

2.2.1 Filterungen

Filterungen haben das Ziel, relevante Bildinhalte von nicht relevanten zu trennen. Filterungen sind sowohl im Orts- als auch im Frequenzbereich möglich, wobei in dieser Arbeit nur der Ortsbereich betrachtet wird.

Im Ortsbereich entspricht eine Filterung einer diskreten Faltung

$$\overline{\mathbf{G}} = \mathbf{G} * \mathbf{W}. \tag{2.1}$$

$\mathbf{G}$ bezeichnet hier das Ausgangsbild, $\overline{\mathbf{G}}$ das Ergebnisbild und $\mathbf{W}$ den sogenannten Faltungsoperator (Filter), der ebenfalls einer Matrix entspricht. Hat $\mathbf{W}$ die Dimension $2n + 1 \times 2m + 1$, so lässt sich die Faltung elementweise mit folgender Formel beschreiben:

$$\overline{\mathbf{G}}(x, y) = \sum_{k=-n}^{n} \sum_{l=-m}^{m} \mathbf{G}(x - k, y - l) \cdot \mathbf{W}(n + 1 + k, m + 1 + l). \tag{2.2}$$

Je nach gewünschtem Effekt werden unterschiedliche Filter verwendet. Im Folgenden seien zwei der wichtigsten vorgestellt: Der glättende *Gauß*-Filter und der kantenextrahierende *Laplace*-Filter.

Der Gaußfilter trägt seinen Namen aufgrund der Tatsache, dass seine Werte einer zweidimensionalen Gaußverteilung entsprechen. Allgemein

hat ein Gauß-Filter der Größe $(n \times m)$ folgende Gestalt:

$$\mathbf{G}(i,j) = \frac{1}{2\pi\sigma} e^{-\frac{(i-n/2)^2+(j-m/2)^2}{2\sigma^2}}. \tag{2.3}$$

Ein *Binomialfilter* stellt eine Annäherung an ein Gaußfilter dar. Ein 3×3-Binomialfilter hat folgende Werte:

$$\mathbf{W} = \frac{1}{16} \begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}. \tag{2.4}$$

Durch die gewichtete Einbeziehung der Nachbarpixel führt eine Faltung mit dem Gauß- bzw. Binomialfilter zur Dämpfung von hochfrequenten Effekten wie Rauschen, aber auch Kanten werden unterdrückt. Diese Eigenschaft besitzen auch alle anderen Filter, für die gilt:

$$\sum_{i=1}^{n} \sum_{j=1}^{n} \mathbf{W}(i,j) = 1. \tag{2.5}$$

(Grauwert-)Kanten stellen Extremwerte der zweiten Ableitungen dar, die sich in digitalen Bilder wie folgt annähern lassen [Kraus, 2004]:

$$\begin{aligned} \nabla^2\mathbf{G}(x,y) &= \mathbf{G}(x+1,y) + \mathbf{G}(x-1,y) + \mathbf{G}(x,y+1) + \\ &\quad \mathbf{G}(x,y-1) - 4\mathbf{G}(x,y) \end{aligned}$$

bzw.

$$\begin{aligned} \nabla^2\mathbf{G}(x,y) &= \mathbf{G}(x+1,y) + \mathbf{G}(x-1,y) + \mathbf{G}(x,y+1) + \\ &\quad \mathbf{G}(x,y-1) + \mathbf{G}(x-1,y-1) + \mathbf{G}(x+1,y+1) + \\ &\quad \mathbf{G}(x+1,y-1) + \mathbf{G}(x-1,y+1) - 8\mathbf{G}(x,y). \end{aligned} \tag{2.6}$$

Der zweite Teil von Formel 2.6 berücksichtigt auch die zweite Ableitung nach xy. Um nun durch Faltung die 2. Ableitung zu ermitteln, muss der Faltungsoperator folgende Gestalt besitzen:

$$\mathbf{W} = \begin{pmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{pmatrix} \text{ bzw. } \mathbf{W} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix} \tag{2.7}$$

Ein solcher Filter wird Laplace-Filter genannt. Wie bei allen anderen kantenextrahierenden Filtern ergeben seine Elemente $\mathbf{W}(i,j)$ in Summe 0 [Kraus, 2004].

2.2.2 Bildpyramiden

Reduziert man die Auflösung eines Ausgangsbildes der Dimension $(n \times m)$, indem man daraus ein Bild der Dimension $(\frac{n}{2} \times \frac{m}{2})$ erzeugt und wiederholt diesen Vorgang (mehrmals) mit dem Ergebnisbild, so spricht man von einer *Bildpyramide*. Dieser Vorgang ist in Abbildung 2.2 veranschaulicht. Eine Bildpyramide ermöglicht einen effizienten Umgang mit den Bilddaten (z.B. beim Zoomen). Die einfachste Methode, ein Bild des nächsthöheren Pyramidenniveau zu erhalten, ist das Streichen jeder zweiten Zeile und Spalte im Ausgangsbild. Allerdings geht bei dieser Methode relativ viel Information verloren, weshalb die Intensitäten der Pixel in den höheren Niveaus oft mittels Gauß- bzw. Binomialfilter ermittelt werden [Kraus, 2004].

Pro Pyramidenniveau wird der Bildmaßstab um die Hälfte kleiner. Dieser Umstand wird in Abschnitt 3.3 von Bedeutung sein.

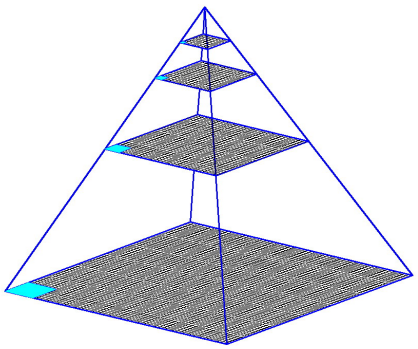


Abbildung 2.2: Schematische Darstellung einer Bildpyramide.

Kapitel 3

Interest-Operatoren

In diesem Kapitel werden die grundsätzlichen Eigenschaften von Interest-Operatoren und zwei bekannte Beispiele, nämlich der *Harris-Operator* (Abschnitt 3.2) und David Lowes *Keypoint Descriptor* (Abschnitt 3.3) ausführlicher vorgestellt.

3.1 Definition von Interest-Operatoren

Nach [Rodehorst, 2004] ist ein *Interest-Operator* ein „Verfahren zur Extraktion markanter Bildstellen, die in ihrer Umgebung möglichst einzigartig sind und mit hoher Wahrscheinlichkeit im korrespondierenden Bild in ähnlicher Weise abgebildet werden." Weiters sollen neben den entsprechenden Punkten auch Kenngrößen ermittelt werden, die eine Bildzuordnung – oder wie in diesem Fall eine Relative Orientierung der Bilder – erleichtern bzw. erst ermöglichen.

[Rodehorst, 2004] stellt an Interest-Operatoren folgende Anforderungen:

- Deutlichkeit,
- Invarianz,
- Robustheit,
- Seltenheit und
- Interpretierbarkeit.

Die Verfahren zur Ermittlung von Interest-Punkten lassen sich in drei Gruppen einteilen, nämlich in die Gruppe der *intensitätsbasierten* Ansätze, wo die Kenngröße direkt aus den Intensitätswerten ermittelt wird, der *konturbasierten* Ansätze, wo zuerst Konturen extrahiert werden und anschließend die Schnittpunkte berechnet werden, und die *modellbasierten* Ansätze, wo

durch Anpassung parametrischer Modelle eine Feindetektion im Subpixelbereich durchgeführt wird.

3.2 Der Harris-*Corner-und-Edge-Detector*

In diesem Abschnitt wird schematisch die Suche nach Interest-Punkten am Beispiel des bekannten *Corner and Edge Detector*[1] von Chris Harris and Mike Stephens [Harris und Stephens, 1988] dargestellt, der sowohl Kanten als auch Ecken (isolierte Punkte) in Grauwertbildern detektiert. Dieser Operator bildet auch eine methodische Grundlage für den *Key Point Descriptor* von David Lowe [Lowe, 1999, 2004], der in Abschnitt 3.3 vorgestellt wird und im Rahmen dieser Arbeit für die Bildzuordnung verwendet wurde.

Für jedes Pixel wird eine *Local Structure Matrix* $\mathbf{M}(x, y)$ berechnet [Liu, 2005]:

$$\mathbf{M}(x,y) = \begin{pmatrix} \left(\frac{\partial I(x,y)}{\partial x}\right)^2 & \frac{\partial I(x,y)}{\partial x}\frac{\partial I(x,y)}{\partial y} \\ \frac{\partial I(x,y)}{\partial x}\frac{\partial I(x,y)}{\partial y} & \left(\frac{\partial I(x,y)}{\partial y}\right)^2 \end{pmatrix}. \tag{3.1}$$

Hierbei bezeichnen $I(x, y)$ die Intensität, $\frac{\partial I(x,y)}{\partial x}$ und $\frac{\partial I(x,y)}{\partial y}$ die partiellen ersten Ableitungen der Intensität an der Stelle (x, y). Die Eigenwerte λ_1 und λ_2 von $\mathbf{M}$ sind proportional zu den Hauptkrümmungen jener Quadrik, die durch $\mathbf{M}(x, y)$ induziert wird. Ihre Werte sind invariant gegenüber Rotationen [Havlicek, 2003b]. Die Größenordnungen der Hauptkrümmungen werden in drei Fälle unterteilt:

1. beide Hauptkrümmungen sind klein ⇒ Pixel ohne signifikante Inhalte (Hintergrund)
2. eine Hauptkrümmung unterscheidet sich signifikant von der anderen ⇒ Pixel, das einer Kante angehört
3. beide Hauptkrümmungen sind groß ⇒ Pixel ist isolierter Punkt

Zusätzlich werden die Ecken- und Kantenpixel aufgrund einer Antwortfunktion $R(x, y)$

$$\begin{aligned} R(x,y) &= \det(\mathbf{M}(x,y)) - k \cdot \mathrm{tr}(\mathbf{M}(x,y)), \\ \mathrm{tr}(\mathbf{M}(x,y)) &= \lambda_1 + \lambda_2, \\ \det(\mathbf{M}(x,y)) &= \lambda_1 \lambda_2. \end{aligned} \tag{3.2}$$

noch in je zwei Klassen unterteilt (siehe Tabelle 3.1). Der Wert von k wird mit $0,04$ angenommen [Rodehorst, 2004; Liu, 2005]. Der Wert t_o bzw. t_u in Tabelle 3.1 bezeichnet den oberen bzw. unteren Schwellwert für die Klasseneinteilung.

[1] er wird in der Literatur auch oft als *Plessey*-Operator bezeichnet, nach *Plessey Research*, wo die beiden Autoren tätig waren.

Wert der Antwortfunktion	Klasse
$R(x,y) > t_o$	Ecke (Klasse 1)
$R(x,y) \leq t_o \wedge R(x,y) > t_u$	Ecke (Klasse 2)
$\lvert R(x,y)\rvert \leq t_u$	Hintergrund
$R(x,y) \leq -t_u \wedge R(x,y) > -t_o$	Kante (Klasse 2)
$R(x,y) < -t_o$	Kante (Klasse 1)

Tabelle 3.1: Klasseneinteilung der Antwortfunktion $R(x,y)$ des Harris-Operators

3.3 Der *Keypoint-Descriptor* von Lowe

Im Gegensatz zum Harris-Operator, der einen Interestpoint durch die Antwortfunktion R beschreibt, welche eine skalare Größe darstellt, ist der Keypoint-Descriptor [Lowe, 2004] eine vektorielle Größe. Er wird in folgenden Arbeitsschritten berechnet:

1. Extremwertsuche im *Scale-Space*
2. Lokalisierung der Keypoints
3. Zuweisung einer Orientierung der Keypoints
4. Berechnung der Keypoint-Descriptors

3.3.1 Extremwertsuche im Scale-Space

Der Scale-Space $L(x,y,\sigma)$ eines Bildes $I(x,y)$ wird in [Lowe, 2004] wie folgt definiert:

$$L(x,y,\sigma) = G(x,y,\sigma) * I(x,y). \tag{3.3}$$

$G(x,y,\sigma)$ ist der sogenannte *Scale-Space-Kernel*, die Variable σ wird als *Scale* bezeichnet. Der Scale-Space-Kernel wird als (diskrete) zweidimensionale Gauß-Funktion angenommen:

$$G(x,y,\sigma) = \frac{1}{2\pi\sigma} e^{-(x^2+y^2)/2\sigma^2}. \tag{3.4}$$

Um nicht durch Glättung zu viele mögliche Keypoints zu verlieren, wird die Bildgröße des Originalbilds vorab verdoppelt, wobei lineare Interpolation zur Berechnung der Intensitätwerte angewandt wird. Das ergibt einen theoretischen Zuwachs der Anzahl der Keypoints auf das Vierfache. Eine weitere Vergrößerung würde sich negativ auf die Rechenzeit auswirken, ohne dass nennenswert mehr Keypoints lokalisiert würden [Lowe, 2004].

Um in s Bildern die Extremwertsuche durchführen zu können, werden $(s+3)$ Bilder erzeugt. Eine solche Menge

$$\{L(x,y,\sigma), L(x,y,k\sigma), \ldots, L(x,y,2\sigma), L(x,y,2k\sigma), L(x,y,2k^2\sigma)\}$$

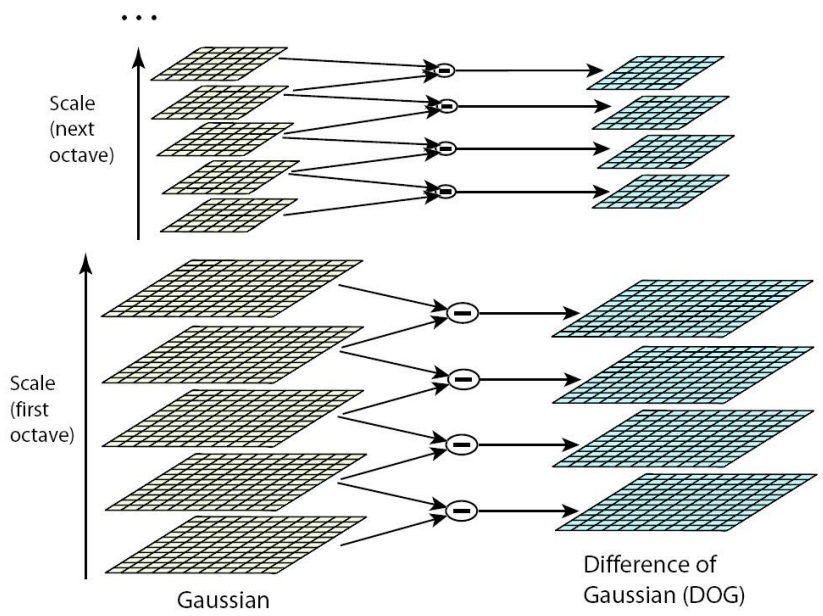


Abbildung 3.1: Der Scale-Space eines Bildes [Lowe, 2004].

wird als *Oktave* bezeichnet. Es ergibt sich $k = 2^{1/s}$.

Die Differenz zweier benachbarter Ebenen einer Oktave,

$$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y), \tag{3.5}$$

von Lowe als *Difference-of-Gaußian (DoG)* bezeichnet, ist eine Näherung des zweiten totalen Differentials, da für $k \approx 1$ gilt:

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}. \tag{3.6}$$

Daraus ergibt sich

$$D(x, y, \sigma) = G(x, y, k\sigma) - G(x, y, \sigma) \approx (k - 1)\sigma^2 \nabla^2 G.$$

Pro Oktave gibt es also $(s+2)$ DoG-Bilder. Die unterste Ebene der nächsten Oktave entsteht aus $L(x, y, 2\sigma)$ durch Streichen jeder zweiten Zeile und Spalte. Von diesem Punkt an wird die bisher beschriebene Prozedur wiederholt. Abbildung 3.1 zeigt ein schematisches Beispiel für einen Scale Space mit $s = 2$.

Potentielle Keypoints entsprechen lokalen Maxima und Minima in diesen DoG-Bildern. Die Suche erfolgt pixelweise in der 26er-Nachbarschaft eines Pixels, das sind die acht benachbarten Pixel im Scale des aktuellen Pixels und die jeweils neun entsprechenden Pixel im Scale darunter und

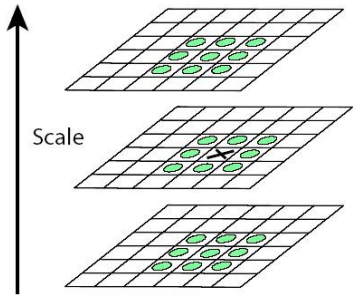


Abbildung 3.2: 26er-Nachbarschaft für lokale Extremwert-Ermittlung [Lowe, 2004].

darüber (siehe Abbildung 3.2). Die Extremwertsuche erfolgt nur in s DoG-Ebenen pro Oktave, weil in der untersten und der obersten DoG-Ebene die 26er-Nachbarschaft nicht erklärt ist. In [Lowe, 2004] wird empfohlen, $s = 3$ sowie im untersten Scale $\sigma = 1,6$ zu wählen, um die Rechenzeit und die Qualität der Ergebnisse zu optimieren.

3.3.2 Lokalisierung der Keypoints

Um einen Keypoint mit Subpixel-Genauigkeit zu orten, wird eine Paraboloidanpassung durchgeführt [Rodehorst, 2004]. Betrachtet wird hierfür ein lokales Maximum in einem DoG-Scale, die Pixel seiner Achter-Nachbarschaft und ihre Funktionswerte $D(x, y, \sigma)$. Diese Funktionswerte werden in das Intervall $[0; 1]$ normiert und das betrachtete Pixel in den Ursprung verschoben. Für das Paraboloid ergibt sich die Gleichung

$$D(x, y) = ax^2 + by^2 + cxy + dx + ey + f. \tag{3.7}$$

Werden die Parameter des Paraboloids zu einem Vektor $\mathbf{x} = (a, b, c, d, e, f)^{\mathrm{T}}$ zusammengefasst, so ergibt sich ein überbestimmtes lineares Gleichungssystem der Form $\mathbf{Ax} = \mathbf{b}$, wobei

$$\mathbf{A} = \begin{pmatrix} x_1^2 & y_1^2 & x_1y_1 & x_1 & y_1 & 1 \\ x_2^2 & y_2^2 & x_2y_2 & x_2 & y_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_9^2 & y_9^2 & x_9y_9 & x_9 & y_9 & 1 \end{pmatrix}, \mathbf{b} = \begin{pmatrix} D(x_1, y_1) \\ D(x_2, y_2) \\ \vdots \\ D(x_9, y_9) \end{pmatrix}.$$

Mit den klassischen Methoden der Ausgleichsrechnung [Niemeier, 2002] erhält man als Resultat $\mathbf{x} = (\mathbf{A}^{\mathrm{T}}\mathbf{A})^{-1}\mathbf{A}^{\mathrm{T}}\mathbf{b}$.

Die Verschiebungen des Paraboloid-Scheitels gegenüber dem zentralen Pixel ergeben sich zu [Rodehorst, 2004]

$$\Delta x = \frac{2bd - ce}{c^2 - 4ab} \quad \text{und} \quad \Delta y = \frac{2ae - cd}{c^2 - 4ab}. \tag{3.8}$$

Das lokale Extremum befindet sich somit an der Stelle

$$(\bar{x}, \bar{y}) = (x + \Delta x, y + \Delta y)$$

und hat den Funktionswert

$$D(\bar{x}, \bar{y}) = a\bar{x}^2 + b\bar{y}^2 + c\bar{x}\bar{y} + d\bar{x} + e\bar{y} + f.$$

Um keine allzu geringen Extrema als Keypoints zuzulassen, wird ein unterer Schwellwert von $D(\bar{x}, \bar{y}) = 0,03$ gesetzt. Weiters müssen noch die Punkte auf Kanten eliminiert werden. Diese stellen nämlich nur lokale Extrema normal zur Kante dar, nicht aber entlang der Kante und sind deshalb für die punktweise Bildzuordnung ungeeignet. Hierfür werden wie in Abschnitt 3.2 die zweiten partiellen Ableitungen betrachtet, die in der Hesse-Matrix

$$\mathbf{H} = \begin{pmatrix} \frac{\partial^2 D}{\partial x^2} & \frac{\partial^2 D}{\partial x \partial y} \\ \frac{\partial^2 D}{\partial x \partial y} & \frac{\partial^2 D}{\partial y^2} \end{pmatrix} \tag{3.9}$$

stehen. Alle betrachteten Pixel, für die

$$\frac{\text{tr}(\mathbf{H})^2}{\det(\mathbf{H})} > \frac{(r+1)^2}{r} \tag{3.10}$$

gilt, d.h. dass die Hauptkrümmungen an dieser Stelle zu sehr differieren, werden eliminert. In [Lowe, 2004] wird $r = 10$ gesetzt.

3.3.3 Zuweisung der Orientierung.

Für jeden auf diese Weise gefundenen Keypoint wird nun eine Magnitude m und eine Orientierung ϑ seines Gradienten ermittelt. Allgemein sind diese Größen folgendermaßen definiert [Lowe, 2004]:

$$m(x,y) = \begin{array}{c} \big((L(x+1,y) - L(x-1,y))^2 + \\ + (L(x,y+1) - L(x,y-1))^2\big)^{1/2}, \end{array} \tag{3.11}$$

$$\vartheta(x,y) = \arctan\frac{L(x,y+1) - L(x,y-1)}{L(x+1,y) - L(x-1,y)}. \tag{3.12}$$

Für jeden Keypoint wird in seiner Nachbarschaft (z.B. im umgebenden (6×6)-Fenster) für jedes Pixel je ein m und ein ϑ berechnet. Die Magnitude wird mit der zweidimensionalen Gauß-Funktion (siehe Formel (3.4)) multipliziert und in ein Orientierungshistogramm eingetragen. Das bedeutet, dass mit wachsendem Abstand vom Keypoint der Einfluss eines Pixels auf das Orientierungshistogramm des Keypoints geringer wird. Die Klassenbreite des Orientierungshistogramms beträgt $10°$.

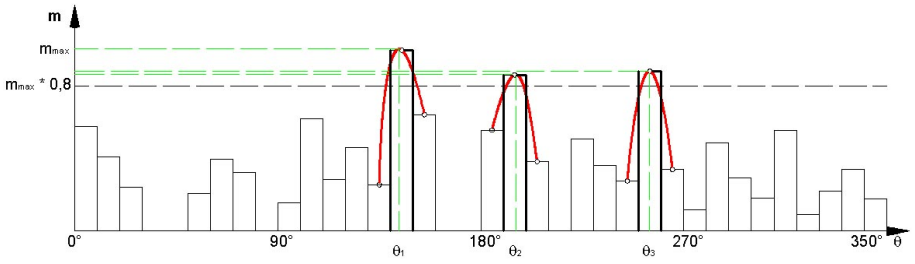


Abbildung 3.3: Orientierungshistogramm und Parabelanpassung der Orientierung(en) eines Keypoints. In diesem Beispiel entstehen an der gleichen Stelle im Bild drei Keypoints mit unterschiedlicher Orientierung.

Die Magnitude und die Orientierung des Keypoints werden nun über den maximalen Eintrag des Orientierungs-Histogramms und seine zwei benachbarten Werte über eine Parabelanpassung gefunden. Selbiges geschieht mit allen Einträgen des Orientierungs-Histogramms, die größer als 80% des Maximus sind (Abbildung 3.3). Gibt es solche Werte, so entsteht ein neuer Keypoint an der gleichen Stelle, aber mit unterschiedlicher Orientierung und Magnitude. Dieser Umstand trägt laut [Lowe, 2004] deutlich zur Stabilität des Verfahrens bei. Eine Darstellung der Keypoints eines Bildes mit Magnitude und Orientierung enthält Abbildung 3.4.

3.3.4 Berechnung des Keypoint-Descriptors

Für diesen Zweck wird ein (16×16)-Fenster um den Keypoint betrachtet. Dieses wird in 16 (4×4)-Bereiche unterteilt (vgl. Abbildung 3.6). Für jedes dieser Fenster wird ein Orientierungs-Histogramm ermittelt, ähnlich der obigen Vorgehensweise. Von jeder Orientierung $\vartheta_i$wird die oben errechnete Orientierung des Keypoints ϑ abgezogen, um eine möglichst hohe Rotationsinvarianz zu erreichen.

Die Klassenbreite beträgt in diesem Fall $45°$. Bezeichnen m_i und ϑ_i Magnitude bzw. Orientierung des aktuell betrachteten Pixels, so wird für dieses Pixel der Wert $\overline{m_i}$ nach folgender Formel in das Orientierungshistogramm eingetragen:

$$\overline{m_i} = m_i \frac{1}{2\pi\sigma} e^{-(x^2+y^2)/2\pi\sigma^2} \left(1 - \frac{|\Delta x'|}{4}\right)\left(1 - \frac{|\Delta y'|}{4}\right)\left(1 - \frac{|\Delta\vartheta|}{45}\right). \quad (3.13)$$

In obiger Formel wird $\sigma = 8$ gewählt, was der halben Fensterweite des Keypoint Descriptors entspricht [Lowe, 2004]. Der Wert $\Delta\vartheta$ ist die Differenz zwischen dem Mittelwert jener Klasse, in den ϑ_i fällt, also

$$\Delta\vartheta = (\vartheta_i - \vartheta) \mod 45.$$

Abbildung 3.4: Keypoints mit Magnitude und Orientierung.

Die weiteren Variablen in obiger Formel sind in Abbildung 3.5 dargestellt. Zusammengefasst ergeben die Orientierungshistogramme der 16 (4×4)-Subfenster einen 128-dimensionalen Vektor (16 Orientierungshistogramme a acht Klassen). Dieser ist invariant gegenüber

- Rotationen (aufgrund der Berücksichtigung der Orientierung des Keypoints),
- Maßstabsunterschieden (aufgrund der verschiedenen Scales und Oktaven) und
- linearen Beleuchtungsänderungen (denn diese Effekte fallen bei der DOG-Bildung heraus).

Um auch Einflüsse von nichtlinearen Beleuchtungsänderungen möglichst zu eliminieren, wird die Länge des Descriptors auf 1 normiert, alle Werte größer als $0,2$ auf $0,2$ gesetzt und dieser Vektor nochmals normiert. Dies erfolgt deshalb, weil nichtlineare Beleuchtungsänderungen sich vor allem in der Magnitude auswirken, nicht aber in der Orientierung [Lowe, 2004].

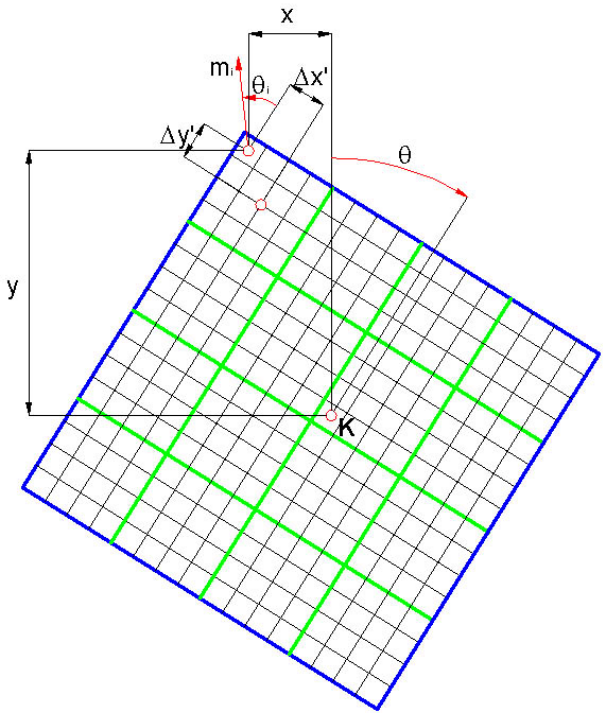


Abbildung 3.5: Grafik zur Visualisierung der Berechnung von $\overline{m_i}$ für ein Pixel aus der (16×16)-Umgebung eines Keypoints K.

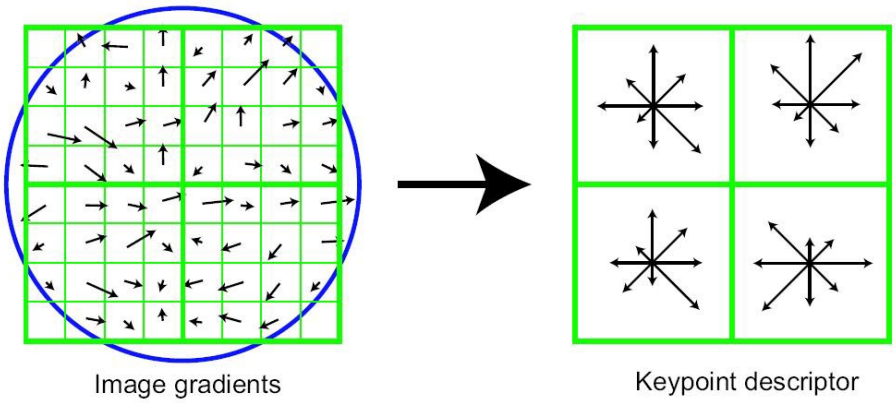


Abbildung 3.6: Beispiel für einen Keypoint Descriptor. Der Einfachheit halber wird ein (8×8)-Fenster um den Keypoint betrachtet, das in vier (4×4) Bereiche unterteilt wird, für die je ein Orientierungshistogramm mit $45°$ Klassenbreite erstellt wird. Für den tatsächlichen Keypoint Descriptor wird ein (16×16)-Fenster herangezogen, das in 16 (4×4)-Bereiche unterteilt wird. [Lowe, 2004].

3.3.5 Matching der Keypoints in zwei Bildern

Zum Auffinden korrespondierender (homologer) Punkte wird das Verfahren der *Nearest Neighbourhood* angewandt, es wird also zu jedem Keypoint Descriptor im ersten Bild jener Keypoint Descriptor im zweiten Bild gesucht, der die geringste euklidische Distanz zum ersten Keypoint Descriptor besitzt. In [Lowe, 2004] wird diese Suche nur in eine Richtung durchgeführt, d.h Mehrfach-Matchings eines Keypoints sind möglich. Dies ist für die in dieser Arbeit behandelte Bestimmung der Relativen Orientierung nicht zweckmäßig, daher wurde das Matching auch in die Gegenrichtung durchgeführt. Ein Punktepaar (P', P'') wurde nur dann als homolog akzeptiert, wenn es zu einem Keypoint Descriptor $des(P')$ aus Bild 1 keinen näheren Nachbarn als $des(P'')$ in Bild 2 gab und umgekehrt.

Dadurch wurden bereits viele potentielle Fehlkorrespondenzen eliminiert. Dieser aufgrund rein bildbasierter Information durchgeführte Vorgang wird in Abschnitt (5.3) noch um eine geometriebasierte robuste Schätzung ergänzt.

Kapitel 4

Kurze Einführung in die Projektive Geometrie

> „Die projektive Geometrie untersucht Eigenschaften geometrischer Gebilde, die durch Projektion nicht zerstört werden."
>
> *Josef Lense, österreichischer Mathematiker (1890-1985)*

4.1 Projektive Ebenen und Projektive Räume

In diesem Abschnitt werden kurz die Eigenschaften Projektiver Ebenen vorgestellt. Diese werden, wo zum Verständnis dieser Arbeit notwendig, auf Projektive Räume höherer Dimension erweitert.

Es sei $\mathcal{P}$ eine Punktmenge und $\mathcal{G}$ eine Geradenmenge. $(\mathcal{P}, \mathcal{G})$ heißt *projektive Ebene*, wenn gilt:

(PE 1) Zu je zwei verschiedenen Punkten gibt es genau eine Gerade, die beide Punkte enthält.

(PE 2) Je zwei verschiedene Geraden haben mindestens einen gemeinsamen Punkt.

(PE 3) Es gibt vier verschiedene Punkte, von denen nie drei gemeinsam einer Geraden angehören.

Die kleinstmögliche projektive Ebene enthält sieben Punkte, eine Darstellung davon enthält Abbildung 4.1. Auch die projektiv erweiterte Anschauungsebene (jede Gerade wird um einen uneigentlichen Punkt, den sog. *Fernpunkt* erweitert, die Ebene selbst um die Menge aller Fernpunkte, die *Ferngerade*) zählt zu den projektiven Ebenen [Havlicek, 2003a].

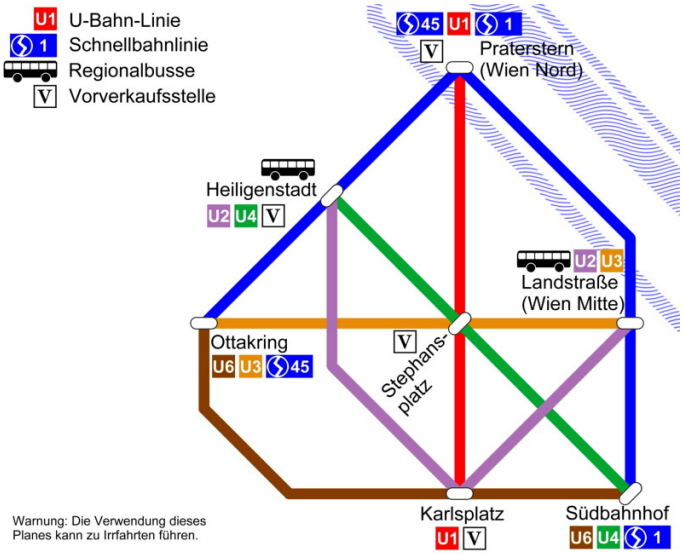


Abbildung 4.1: Projektive Minimalebene als Variante des Wiener U-Bahnnetzes (© by Hans Havlicek)

4.1.1 Dualität

In der projektiven Ebene $\mathcal{P}^2$ ist eine Gerade eine eindimensionale Punktmenge, ein Punkt ist nulldimensional und legt umkehrbar eindeutig eine Geradenmenge (*Geradenbüschel*) fest. Eine Aussage A^*, die durch Vertauschen der Worte *Punkt* und *Gerade* sowie durch das Umkehren von Inzidenzen ($\ni$) in einer Aussage A entsteht, nennt man die duale Aussage zu A.

Gilt eine Aussage in allen projektiven Ebenen, so gilt die dazu duale Aussage auch in allen projektiven Ebenen. In höherdimensionalen projektiven Räumen gilt Gleiches sinngemäß für Punkte und Hyperebenen statt für Punkte und Geraden [Havlicek, 2003a; Ressl, 2003; Hartley und Zisserman, 2001].

4.1.2 Koordinatendarstellung

Wie bereits angesprochen, werden in der Projektiven Geometrie Geraden und Ebenen um je ein „unsichtbares“ Element (Fernpunkt, Ferngerade) erweitert. Diese Elemente lassen sich in euklidischen Koordinaten nicht darstellen, daher führt man sogenannte *homogene Koordinaten* ein, welche nun bis auf einen gemeinsamen Faktor ident sind. Deshalb ist statt dem gewohnten „=“ das Symbol „$\sim$“ (steht für *ist proportional zu*) in allen Berechnungen zu finden, die homogene Koordinaten beinhalten. Eine Veranschaulichung der homogenen Koordinaten ist in Abbildung 4.2 dargestellt.

In der Ebene wird nun ein Punkt X durch einen dreidimensionalen Spaltenvektor $\mathbf{x}$ bzw. durch seine Vielfachen repräsentiert:

$$\mathbf{x} \sim \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} u \\ v \\ w \end{pmatrix} \sim \begin{pmatrix} \mathbf{x}_O \\ x_H \end{pmatrix}. \tag{4.1}$$

Ähnliches gilt für Geraden in der Ebene: Eine Gerade λ besitzt einen dreidimensionalen Spaltenvektor $\boldsymbol{\lambda}$ bzw. seine Vielfachen als Repräsentant:

$$\boldsymbol{\lambda} \sim \begin{pmatrix} a \\ b \\ c \end{pmatrix} \sim \begin{pmatrix} \boldsymbol{\lambda}_H \\ \lambda_O \end{pmatrix}. \tag{4.2}$$

Die Indices $_O$ bzw. $_H$ repräsentieren den euklidischen resp. homogenen Anteil der Vektoren, der homogene Anteil ist bei uneigentlichen Elementen Null. Dies bedeutet, dass man in homogenen Koordinaten auch mit Fernpunkten rechnen kann.Ein weiterer Vorteil der homogenen Koordinatendarstellung ist, dass sich Translationen als Matrizenmultiplikationen anschreiben lassen. [Ressl, 2003].

Ein Punkt $P \sim \mathbf{p}$ liegt dann auf einer Geraden $\lambda \sim \boldsymbol{\lambda}$, wenn gilt:

$$\mathbf{p}^{\mathrm{T}}\boldsymbol{\lambda} = 0. \tag{4.3}$$

Wird eine Gerade λ durch zwei Punkte $A \sim \mathbf{a}$ und $B \sim \mathbf{b}$ aufgespannt, so ergibt sich ihr Repräsentant $\boldsymbol{\lambda}$ zu

$$\boldsymbol{\lambda} \sim \mathbf{a} \times \mathbf{b} = \mathbf{S}(\mathbf{a})\mathbf{b}. \tag{4.4}$$

Die Matrix $\mathbf{S}$ heißt auch *Axiator* und repräsentiert das Kreuzprodukt zweier Vektoren $\mathbf{a} = (u, v, w)^{\mathrm{T}}$ und $\mathbf{b} = (x, y, z)^{\mathrm{T}}$ [Ressl, 2003]:

$$\mathbf{a} \times \mathbf{b} = \mathbf{S}(\mathbf{a})\mathbf{b} = \begin{pmatrix} 0 & -w & v \\ w & 0 & -u \\ -v & u & 0 \end{pmatrix} \mathbf{b} = -\mathbf{S}(\mathbf{b})\mathbf{a}. \tag{4.5}$$

Der Schnittpunkt $P \sim \mathbf{p}$ zweier Geraden $\alpha \sim \boldsymbol{\alpha}$ und $\beta \sim \boldsymbol{\beta}$ lässt sich auf die gleiche Weise finden, da die Begriffe Punkt und Gerade in der Ebene dual zueinander sind, wie bereits erwähnt:

$$\mathbf{p} \sim \boldsymbol{\alpha} \times \boldsymbol{\beta} = \mathbf{S}(\boldsymbol{\alpha})\boldsymbol{\beta}. \tag{4.6}$$

Geht man auf allgemeine projektive Räume $\mathcal{P}$ der Dimension n über, so lassen sich Punkte X und *Hyperebenen* λ jeweils durch $(n+1)$-dimensionale Vektoren $\mathbf{x}$ bzw. $\boldsymbol{\lambda}$ repräsentieren. Um die Bedeutung des Begriffes Hyperebene klarzulegen, sei auf Tabelle 4.1 hingewiesen.

Dim. $\mathcal{P}$	Bezeichnung für $\mathcal{P}$	Dim. λ	Bezeichnung für λ
1	Gerade	0	Punkt
2	Ebene	1	Gerade
3	„3-Raum“	2	Ebene
n	n-dim. Raum	$(n-1)$	Hyperebene der Dim. $(n-1)$

Tabelle 4.1: Hyperebenen λ in projektiven Räumen $\mathcal{P}$ verschiedener Dimension n.

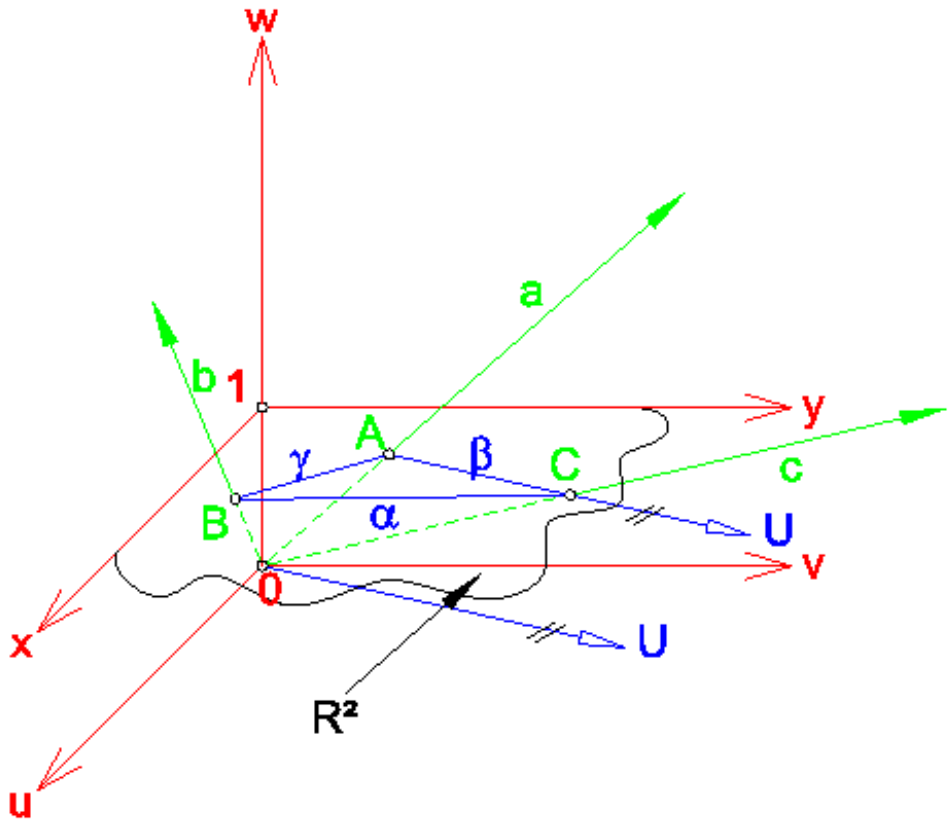


Abbildung 4.2: Die projektiv abgeschlossene Anschauungsebene $\mathcal{P}^2$ in den $\mathbb{R}^3$ eingebettet [Ressl, 2003].

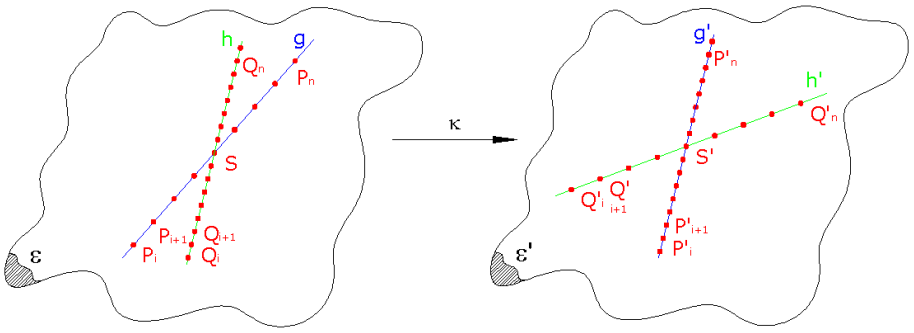


Abbildung 4.3: Kollineation (Homographie) $\kappa : \varepsilon \rightarrow \varepsilon'$ in der Ebene. Kollineare Punkte gehen unter κ in kollineare Punkte über.

4.2 Kollineationen und Korrelationen

Kollineationen. Sind $(\mathcal{P}, \mathcal{G})$ und $(\mathcal{P}', \mathcal{G}')$ projektive Ebenen, so heißt eine Abbildung $\kappa : \mathcal{P} \rightarrow \mathcal{P}'$ eine Kollineation (auch Homographie), wenn gilt:

(K 1) κ ist bijektiv.

(K 2) Je drei kollineare Punkte gehen unter κ in kollineare Punkte über.

(K 3) Je drei nicht kollineare Punkte gehen unter κ in nicht kollineare Punkte über.

Es lässt sich zeigen, dass bereits **(K 1)** und **(K 2)** eine Kollineation ausreichend beschreiben [Havlicek, 2003a]. Ein Beispiel für eine Kollineation stellt Abbildung 4.3 dar. Die obigen Aussagen gelten sinngemäß auch für höherdimensionale projektive Räume. Im $\mathcal{P}^3$ übernehmen Ebenen die Rolle der Geraden, in einem allgemeinen n-dimensionalen Raum sind es die $(n-1)$-dimensionalen Unterräume, sogenannte *Hyperebenen*.

Korrelationen. Eine Abbildung κ^* eines projektiven Raumes $\mathcal{P} \rightarrow \mathcal{P}'$ heißt Korrelation (oder duale Kollineation), wenn κ^* bijektiv ist und kollineare Punkte auf kopunktale Hyperebenen abgebildet werden [Ressl, 2003]. In $\mathcal{P}^2$ wird also ein Punkt auf eine Gerade, in $\mathcal{P}^3$ auf eine Ebene abgebildet. Eine Korrelation in $\mathcal{P}^2$ ist in Abbildung 4.4 abgebildet.

4.3 Kegelschnitte und Quadriken

In einem n-dimensionalen projektiven Raum $(\mathcal{P}, \mathcal{G})$ wird eine Quadrik Q durch eine symmetrische Matrix $\mathbf{Q}$ der Dimension $(n+1 \times n+1)$ beschrieben. Für alle Punkte $X \sim \mathbf{x}$, die auf der Quadrik liegen, gilt [Havlicek,

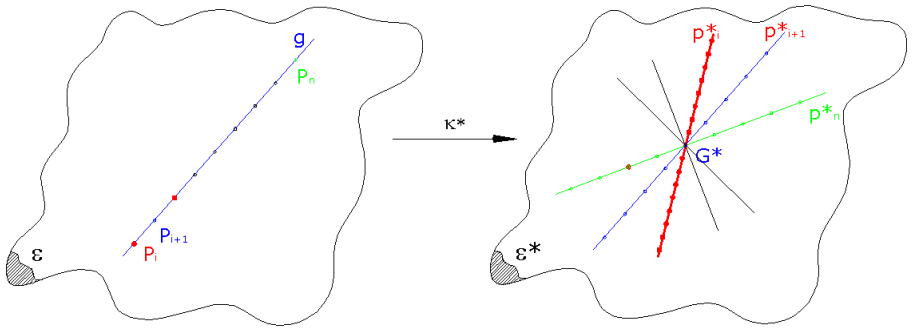


Abbildung 4.4: Korrelation (duale Kollineation) $\kappa^* : \varepsilon \rightarrow \varepsilon^*$ in der Ebene. Kollineare Punkte gehen dabei in kopunktale Geraden über.

2003a; Hartley und Zisserman, 2001]:

$$\mathbf{x}^{\mathrm{T}}\mathbf{Q}\mathbf{x} = 0. \tag{4.7}$$

Neben den *Punktquadriken* gibt es auch *duale Quadriken*, deren Elemente Hyperebenen sind. Die duale Quadrik Q^* zu einer Punktquadrik wird durch die Matrix $\mathbf{Q}^{-1}$ induziert, falls Q nicht singulär ist. Jene Hyperebenen $\varepsilon \sim \boldsymbol{\varepsilon}$, die Elemente von Q^* sind, erfüllen die Gleichung

$$\boldsymbol{\varepsilon}^{\mathrm{T}}\mathbf{Q}^{-1}\boldsymbol{\varepsilon} = 0. \tag{4.8}$$

Kegelschnitte sind nichts anderes als Quadriken in der projektiven Ebene. Ein *Punktkegelschnitt* ω wird folglich durch eine symmetrische (3×3)-Matrix

$$\omega \sim \begin{pmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{12} & \omega_{22} & \omega_{23} \\ \omega_{31} & \omega_{32} & \omega_{33} \end{pmatrix} \tag{4.9}$$

induziert, der entsprechende duale *Geradenkegelschnitt* durch ω^{-1}.

In Photogrammetrie und Computer Vision kommt der sogenannten *absoluten Quadrik* Q_∞ eine gewichtige Rolle zu, da sie für die Selbstkalibrierung benützt werden kann (siehe Abschnitt 5). Ihre Gleichung lautet in kanonischer Form im $\mathcal{P}^3$: $x_1^2 + x_2^2 + x_3^2 = 0$, ihre induzierende Matrix daher

$$\mathbf{Q}_{\mathrm{M}}^* \sim \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}.$$

Ihr Schnitt mit der Fernebene π_∞ ist der sogenannte *absolute Kegelschnitt* ω_∞. Wie man leicht sieht, enthalten sowohl Q_∞ als auch ω_∞ keine reellen Punkte.

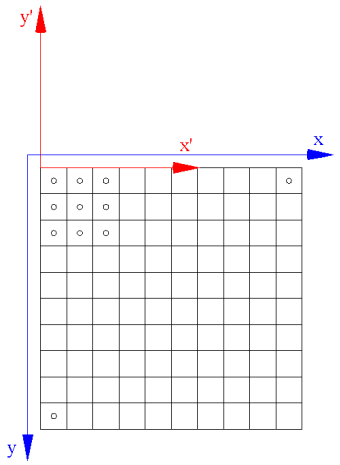


Abbildung 4.5: Wechsel des Bildkoordinatensystems vom Linkssystem (x, y) auf das Rechtssystem (x', y').

4.4 Perspektive in der Projektiven Geometrie

Vor Beginn der weiteren Betrachtungen sei ein kleiner Absatz über die ab hier verwendeten Bildkoordinaten eingeschoben: In Abschnitt 2.1 wurde das Bildkoordinatensystem als *Linkssystem* definiert, um der Konvention der verwendeten Software MATLAB zu folgen, wo Bilder als Matrizen betrachtet werden und die Pixel mit Zeilen- und Spaltenindex angesprochen werden. In der Folge werden nun – um Spiegelungen zu vermeiden – kartesische *Rechtskoordinaten* der Form (x', y') verwendet, wobei gilt:

$$\begin{aligned} x' &= x - 0.5, \\ y' &= -y - 0.5. \end{aligned} \tag{4.10}$$

Der Ursprung dieses System liegt in der linken oberen Ecke des linken oberen Pixels, die positive x'-Achse weist nach rechts, die positive y'-Achse weist nach oben, wie in Abbildung 4.5 ersichtlich. In einem Bild treten nur positive x'-Koordinaten und nur negative y'-Koordinaten auf.

Das *Pinhole Camera Model*. Als Ausgangspunkt zur Betrachtung dieser Thematik stelle man sich vor, das Projektionszentrum Z befinde sich im Ursprung, die z-Achse verlaufe durch den Hauptpunkt und die Achsen des Kamera-Koordinatensystems seien parallel zu denen des Objektkoordinatensystems (siehe Abbildung 4.6). Der Normalabstand des Projektionszentrums von der Bildebene (Kamerakonstante) sei c. Hat nun ein Objektpunkt X die euklidischen Koordinaten $(x, y, z)^{\mathrm{T}}$ und damit die homogenen Koordinaten $\mathbf{x} \sim (x, y, z, 1)^{\mathrm{T}}$, so folgt aus dem Strahlensatz für seinen Bildpunkt

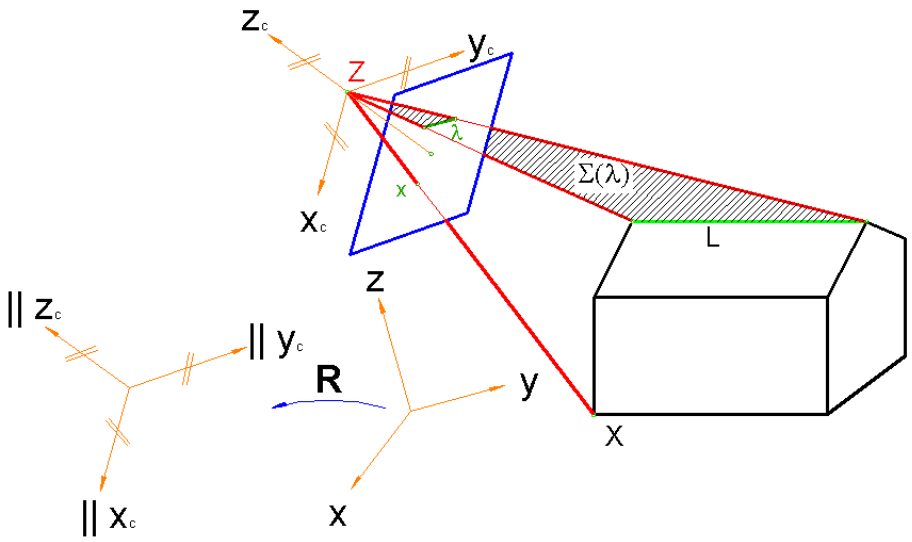


Abbildung 4.6: Abbildung von Punkten und Geraden unter einer Perspektive mit Projektionszentrum Z. Die Rotationsmatrix $\mathbf{R}$ transformiert vom Objekt- in das Kamerakoordinatensystem.

in inhomogenen Koordinaten:

$$X' \sim \mathbf{x}' = \left(c\frac{x}{z}, c\frac{y}{z}, c\right)^{\mathrm{T}} \tag{4.11}$$

Aus der oben genannten Lagerung des Koordinatensystems folgt, dass dies auch die homogenen Koordinaten des Bildpunktes im Kamerakoordinatensystem sind:

$$\mathbf{x}' \sim \left(c\frac{x}{z}, c\frac{y}{z}, c\right)^{\mathrm{T}} \sim (x, y, z)^{\mathrm{T}}.$$

Die Projektion, die $X \mapsto X'$ leistet, lässt sich in Matrixschreibweise wie folgt anschreiben:

$$\mathbf{x}' \sim \left(\begin{array}{ccc|c} 1 & & & 0 \\ & 1 & & 0 \\ & & 1 & 0 \end{array}\right) \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix} \sim (\mathbf{I}|\mathbf{0})\mathbf{x} \sim \mathbf{P}\mathbf{x}. \tag{4.12}$$

Im Allgemeinen befindet sich das Projektionszentrum an der Position $Z \sim \mathbf{z}$, die Orientierung des Kamerakoordinatensystems in Bezug auf das Objektkoordinatensystem ist durch eine Rotationsmatrix $\mathbf{R}$ gegeben. Weiters weicht die innere Orientierung von einer idealen Perspektive ab und

wird durch die sogenannte Kalibrierungsmatrix

$$\mathbf{C} = \begin{pmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{pmatrix} \tag{4.13}$$

beschrieben. In obiger Gleichung sind

- α_x bzw. α_y die Kamerakonstante in Pixelbreite bzw. höhe,
- s der Scherungsfaktor und
- x_0, y_0 die Koordinaten des Bildhauptpunktes.

Das Seitenverhältnis der Pixel ergibt sich zu $\gamma = \frac{\alpha_y}{\alpha_x}$. Die Projektionsmatrix $\mathbf{P}$ lautet also in ihrer allgemeinen Form folgendermaßen:

$$\mathbf{P} \sim \mathbf{CR}(\mathbf{I}| - \mathbf{z}_O). \tag{4.14}$$

In obiger Gleichung ist $\mathbf{z}_O$ der euklidische Anteil der homogenen Koordinaten von Z im Objektkoordinatensystem.

Da die Matrix $\mathbf{P}$ nur bis auf den Maßstab bestimmbar ist, besitzt sie 11 Freiheitsgrade [Ressl, 2003; Kraus, 1997].

Kapitel 5

Projektive Relative Orientierung und Selbst-Kalibrierung

Zwei Bilder gelten als relativ orientiert, wenn sich die Projektionsstrahlen korrespondierender Bildpunkte jeweils im Raum schneiden [Kraus, 2004].

Handelt es sich bei den relativ zu orientierenden Bildern um kalibrierte Aufnahmen, so lässt sich dieses Problem als räumliche Ähnlichkeitstransformation ausdrücken. Bei zwei Bildern hat diese Ähnlichkeitstransformation 5 Freiheitsgrade, allgemein bei n Bildern $(6n - 7)$ Freiheitsgrade. Im Falle von unkalibrierten Bildern handelt es sich bei der Relativen Orientierung um eine Projektivtransformation, da neben der Ähnlichkeitstransformation noch eine Kollineation in der Bildebene zu berücksichtigen ist. Diese Projektivtransformation hat 7 Freiheitsgrade bei zwei Bildern, bei n Bildern $(11n - 15)$. Der Beweis hierfür ist u. a. in [Ressl, 2003] nachzulesen. Der in dieser Arbeit verwendete Ansatz der Relativen Orientierung basiert auf dem sogenannten *Folgebildanschluss* [Kraus, 2004]: Das Projektionszentrum eines Bildes wird o. B. d. A. im Ursprung angenommen, die Achsen des Bildkoordinatensystems sind gegenüber den Achsen des Objektkoordinatensystems nicht verdreht. Das zweite Bild wird nun gegenüber dem ersten verdreht und verschoben, bis sich die korrespondierenden Projektionsstrahlen schneiden. Das Objektmodell bleibt davon unberührt.

Es gilt also für das erste Projektionszentrum $Z_1 \sim \mathbf{z}_1 = \mathbf{0}$ und für die zugehörige Rotationsmatrix $\mathbf{R}_1 \sim \mathbf{I}$.

5.1 Die Fundamentalmatrix $\mathbf{F}_{12}$

Sind zwei Bilder relativ zueinander orientiert, so spannen die Verbindungsgerade der beiden Projektionszentren Z_1 und Z_2 (auch *Basisgerade* genannt)

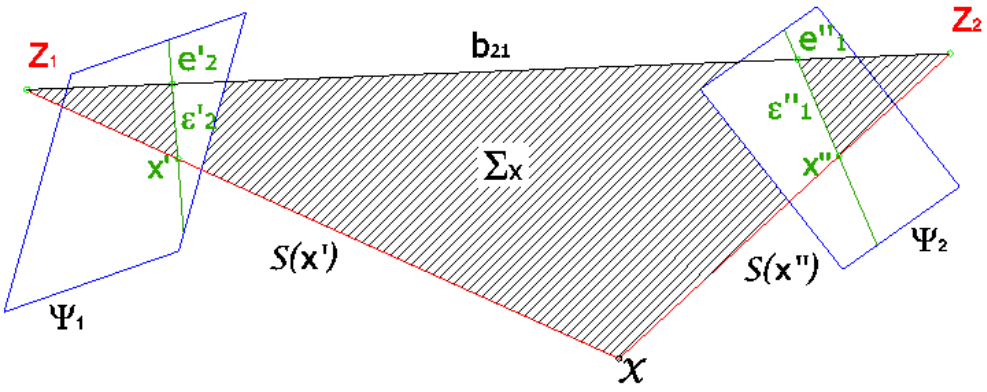


Abbildung 5.1: Relative Orientierung und Epipolarbedingung für zwei Bilder.

und ein Bildpunkt $X' \sim \mathbf{x}'$ eine Ebene auf. In dieser Ebene liegen sowohl der Objektpunkt X als auch der entsprechende Bildpunkt $X'' \sim \mathbf{x}''$ im zweiten Bild (siehe Abbildung 5.1). Jede solche Ebene schneidet aus dem zweiten Bild eine (Halb-)Gerade aus, den sogenannten *Kernstrahl* oder *Epipolarlinie*. Diese Abbildung Punkt $\mapsto$ Gerade stellt eine Korrelation dar (siehe Kapitel 4.2) und wird durch eine (3×3)-Matrix $\mathbf{F}_{12}$ induziert. Da der zweite Bildpunkt auf dieser Geraden liegt, gilt:

$$\mathbf{x}'^{\mathrm{T}} \mathbf{F}_{12} \mathbf{x}'' = 0. \tag{5.1}$$

Die Matrix $\mathbf{F}_{12}$ wird *Fundamentalmatrix* genannt. Aus Gleichung (5.1) ist leicht ersichtlich, dass $\mathbf{F}_{21} = \mathbf{F}_{12}^{\mathrm{T}}$ gilt, wenn der gesamte Ausdruck transponiert wird. $\mathbf{F}_{12}$ lässt sich geometrisch in folgender Weise deuten:

$$\mathbf{F}_{12} \sim \mathbf{C}_1^{-\mathrm{T}} \mathbf{R}_1 \mathbf{S}(\mathbf{z_2} - \mathbf{z_1}) \mathbf{R}_2^{\mathrm{T}} \mathbf{C}_2^{-1}. \tag{5.2}$$

Setzt man Gleichung (5.2) in (5.1) ein, so ergibt sich

$$\mathbf{x}'^{\mathrm{T}} \mathbf{C}_1^{-\mathrm{T}} \mathbf{R}_1 \mathbf{S}(\mathbf{z_2} - \mathbf{z_1}) \mathbf{R}_2^{\mathrm{T}} \mathbf{C}_2^{-1} \mathbf{x}'' = 0.$$

Hierbei steht der Ausdruck $\mathbf{x}'^{\mathrm{T}} \mathbf{C}_1^{-\mathrm{T}} \mathbf{R}_1$ für den Richtungsvektor des ersten Projektionsstrahles, $\mathbf{S}(\mathbf{z_2} - \mathbf{z_1})$ für das Kreuzprodukt mit dem Basisvektor und $\mathbf{R}_2^{\mathrm{T}} \mathbf{C}_2 \mathbf{x}''$ für den Richtungsvektor des zweiten Projektionsstrahles. Die ersten zwei Ausdrücke spannen die bereits angesprochene Ebene auf, in der auch der zweite Punkt liegt. Das Skalarprodukt ergibt also Null. Ein ausführlicher Beweis für Formel (5.2) findet sich in [Ressl, 2003].

Gleichung (5.2) lässt sich zu

$$\mathbf{F}_{12} \sim \mathbf{C}_1^{-\mathrm{T}} \mathbf{S}(-\mathbf{z_2}) \mathbf{R}_2^{\mathrm{T}} \mathbf{C}_2^{-1} \tag{5.3}$$

vereinfachen, wenn das erste Projektionszentrum im Ursprung liegt. Eine weitere interessante Eigenschaft der Fundamentalmatrix zeigt sich bei

Betrachtung der Kernpunkte $E_i^{(j)} \sim \mathbf{e}_i^{(j)}$. Ein *Kernpunkt* (auch *Epipol* genannt) ist der Durchstoßpunkt der Basisgeraden mit einer Bildebene, also das jeweilige Abbild des Projektionszentrums Z_i im Bild j. Da das Projektionszentrum selbst in der Perspektive auf keinen Punkt abgebildet wird, liefert die Gleichung der perspektivischen Abbildung den Nullvektor (vgl. Gleichung 4.14). Gleiches muss für den Kernpunkt $E_i^{(j)}$ unter der Korrelation gelten, die durch die Fundamentalmatrix $\mathbf{F}_{ij}$ induziert wird, also erhält man

$$\mathbf{F}_{12}\mathbf{e}_1'' = \mathbf{0} \text{ bzw. } \mathbf{F}_{12}^{\mathrm{T}}\mathbf{e}_2' = \mathbf{0}. \tag{5.4}$$

Die Kernpunkte stellen den links- bzw. rechtsseitigen Nullraum der $\mathbf{F}_{12}$-Matrix dar, also den *Kern* der durch sie induzierten Abbildung, wodurch sich auch der Name *Kernpunkte* erklären lässt. Die *Kernstrahlen* $\varepsilon_2' \sim \boldsymbol{\varepsilon}_2'$ und $\varepsilon_1'' \sim \boldsymbol{\varepsilon}_1''$lassen sich mit Hilfe der Fundamentalmatrix folgendermaßen darstellen:

$$\boldsymbol{\varepsilon}_2' \sim \mathbf{F}_{12}\mathbf{x}'' \quad \text{bzw.} \quad \boldsymbol{\varepsilon}_1'' \sim \mathbf{F}_{12}^{\mathrm{T}}\mathbf{x}' \tag{5.5}$$

Da alle Kernstrahlen durch den Kernpunkt verlaufen, ist die Matrix $\mathbf{F}_{12}$ singulär [Ressl, 2003; Hartley und Zisserman, 2001].

5.2 Berechnung der Fundamentalmatrix

Im folgenden Abschnitt werden zwei der gängigsten Methoden zur Berechnund der Fundamentalmatrix vorgestellt. In Abschnitt 5.3 wird die schließlich implementierte Methode, eine modifizierte Version von 5.2.1, zur robusten Schätzung von $\mathbf{F}_{12}$ präsentiert.

5.2.1 Linearer Normierter 8-Punkte-Algorithmus

Gegeben seien $n \geq 8$ Punktkorrespondenzen $(\mathbf{x}_i', \mathbf{x}_i'')$. Gesucht ist eine Matrix $\mathbf{F}_{12}$, sodass die Bedingung

$$\mathbf{x}_i'^{\mathrm{T}}\mathbf{F}_{12}\mathbf{x}_i'' = 0$$

erfüllt ist.

Im ersten Schritt werden die Bildkoordinaten $\mathbf{x}_i^{(j)}$ Bilder *normalisiert*, indem der Schwerpunkt der beobachteten Punkte jeweils in den Urpsrung verschoben wird und die mittlere Distanz der Punkte zum Schwerpunkt auf $\sqrt{2}$ gesetzt wird [Hartley und Zisserman, 2001]. Diese Transformationen lassen sich in Matrizenform durch Matrizen $\mathbf{T}_j$ anschreiben:

$$\hat{\mathbf{x}}_i^{(j)} \sim \mathbf{T}_j\mathbf{x}_i^{(j)}. \tag{5.6}$$

Der Grund für die Normalisierung der Bildkoordinaten ist, dass in digitalen Bildern die Koordinaten zwischen 1 und mehr als 1.000 variieren

können. Die Elemente von $\mathbf{A}$ hätten dann also Werte von 1^2 bis über 1.000^2, was zu nummerischen Schwierigkeiten führt [Ressl, 2003].

Enthält ein Vektor $\hat{\mathbf{f}} = (\hat{f}_{11}, \hat{f}_{12}, \hat{f}_{13}, \hat{f}_{21}, \hat{f}_{22}, \hat{f}_{23}, \hat{f}_{31}, \hat{f}_{32}, \hat{f}_{33})^{\mathrm{T}}$ die Elemente der Matrix $\hat{\mathbf{F}}_{12}$, so kann ein lineares Gleichungssystem $\hat{\mathbf{A}}\hat{\mathbf{f}} = \mathbf{0}$ angesetzt werden mit

$$\hat{\mathbf{A}} = \begin{pmatrix} \hat{x}'_1\hat{x}''_1 & \hat{x}'_1\hat{y}''_1 & \hat{x}'_1 & \hat{y}'_1\hat{x}''_1 & \hat{y}'_1\hat{y}''_1 & \hat{y}'_1 & \hat{x}''_1 & \hat{y}''_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \hat{x}'_n\hat{x}''_n & \hat{x}'_n\hat{y}''_n & \hat{x}'_n & \hat{y}'_n\hat{x}''_n & \hat{y}'_n\hat{y}''_n & \hat{y}'_n & \hat{x}''_n & \hat{y}''_n & 1 \end{pmatrix} \hat{\mathbf{f}} = \mathbf{0}. \quad (5.7)$$

Die Matrix $\hat{\mathbf{F}}_{12}$ lässt sich nun aus dem zum kleinsten Singulärwert von $\hat{\mathbf{A}}$ gehörigen Vektor bestimmen [Hartley und Zisserman, 2001].

Nun muss noch die Singularitätsbedingung $\det(\hat{\mathbf{F}}_{12}) = 0$ berücksichtigt werden. Dies wird erreicht, indem der kleinste Singulärwert von $\hat{\mathbf{F}}_{12}$ auf 0 gesetzt wird. Die so erhaltene Matrix $\hat{\mathbf{F}}'_{12}$ muss nun noch *denormalisiert* werden. Somit ergibt sich

$$\mathbf{F}_{12} = \mathbf{T}_1^{\mathrm{T}}\hat{\mathbf{F}}'_{12}\mathbf{T}_2. \quad (5.8)$$

5.2.2 Minimaler 7-Punkte-Algorithmus

Nimmt man von Anfang an die Singularitätsbedingung als zusätzliche Information hinzu, so reichen sieben homologe Punktepaare $\mathbf{x}_i^{(j)}$ aus, um die Fundamentalmatrix zu bestimmen. Da es sich hierbei aber um keine lineare, sondern eine kubische Bedingung in den Elementen von $\mathbf{F}_{12}$ handelt, ergeben sich bis zu drei Lösungen [Ressl, 2003; Rodehorst, 2004].

Die Matrix $\mathbf{A}$ aus Gleichung (5.7) hat nun die Dimension (7×9). Die Lösung des Gleichungssystems ist der *zweiparametrige rechte Nullraum* von $\mathbf{A}$. Es seien nun $\mathbf{u}$ und $\mathbf{v}$ die zum Singulärwert 0 gehörenden rechten Nullvektoren von $\mathbf{A}$, $\mathbf{U}$ und $\mathbf{V}$ ihre äquivalenten (3×3)-Matrizen. Die Fundamentalmatrix ist nun als

$$\mathbf{F}_{12} = t\mathbf{U} + (1-t)\mathbf{V} \quad (5.9)$$

darstellbar, der Skalar t ergibt sich aus der Singularitätsbedingung

$$\det(\mathbf{F}_{12}) = \det(t\mathbf{U} + (1-t)\mathbf{V}) = 0, \quad (5.10)$$

woraus ein Polynom $p(t)$ der Form

$$p(t) = at^3 + bt^2 + ct + d = 0 \quad (5.11)$$

folgt. Die Koeffizienten dieses kubischen Polynoms sind in [Rodehorst, 2004] nachzulesen. Es können sich eine bzw. drei reelle Lösungen ergeben.

Eine ausführliche Darstellung verschiedener Algorithmen zur Berechnung der Fundamentalmatrix findet sich in [Hartley und Zisserman, 2001].

Hinzuzufügen ist noch, dass eine eindeutige Bestimmung der Fundamentalmatrix nicht möglich ist, wenn sich beide Projektionszentren und alle in Betracht gezogenen Objektpunkte auf einer Regelquadrik befinden, also auf einem einschaligen Hyperboloid, einer HP-Fläche, einem Kegel, einem Zylinder oder auf zwei Ebenen. In diesem Fall ergeben sich bis zu drei Lösungen, unabhängig von der Anzahl der homologen Punktepaare [Ressl, 2003].

5.3 Robuste Schätzung der Fundamentalmatrix

Robuste Schätzverfahren haben zum Ziel, „qualitativ gute Schätzungen der zu bestimmenden Parameter“ zu liefern, auch wenn die „Modellannahme der kleinsten Quadrate nicht erfüllt“ sind [Niemeier, 2002]. Ein robuster Schätzer sollte folgende Anforderungen möglichst gut erfüllen:

- Verteilungsrobustheit,
- Modellrobustheit,
- Datenrobustheit,
- hohe Trennfähigkeit der Beobachtungen sowie
- optimale Ergebnisse bei „fehlerfreien“ Beobachtungen und korrekten Modellannahmen.

Die vorhandenen Fehlerquellen lassen sich in zwei Gruppen aufteilen [Rodehorst, 2004]:

Rauschen: Trotz gewissenhafter Messungen sind diese nicht vollkommen fehlerfrei; man nennt sie *verrauscht*. Es kann jedoch eine Normalverteilung der Fehler angenommen werden.

Grobe Fehler: Ihre Entstehung kann etwa auf Fehlzuordnung von Punktepaaren oder andere gravierende „Irrtümer“ zurückgeführt werden. Sie beeinflussen das Ergebnis beträchtlich, ihre Verteilung ist unregelmäßig.

Der wichtigste Schritt eines robusten Verfahrens besteht darin, die groben Fehler zu erkennen und zu eliminieren. Dafür gibt es mehrere Ansätze, wie z. B. jede Beobachtung invers zum Betrag ihrer normierten Verbesserung nach der ersten Iteration zu gewichten. Die normierte Verbesserung ergibt sich durch die Division der Verbesserung durch die Standardabweichung der Beobachtung. Es kann entweder die Standardabweichung *a priori* als auch die Standardabweichung *a posteriori* verwendet werden. Die zweite Möglichkeit ist auch unter *Data Snooping* bekannt [Kraus, 1997].

In einem anderen Ansatz wird zufällig aus der Menge der Beobachtungen ein minimaler Datensatz (*Random Sample*) ausgewählt, durch den die Modellparameter eindeutig bestimmt werden können. Pro Random Sample wird dann die Anzahl der „passenden" Beobachtungen gezählt. Diese Methode wurde in dieser Arbeit zur Schätzung der Fundamentalmatrix angewandt und wird in Abschnitt 5.3.1 vorgestellt.

5.3.1 Random Sample Consensus (RANSAC)

Der Ausgangspunkt für diese Methode ist eine gegebene Datenmenge S, die mit zufälligen und groben Fehlern behaftet ist, sowie ein aus diesen Daten bestmöglich zu bestimmendes Modell M [Hartley und Zisserman, 2001]. Im vorliegenden Fall ist S eine Menge von Punktkorrespondenzen, die mit dem Key Point Descriptor (siehe Abschnitt 3.3) gefunden wurde, und das Modell M die Fundamentalmatrix $\mathbf{F}_{12}$. Die Mächtigkeit des Random Samples s richtet sich nach dem verwendeten Algorithmus; in dieser Arbeit wurde der lineare 8-Punkt-Algorithmus verwendet, der in Abschnitt 5.2 vorgestellt wurde, deshalb ist $\#s = 8$.

Aus diesen acht zufällig ausgewählten Punktkorrespondenzen wird nun das Modell instantiiert, indem die Fundamentalmatrix $\mathbf{F}_{12}$ eindeutig mit dem 8-Punkt-Algorithmus berechnet wird. Nun werden alle Punktepaare $(\mathbf{x}', \mathbf{x}'')$ zur Menge $S_i \subset S$ hinzugenommen, für die gilt:

$$d(\mathbf{F}_{12}^{\mathrm{T}}\mathbf{x}', \mathbf{x}'')^2 < t^2.$$

Das sind also alle Punktepaare, für welche die Epipolargerade des ersten Bildpunktes $\mathbf{x}'$ vom korrespondierenden Punkt im zweiten Bild $\mathbf{x}''$ einen Normalabstand d besitzt, der kleiner ist als ein zuvor definierter Schwellwert t (*distance threshold*). Die Anzahl der Elemente von S_i wird als *Consensus* der dazugehöroigen Stichprobe s bezeichnet. Ist $\#S_i > T$, so bricht der Algorithmus ab und $\mathbf{F}_{12}$ wird überbestimmt mit allen Punktepaaren aus S_i berechnet.

Ist jedoch $\#S_i \leq T$, so wird solange ein neues Random Sample aus S gewählt, bis entweder obiges Kriterium für $\#S_i$ erfüllt ist oder bereits eine Maximalanzahl N an Random Samples durchprobiert wurden. In letzterem Fall wird jene Teilmenge S_i ausgewählt, für die der Consensus am höchsten ist. Die Fundamentalmatrix wird dann – wie oben bereits beschrieben – mit allen Punktepaaren aus S_i überbestimmt berechnet [Hartley und Zisserman, 2001].

Ermittlung der Schwellwerte. Die oben erwähnten Schwellwerte t, T und N ergeben sich aus deren angenommenen statistischen Verteilungen. Im Falle der Fundamentalmatrix ergibt sich $t^2 = 3,84 \cdot \sigma^2$, σ bezeichnet hier die Bildmessgenauigkeit [Hartley und Zisserman, 2001].

Die Zahl N ist ein Maß dafür, wie viele Samples „durchprobiert" werden müssen, um mit einer gewissen Wahrscheinlichkeit p zumindest ein Random Sample gefunden zu haben, das nur korrekte Punktepaare enthält. Wird $p = 0,99$ angenommen, und ist w die Wahrscheinlichkeit, dass eine zufällig ausgewählte Punktkorrespondenz korrekt ist, so gilt

$$(1 - w^{\#s})^N = (1 - p) = 0,01 \tag{5.12}$$

bzw. mit $\varepsilon = (1 - w)$ als Wahrscheinlichkeit, einen Ausreißer gefunden zu haben:

$$N = \frac{\log(1-p)}{\log(1-(1-\varepsilon)^{\#s})} = \frac{\log(1-p)}{\log(1-w^{\#s})}. \tag{5.13}$$

Geht man davon aus, dass die Anzahl der Elemente von S_i der Anzahl der korrekten Korrespondenzen in S entsprechen sollte, so ergibt sich bei $\#S = n$ der Schwellwert T zu

$$T = (1 - \varepsilon)n = w \cdot n. \tag{5.14}$$

Ein Beispiel: Es wird angenommen, im vorliegenden Datensatz mit $n = 100$ Punktepaaren wären 10% Ausreißer, also $\varepsilon = 0,1$. Somit ergeben sich

$$N = \frac{\log(1-0,99)}{\log(1-(1-0,1)^8)} = 8 \text{ und } T = (1-0,1) \cdot 100 = 90.$$

Adaptive Ermittlung der Schwellwerte. Die Anzahl der Ausreißer in S kann a priori meist schlecht geschätzt werden. Es bietet sich daher an, einen Algorithmus zu verwenden, in dem die Schwellwerte an das bisher „gesichtete" Datenmaterial angepasst werden. Dieser Algorithmus ist in Tabelle 5.1 beschrieben. Nach dem Terminieren dieses Algorithmus wird wieder die $\mathbf{F}_{12}$-Matrix mit allen Punkten desjenigen S_i berechnet, das den größten Consensus hatte [Hartley und Zisserman, 2001].

5.4 Die Essentielle Matrix $\mathbf{E}_{12}$

Sind die inneren Orientierungen der verwendeten Kameras und damit auch die Kalibrierungsatrizen $\mathbf{C}_1$ und $\mathbf{C}_2$ bekannt, so kann man die Gleichung (5.2) zu

$$\mathbf{E}_{12} \sim \mathbf{C}_1^T \mathbf{F}_{12} \mathbf{C}_2 = \mathbf{R}_1 \mathbf{S}(\mathbf{z}_2 - \mathbf{z}_1)\mathbf{R}_2^{\mathrm{T}}. \tag{5.15}$$

umformen. Die Matrix $\mathbf{E}_{12}$ in obiger Gleichung wird als *Essentielle Matrix* bezeichnet [Ressl, 2003; Hartley und Zisserman, 2001]. Setzt man wie in Abschnitt 5.1 $\mathbf{z}_1 = \mathbf{0}$ und $\mathbf{R}_1 = \mathbf{I}$, so vereinfacht sich Gleichung (5.15) zu

$$\mathbf{E}_{12} \sim \mathbf{S}(\mathbf{z}_2)\mathbf{R}_2^{\mathrm{T}}.$$

- $N = \infty$, `sample_count` $= 0$,
- While $N >$ `sample_count` do
  - Zufällige Stichprobe wählen und Anzahl der *Inliers* zählen ($\#S_i$)
  - $\varepsilon = (1 - \frac{\#S_i}{n})$ setzen
  - N aus Formel (5.13) berechnen ($p = 0,99$)
  - `sample_count` um 1 erhöhen
- Beenden

Tabelle 5.1: Adaptive Bestimmung der RANSAC-Parameter (in Pseudocode), nach [Hartley und Zisserman, 2001].

Aus dieser Gleichung ist auch leicht die Tatsache ersichtlich, dass $\mathbf{E}_{12}$ fünf Freiheitsgrade besitzt: drei Freiheitsgrade ergeben sich aus den in $\mathbf{R}_2$ implizit enthaltenen Rotationswinkeln, zwei weitere aus dem Basisvektor $\mathbf{z}_2$, da die Essentielle Matrix nur bis auf den Maßstab bestimmbar ist.

Um nun die Orientierungsparameter in Gestalt von $\mathbf{R}_2$ und $\mathbf{z}_2$ zu berechnen, wird eine Singulärwertzerlegung von $\mathbf{E}_{12} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}$ durchgeführt [Ressl, 2003].

Aus der Singulärwertzerlegung von

$$\mathbf{E}_{12} = \mathbf{U}\mathbf{S}\mathbf{V}^{\mathrm{T}}, \quad \mathbf{U} = (\mathbf{u}_1, \mathbf{u}_2, \mathbf{u}_3)$$

ergeben sich

$$\mathbf{z}_2 = \mathbf{u}_3, \quad \overline{\mathbf{z}}_2 = -\mathbf{u}_3 \tag{5.16}$$

und

$$\mathbf{R}_2 = \mathbf{U}\mathbf{W}\mathbf{V}^{\mathrm{T}}, \quad \overline{\mathbf{R}}_2 = \mathbf{U}\mathbf{W}^{\mathrm{T}}\mathbf{V}^{\mathrm{T}}, \qquad \mathbf{W} = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix}. \tag{5.17}$$

Gegebenfalls müssen $\mathbf{R}_2$ und $\overline{\mathbf{R}}_2$ noch mit (-1) multipliziert werden, damit die Bedingung $\det(\mathbf{R}_2) = \det(\overline{\mathbf{R}}_2) = 1$ für orthonormierte Matrizen erfüllt ist.

Von den sich ergebenden vier Kombinationen $(\mathbf{R}_2, \mathbf{z}_2)$, $(\overline{\mathbf{R}}_2, \mathbf{z}_2)$, $(\mathbf{R}_2, \overline{\mathbf{z}}_2)$ und $(\overline{\mathbf{R}}_2, \overline{\mathbf{z}}_2)$ ist nun die einzig passende Lösung auszuwählen. Dies geschieht durch die Bedingung der Positivstellung der Bilder (der Bildpunkt liegt jeweils zwischen Projektionszentrum und Objektpunkt):

$$\mu_1 \mathbf{C}_1^{-1}\mathbf{x}' = \mathbf{z}_2 + \mu_2 \mathbf{R}_2^{\mathrm{T}}\mathbf{C}_2\mathbf{x}''. \tag{5.18}$$

$\mathbf{x}'$ und $\mathbf{x}''$ bezeichnen in dieser Formel eine bekannte Punktkorrespondenz. Multipliziert man obige Gleichung von links mit $\mathbf{S}(\mathbf{z}_2)$ und fordert für die Skalare μ_1 und μ_2 ein positives Vorzeichen, so ergibt sich

$$\operatorname{sgn}\big(\mathbf{S}(\mathbf{z}_2)\mathbf{C}_1^{-1}\mathbf{x}'\big) = \operatorname{sgn}\Big(\mathbf{S}(\mathbf{z}_2)\mathbf{R}_2^{\mathrm{T}}\mathbf{C}_2^{-1}\mathbf{x}''\Big). \tag{5.19}$$

Diese Bedingung erfüllt nur eine der vier oben aufgeführten Kombinationen für die Rotationsmatrix $\mathbf{R}_2$ und den Basisvektor $\mathbf{z}_2$. Damit ist die Relative Orientierung beider Bilder eindeutig bestimmt [Ressl, 2003].

5.4.1 Anpassung der projektiven Tiefe

Bislang lieferte die Relative Orientierung nur maßstabslose Größen. Um Bildpaare explizit verknüpfen zu können, muss die Länge des Basisvektors vorgegeben werden, die anderen werden daran angepasst. Wird etwa bei drei Bildern die Basis $\mathbf{b}_{12}$ auf die Länge 1 normiert, so muss für die anderen Basisvektoren $\mathbf{b}_{13}$ und $\mathbf{b}_{23}$ gelten:

$$\mathbf{b}_{12} + \lambda_{23}\mathbf{R}_2^{\mathrm{T}}\mathbf{b}_{23} - \lambda_{13}\mathbf{b}_{13} = \mathbf{0}. \tag{5.20}$$

Es ergeben sich somit drei Gleichungen für die zwei Unbekannnten λ_{13} und λ_{23}, die auch *projektive Tiefen* genannt werden.

5.5 Selbstkalibrierung

In dieser Arbeit wurde nur mit kalibrierten Aufnahmen gearbeitet. Der Vollständigkeit halber seien nun noch Verfahren vorgestellt, mit denen man Informationen über die innere Geometrie der Aufnahmen rein über Punktkorrespondenzen und Annahmen über die Aufnahmesysteme selbst erhält. Diese Verfahren werden unter dem Begriff *Selbstkalibrierung* oder *Autokalibrierung* zusammengefasst.

Die verschiedenen Ansätze zur Selbstkalibrierung können in zwei Klassen unterteilt werden [Rodehorst, 2004]:

Klasse A: Die Matrix $\mathbf{C}$ wird aus den Fundamentalmatrizen $\mathbf{F}_{ij}$ berechnet. Die innere Orientierung wird als unbekannt, aber über alle Bilder konstant angenommen.

Klasse B: Die Kamera-Matrizen $\mathbf{C}_i$ werden aus den projektiven Rekonstruktionen der Aufnahmen bestimmt (Projektionsmatrizen $\mathbf{P}_i$).

Die Verfahren der Klasse B haben zwar einen höheren Berechnungsaufwand, es ergeben sich aber weniger degenerierte Situationen als bei den Verfahren der Klasse A [Hartley und Zisserman, 2001; Rodehorst, 2004]. Je nachdem, wie die innere Geometrie der Kamera bzw. Kameras angenommen wird, ist eine unterschiedliche Mindestanzahl an Bildern für die

Bedingung	bekannt	konstant	# Bilder
keine Scherung s	$s = 0$		≥ 8
keine Scherung s, konst. Seitenverhältnis der Pixel	$s = 0$	$\frac{\alpha_x}{\alpha_y}$	≥ 5
keine Scherung s , bek. Seitenverhältnis der Pixel	$s = 0$, $\frac{\alpha_x}{\alpha_y}$		≥ 4
nur Kamerakonstante unbekannt	s, $\frac{\alpha_x}{\alpha_y}$, x_0, y_0		≥ 2
Standardproblem		s, $\frac{\alpha_x}{\alpha_y}$, x_0, y_0	≥ 3

Tabelle 5.2: Verschiedene Varianten der Selbstkalibrierung [Pollefeys et al., 1999].

Selbstkalibrierung nötig, wobei immer folgende Ungleichung erfüllt sein muss [Pollefeys et al., 1999]:

$$n \cdot b + (n-1) \cdot k \geq 8. \tag{5.21}$$

In obiger Formel bezeichnet n die Anzahl der Bilder, b die Anzahl *bekannter Parameter* und k die Anzahl der *über alle Bilder konstanten, aber unbekannten Parameter*. Die unterschiedlichen Szenarien sind in Tabelle 5.2 aufgelistet. In den folgenden Abschnitten wird je ein Verfahren der Klasse A und der Klasse B vorgestellt.

5.5.1 Selbstkalibrierung mit den *Kruppa-Gleichungen*

Diese Methode war die erste in der Geschichte der Selbstkalibrierung. Sie geht auf den österreichischen Mathematiker Erwin Kruppa (1885-1967) zurück, der sie bereits 1913 publizierte. In den 90er-Jahren des 20. Jahrhunderts wurde dieses Verfahren in der Computer Vision wiederentdeckt [Hartley und Zisserman, 2001; Rodehorst, 2004].

Für jedes Bild erhält man eine positiv definite, symmetrische Matrix ω^* der Form

$$\boldsymbol{\omega}^* \sim \mathbf{C}\mathbf{C}^{\mathrm{T}}. \tag{5.22}$$

Diese Matrix, welche die Abbildung des *dualen absoluten Kegelschnitts*[1] definiert, besitzt fünf Freiheitsgrade und wird *Kruppa-Matrix* benannt. Die eigentlichen Kruppa-Gleichungen lauten

$$\mathbf{S}(\mathbf{e}'')\omega'^*\mathbf{S}(\mathbf{e}'') \sim \mathbf{F}^{\mathrm{T}}\omega^*\mathbf{F}. \tag{5.23}$$

Die Kruppa-Matrix ist direkt aus der Fundamentalmatrizen $\mathbf{F}_{ij}$ bestimmbar. Unter konstanter innerer Orientierung gilt $\omega'^* = \omega^*$. Da sich aus jeder $\mathbf{F}_{ij}$-Matrix zwei unabhängige quadratische Gleichungen ergeben, ist eine nichtlineare Lösung für ω^* ab drei Bildern möglich, aber recht instabil. Genauere Behandlungen der Kruppa-Gleichungen finden sich in [Zeller und Faugeras, 1996] und [Hartley und Zisserman, 2001].

[1] Schnitt der dualen absoluten Quadrik Q^*_∞ mit der Fernebene π_∞.

5.5.2 Selbstkalibrierung mit der dualen absoluten Quadrik

Dieses Verfahren operiert linear und gehört der Klasse B an. Statt der Fundamentalmatrizen $\mathbf{F}_{ij}$ werden die Projektionsmatrizen $\mathbf{P}_i$ der einzelnen Aufnahmen benützt. Diese ergeben sich jedoch sehr einfach aus den Fundamentalmatrizen, wenn man ein Projektionszentrum in den Ursprung legt und die zugehörige Projektionsmatrix

$$\mathbf{P}_1 = \left(\begin{array}{ccc|c} 1 & & & \\ & 1 & & \mathbf{0} \\ & & 1 & \end{array} \right) \tag{5.24}$$

setzt. Die Projektionsmatrizen der anderen Aufnahmen $\mathbf{P}_j$ ergeben sich zu [Hartley und Zisserman, 2001]

$$\mathbf{P}_j \sim \left(\mathbf{S}(\mathbf{e}_1^{(j)}) \mathbf{F}_{1j}^{\mathrm{T}} | \mathbf{e}_1^{(j)} \right). \tag{5.25}$$

Wie gesagt stellen diese Matrizen nur eine projektive Rekonstruktion der Aufnahmesituation dar, was bei Betrachtung von Gleichung (5.24) sofort klar wird. Die *metrischen Rekonstruktionen* $\mathbf{P}_j^{\mathrm{M}}$ der Szene werden durch die *rektifizierende Kollineation* κ erreicht. Diese wird durch die Matrix $\mathbf{H}$ induziert und es gilt:

$$\mathbf{P}_j^{\mathrm{M}} \sim \mathbf{P}_j \mathbf{H}. \tag{5.26}$$

Die Gleichung für die Selbstkalibrierung lautet folgendermaßen:

$$\omega_i^* \sim \mathbf{C}_i \mathbf{C}_i^{\mathrm{T}} \sim \mathbf{P}_i \mathbf{Q}_\infty^* \mathbf{P}_i^{\mathrm{T}}. \tag{5.27}$$

Diese Gleichung ist linear, wenn keine Scherung vorliegt und sich der Bildhauptpunkt jeweils im Ursprung des Kamera-Koordinatensystems befindet. Daraus folgernd kann man ω_i^* als

$$\omega_i^* \sim \begin{pmatrix} \alpha_x^2 & & \\ & \alpha_y^2 & \\ & & 1 \end{pmatrix} \sim \mathbf{P}_i \begin{pmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{12} & q_{22} & q_{23} & q_{24} \\ q_{13} & q_{23} & q_{33} & q_{34} \\ q_{14} & q_{24} & q_{34} & q_{44} \end{pmatrix} \mathbf{P}_i^{\mathrm{T}} \tag{5.28}$$

darstellen. Ein lineares homogenes Gleichungssystem in den Elementen q_{ij} von $\mathbf{Q}_\infty^*$ lässt sich in der Form $\mathbf{A}\mathbf{q} = \mathbf{0}$ anschreiben; hierbei ist

$$\mathbf{q} = (q_{11}, q_{12}, q_{13}, q_{14}, q_{22}, q_{23}, q_{24}, q_{33}, q_{34}, q_{44})^{\mathrm{T}}$$

und jede Projektionsmatrix $\mathbf{P}_i$ liefert zur Modellmatrix $\mathbf{A}$ die drei Zeilen

$$\mathbf{A}_i = \begin{pmatrix} p_{11}p_{21} & p_{11}p_{31} & p_{21}p_{31} \\ p_{11}p_{22}+p_{12}p_{21} & p_{11}p_{32}+p_{12}p_{31} & p_{21}p_{32}+p_{22}p_{31} \\ p_{11}p_{23}+p_{13}p_{21} & p_{11}p_{33}+p_{13}p_{31} & p_{21}p_{33}+p_{23}p_{31} \\ p_{11}p_{24}+p_{14}p_{21} & p_{11}p_{34}+p_{14}p_{31} & p_{21}p_{34}+p_{24}p_{31} \\ p_{12}p_{22} & p_{12}p_{32} & p_{22}p_{32} \\ p_{12}p_{23}+p_{13}p_{22} & p_{12}p_{33}+p_{13}p_{32} & p_{22}p_{33}+p_{23}p_{32} \\ p_{12}p_{24}+p_{14}p_{22} & p_{12}p_{34}+p_{14}p_{32} & p_{22}p_{34}+p_{24}p_{32} \\ p_{13}p_{23} & p_{13}p_{33} & p_{23}p_{33} \\ p_{13}p_{24}+p_{14}p_{23} & p_{13}p_{34}+p_{14}p_{33} & p_{23}p_{34}+p_{24}p_{33} \\ p_{14}p_{24} & p_{14}p_{34} & p_{24}p_{34} \end{pmatrix}^{\mathrm{T}}. \tag{5.29}$$

Ist zusätzlich das Seitenverhältnis γ der Pixel bekannt, ergibt sich folgende Bedinungsgleichung [Rodehorst, 2004]:

$$\mathbf{p}_1^{\mathrm{T}}\mathbf{Q}_\infty^*\mathbf{p}_1 = \mathbf{p}_2^{\mathrm{T}}\mathbf{Q}_\infty^*\mathbf{p}_2, \quad \mathbf{P} \sim (\mathbf{p}_1, \mathbf{p}_2, \mathbf{p}_3)^{\mathrm{T}}. \tag{5.30}$$

Für jede Projektionsmatrix $\mathbf{P}_i$ wird in diesem Falle der Modellmatrix $\mathbf{A}$ die Zeile

$$\mathbf{A}_i = \begin{pmatrix} \gamma p_{11}^2 - p_{21}^2 \\ 2(\gamma p_{11}p_{12} - p_{21}p_{22}) \\ 2(\gamma p_{11}p_{13} - p_{21}p_{23}) \\ 2(\gamma p_{11}p_{14} - p_{21}p_{24}) \\ \gamma p_{12}^2 - p_{22}^2 \\ 2(\gamma p_{12}p_{13} - p_{22}p_{23}) \\ 2(\gamma p_{12}p_{14} - p_{22}p_{24}) \\ \gamma p_{13}^2 - p_{23}^2 \\ 2(\gamma p_{13}p_{14} - p_{23}p_{24}) \\ \gamma p_{14}^2 - p_{24}^2 \end{pmatrix}^{\mathrm{T}} \tag{5.31}$$

hinzugefügt.

Die Matrix $\mathbf{Q}_\infty^*$ ist symmetrisch und nur bis auf den Maßstab bestimmbar. Sie besitzt daher 9 Freiheitsgrade. Sind drei oder mehr Aufnahmen vorhanden, so hat die Modellmatrix ≥ 9 Zeilen. Die Aufgabe der Selbstkalibrierung ist also ab drei Aufnahmen lösbar.

5.6 Berechnung der Orientierungsparameter

Für die metrische Rekonstruktion muss die duale absolute Quadrik in ihre kanonische Form $\mathbf{Q}_{\mathrm{M}}^* = \mathrm{diag}(1,1,1,0)$ gebracht werden. Dies geschieht durch die oben erwähnte rektifizierende Kollineation κ, daher gilt:

$$\mathbf{Q}_\infty^* \sim \mathbf{H}\mathbf{Q}_{\mathrm{M}}^*\mathbf{H}^{\mathrm{T}}. \tag{5.32}$$

$\mathbf{H}$ wird mit Hilfe der Eigenwert-Zerlegung [Havlicek, 2003b] aus $\mathbf{Q}_{\mathrm{M}}^{*} = \mathbf{E}\mathbf{D}\mathbf{E}^{\mathrm{T}}$ bestimmt und ergibt sich zu

$$\mathbf{H} \sim \mathbf{E}\overline{\mathbf{D}}, \quad \overline{\mathbf{D}} = \begin{pmatrix} \sqrt{d_{11}} & 0 & 0 & 0 \\ 0 & \sqrt{d_{33}} & 0 & 0 \\ 0 & 0 & \sqrt{d_{33}} & 0 \\ 0 & 0 & 0 & 0 \end{pmatrix}. \tag{5.33}$$

Um zu vermeiden, dass $\mathbf{H}$ singulär wird, wird $h_{44} = 1$ gesetzt [Rodehorst, 2004].

Nun können, wie in Abschnitt 4.4 beschrieben, die ersten drei Spalten der metrischen Rekonstruktionen $\mathbf{P}_j^{\mathrm{M}}$ mittels QR-Zerlegung [Havlicek, 2003b] in eine orthonormale Matrix $\mathbf{R}_j$ (Rotationsmatrix) und eine obere Dreiecksmatrix $\mathbf{C}_j$ (Kalibrierungsmatrix) aufgespalten werden. Der Basisvektor ergibt sich bis auf den Maßstab aus der vierten Spalte der Projektionsmatrix.

Ist ausschließlich die Kamerakonstante c unbekannt, so genügen bereits zwei Aufnahmen zur Selbstkalibrierung [Hartley und Zisserman, 2001; Rodehorst, 2004].

Kapitel 6

Experimentelle Ergebnisse

Das in den vorigen Kapiteln beschriebene Verfahren zur automatischen Berechnung der Relativen Orientierung wurde anhand von drei Beispielen im Nahbereich getestet. Es wurde jeweils die digitale Spiegelreflexkamera Nikon D70 mit einer Auflösung von 3.008×2.000 Pixeln und ein $15mm$-Objektiv verwendet. Es wurde jedoch nur ein Viertel der Auflösung der Bilder benützt, da das Demo-Programm zur Keypoint-Extraktion von David Lowe nur Bilder bis zu einer gewissen Größe akzeptierte. Die Kalibrierungsmatrix $\mathbf{C}$ hatte also folgende Gestalt (bei einer Pixelgröße von 8 μm):

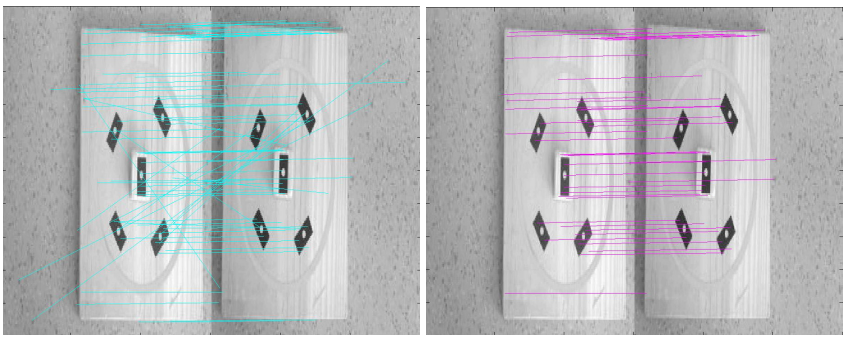
$$\mathbf{C} = \begin{pmatrix} -1930 & 0 & 1504 \\ 0 & -1930 & -1000 \\ 0 & 0 & 1 \end{pmatrix}.$$

Bei allen Versuchen standen Passpunktfelder zur Verfügung, sodass die Ergebnisse der automatisch ermittelten Relativen Orientierung kontrolliert werden konnten. Die Referenzwerte wurden mittels Bündelblockausgleich in der Photogrammetriesoftware ORPHEUS ermittelt. In der Folge werden nun die verschiedenen Testszenarien mit ihren wichtigsten Eigenschaften sowie den Ergebnissen dargestellt. Die Größen ω, φ und κ entsprechen den relativen Drehwinkeln in gon, die $\mathbf{b}_{ij}$ entsprechen den normierten Basisvektoren von Projektionszentrum i nach Projektionszentrum j.

6.1 EO Device

Die retroreflektierenden Marken des auf den Abbildungen 6.1 bis 6.3 dargestellten *EO Device*[1] stellen die Realisierung eines Koordinatensystems dar. Ein EO Device wird in der Präzisions-Nahbereichsphotogrammetrie als datumgebendes Element eingesetzt. Von diesem EO Device wurden je zwei Aufnahmen gemacht mit

[1] EO steht für *Exterior Orientation*, also Äußere Orientierung.



(a) Homologe Punktepaare nach dem Matching.

(b) Homologe Punktepaare nach der robusten Schätzung.

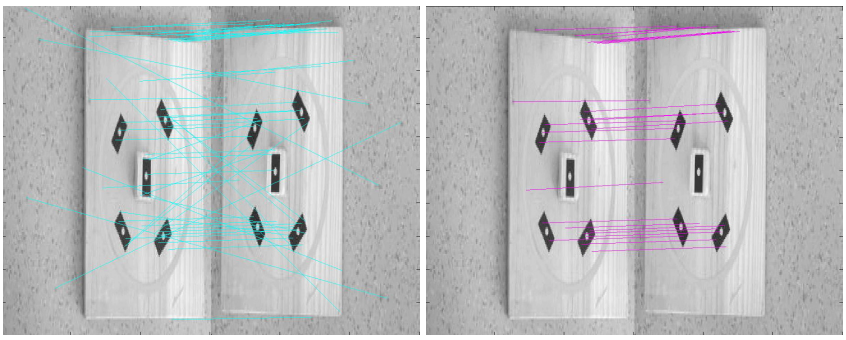
Abbildung 6.1: Genäherter Normalfall.

- nahezu parallelen Hauptstrahlen (Normalfall),
- ca. 10° konvergenten Hauptstrahlen und
- ca. 30° konvergenten Hauptstrahlen.

Da die retroreflektierenden Marken Passpunkte darstellen, konnten in ORPHEUS die Absoluten Orientierungen der Aufnahmen in Bezug auf den EO Device über eine Bündelblockausgleichung berechnet werden. In den nächsten Abschnitten sind die Ergebnisse der automatischen Relativen Orientierung sowie die Vergleiche mit den Werten, die sich aus der Bündelblockausgleichung ergaben, aufgelistet.

Genäherter Normalfall.

Größe	MATLAB	ORPHEUS
ω_{12}	$-0,619$	$3,316$
φ_{12}	$-3,718$	$3,149$
κ_{12}	$-0,537$	$1,248$
$\mathbf{b}_{12}$	$\begin{pmatrix} 0,93 \\ 0,02 \\ 0,35 \end{pmatrix}$	$\begin{pmatrix} 0,893 \\ 0,445 \\ -0,064 \end{pmatrix}$



(a) Homologe Punktepaare nach dem Matching.

(b) Homologe Punktepaare nach der robusten Schätzung.

Abbildung 6.2: Aufnahmen vom EO Device mit 10° konvergenten Hauptstrahlen.

10° Konvergenz.

Größe	MATLAB	ORPHEUS
ω_{12}	$-5,479$	$-1,7604$
φ_{12}	$-30,645$	$-20,565$
κ_{12}	$-2,979$	$-0,951$
$\mathbf{b}_{12}$	$\begin{pmatrix} 0,881 \\ 0,168 \\ -0,439 \end{pmatrix}$	$\begin{pmatrix} 0,983 \\ 0,042 \\ -0,177 \end{pmatrix}$

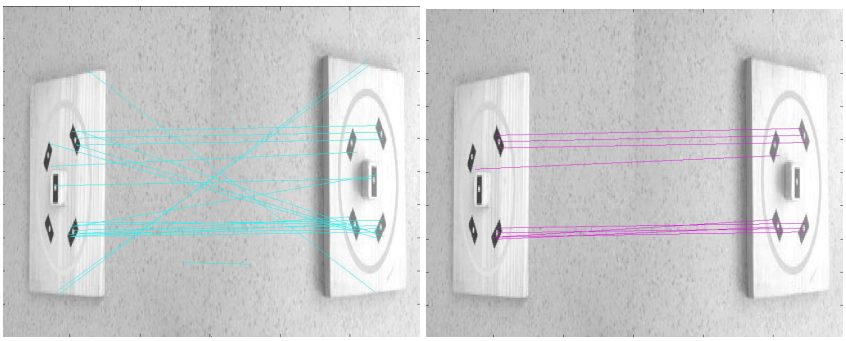
30° Konvergenz.

Größe	MATLAB	ORPHEUS
ω_{12}	$8,541$	$4,6502$
φ_{12}	$-60,380$	$-76,367$
κ_{12}	$-4,0734$	$6,324$
$\mathbf{b}_{12}$	$\begin{pmatrix} -0,94 \\ -0,16 \\ -0,31 \end{pmatrix}$	$\begin{pmatrix} -0,799 \\ -0,035 \\ -0,599 \end{pmatrix}$

Die teilweise recht ungenauen Orientierungsparameter lassen sich auf das schwach texturierte Aufnahmeobjekt und die geringe Fläche, die die korrespondierenden Punkte im Bild einehmen, zurückführen.

6.2 Messkeller

Hier wurde das Verfahren vor allem auf seine Invarianz gegenüber Beleuchtungsänderungen getestet, da nicht in allen Aufnahmen geblitzt wur-



(a) Homologe Punktepaare nach dem Matching.

(b) Homologe Punktepaare nach der robusten Schätzung.

Abbildung 6.3: Aufnahmen mit vom EO Device 30° konvergenten Hauptstrahlen.

de und auch nicht in allen Aufnahmen die Scheinwerfer eingeschaltet wurden. Insgesamt wurden 13 Fotos von wechselnden Standpunkten und mit teilweise stark variierenden Orientierungen gemacht. Es wurde die Bildsequenzen DSC0002–DSC0003 und DSC0003–DSC0007 zur Analyse ausgewählt, da in ihnen die stärksten Beleuchtungsvariationen auftraten. Die von der robusten Schätzung als korrekt befundenen Punktkorrespondenzen sind in den Abbildungen 6.4 bis 6.13 dargestellt.

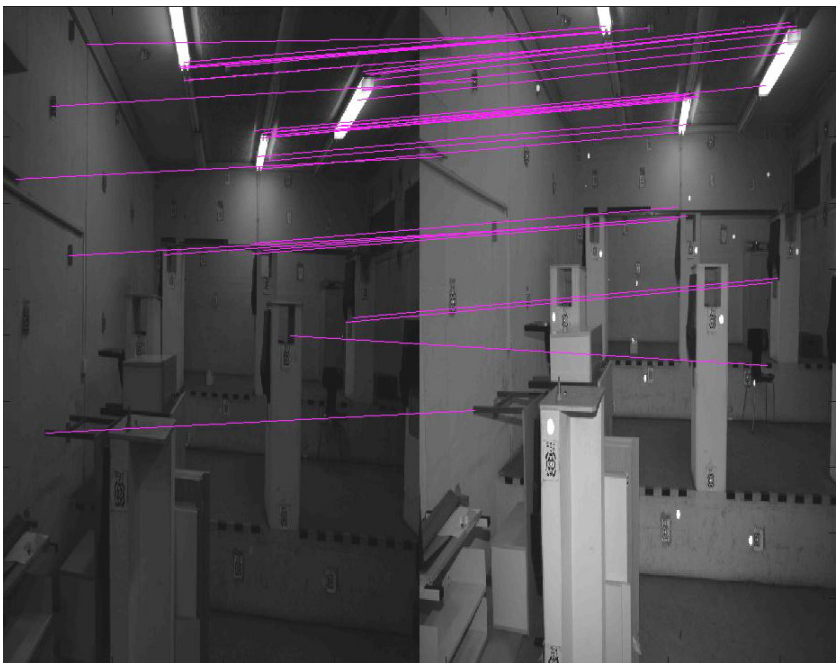


Abbildung 6.4: Homologe Punktepaare der Bilder DSC0002 und DSC0003.

DSC0002–DSC0003.

Größe	MATLAB	ORPHEUS
ω_{23}	$-3,2457$	$-2,9369$
φ_{23}	$1,5148$	$0,6975$
κ_{23}	$-1,7210$	$-1,5758$
$\mathbf{b}_{23}$	$\begin{pmatrix} -0,311 \\ 0,329 \\ -0,8926 \end{pmatrix}$	$\begin{pmatrix} -0,984 \\ 0,177 \\ -0,016 \end{pmatrix}$

Die große Abweichung des Basisvektors vom Sollwert lässt sich mit der extrem kurzen Basislänge von nur etwa 5 cm erklären. Sie ist bei einer weiteren Ausgleichung nicht von Bedeutung.

DSC0002–DSC0004. Aufgrund der zu starken Abweichungen von den Sollwerten sind die automatisch abgeleiteten Orientierungsparameter nicht dargestellt. Der Ursache für die schlechten Ergebnisse könnte vielleicht in der schlechten Ausleuchtung im Bild DSC0002 begründet liegen.

DSC0003–DSC0004. Aufgrund der zu starken Abweichungen von den Sollwerten sind die automatisch abgeleiteten Orientierungsparameter nicht

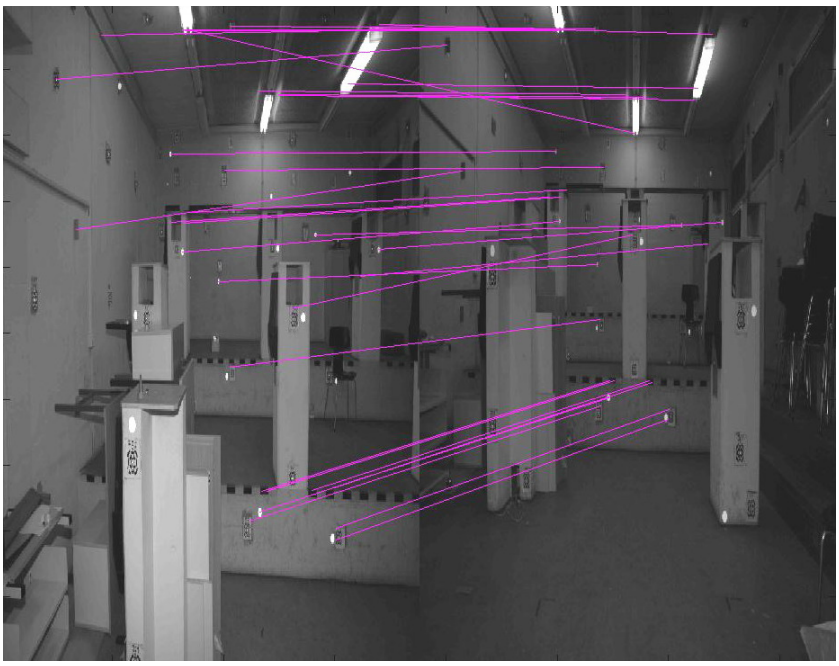


Abbildung 6.5: Homologe Punktepaare der Bilder DSC0003 und DSC0005.

dargestellt. Der Grund für die schlechten Ergebnisse könnte vielleicht in der schlechten Ausleuchtung im Bild DSC0002 begründet liegen.

DSC0003–DSC0005.

Größe	MATLAB	ORPHEUS
ω_{35}	$4,5195$	$3,8284$
φ_{35}	$-3,2552$	$-3,7089$
κ_{35}	$1,3779$	$0,6104$
$\mathbf{b}_{35}$	$\begin{pmatrix} 0,512 \\ -0,564 \\ 0,647 \end{pmatrix}$	$\begin{pmatrix} 0,490 \\ -0,574 \\ 0,655 \end{pmatrix}$

DSC0003–DSC0006.

Größe	MATLAB	ORPHEUS
ω_{36}	$4,5292$	$4,3576$
φ_{36}	$-10,1190$	$-10,4500$
κ_{36}	$2,9866$	$0,5839$
$\mathbf{b}_{36}$	$\begin{pmatrix} -0,121 \\ -0,071 \\ 0,989 \end{pmatrix}$	$\begin{pmatrix} -0,155 \\ -0,075 \\ 0,985 \end{pmatrix}$

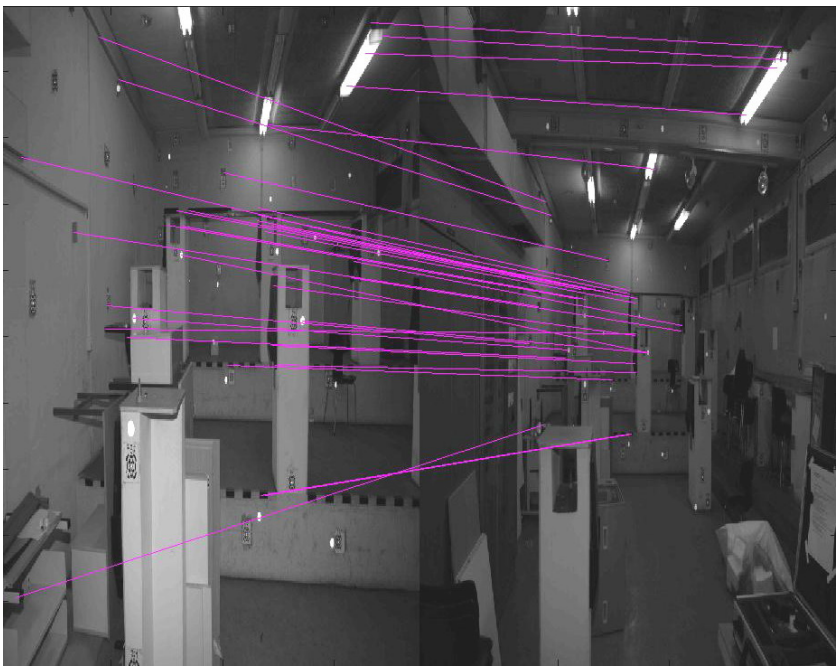


Abbildung 6.6: Homologe Punktepaare der Bilder DSC0003 und DSC0006.

DSC0003–DSC0007.

Größe	MATLAB	ORPHEUS
ω_{37}	$4,3914$	$4,0356$
φ_{37}	$15,617$	$17,161$
κ_{37}	$-0,1717$	$0,7350$
$\mathbf{b}_{37}$	$\begin{pmatrix} 0,667 \\ -0,059 \\ 0,742 \end{pmatrix}$	$\begin{pmatrix} 0,691 \\ -0,052 \\ 0,721 \end{pmatrix}$

DSC0004–DSC0005.

Größe	MATLAB	ORPHEUS
ω_{45}	$1,7670$	$1,6116$
φ_{45}	$0,2924$	$-0,2021$
κ_{45}	$1,2344$	$0,525$
$\mathbf{b}_{45}$	$\begin{pmatrix} -0,859 \\ -0,508 \\ -0,069 \end{pmatrix}$	$\begin{pmatrix} -0,854 \\ -0,494 \\ -0,165 \end{pmatrix}$

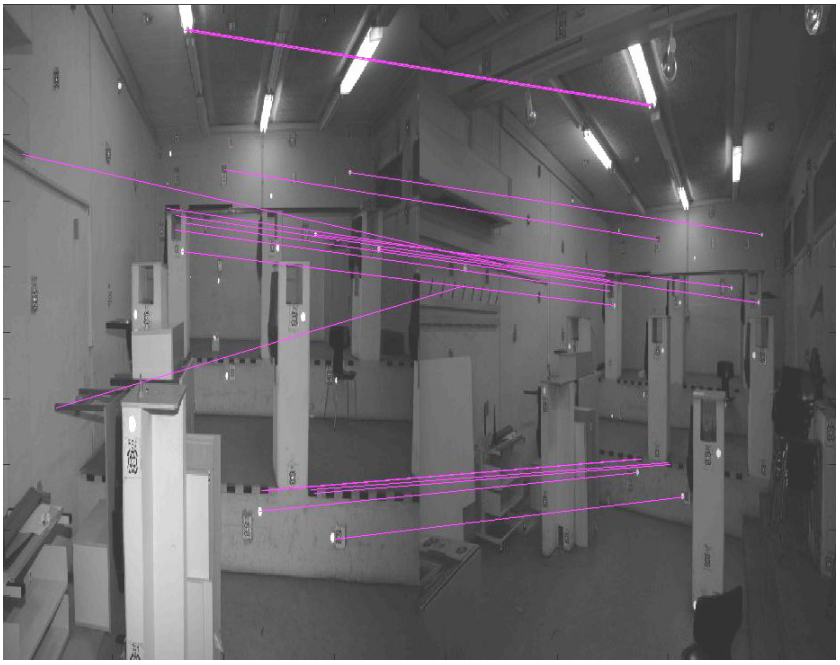


Abbildung 6.7: Homologe Punktepaare der Bilder DSC0003 und DSC0007.

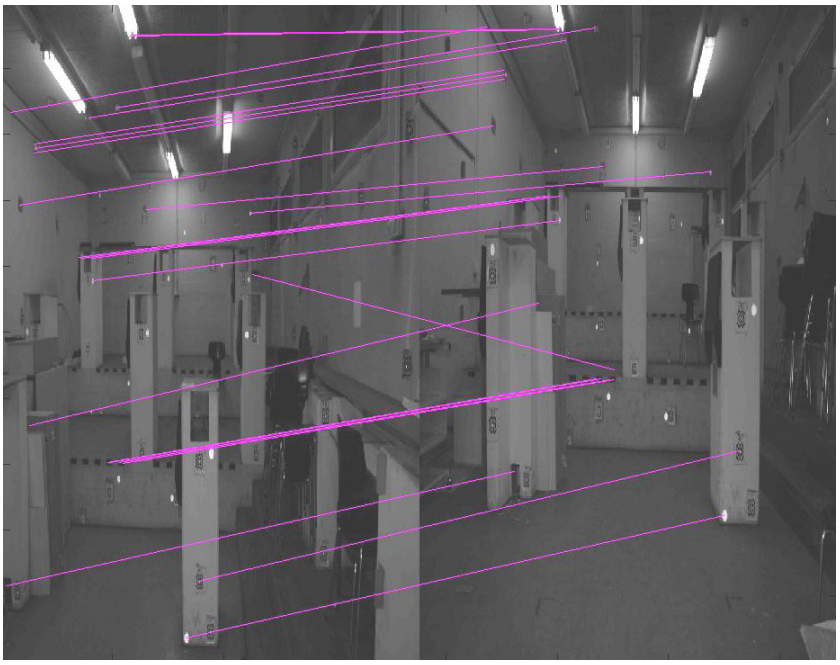


Abbildung 6.8: Homologe Punktepaare der Bilder DSC0004 und DSC0005.

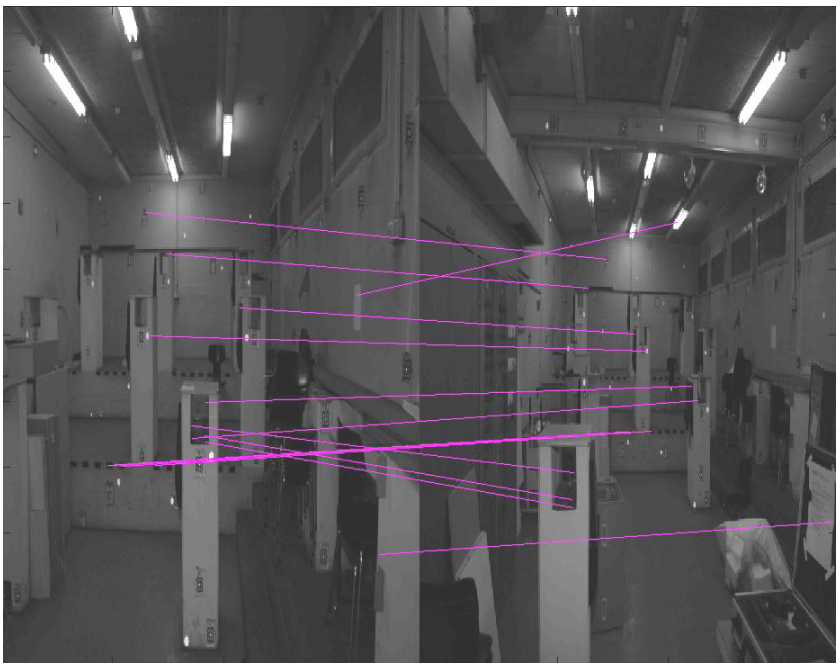


Abbildung 6.9: Homologe Punktepaare der Bilder DSC0004 und DSC0006.

DSC0004–DSC0006.

Größe	MATLAB	ORPHEUS
ω_{46}	$1,7670$	$1,6116$
φ_{46}	$0,2924$	$-0,2021$
κ_{46}	$1,2344$	$0,525$
$\mathbf{b}_{46}$	$\begin{pmatrix} -0,859 \\ -0,508 \\ -0,069 \end{pmatrix}$	$\begin{pmatrix} -0,854 \\ -0,494 \\ -0,165 \end{pmatrix}$

DSC0004–DSC0007.

Größe	MATLAB	ORPHEUS
ω_{47}	$1,8051$	$1,8151$
φ_{47}	$21,102$	$20,668$
κ_{47}	$0,357$	$0,6328$
$\mathbf{b}_{47}$	$\begin{pmatrix} 0,052 \\ 0,018 \\ 0,999 \end{pmatrix}$	$\begin{pmatrix} 0,026 \\ -0,039 \\ 0,999 \end{pmatrix}$

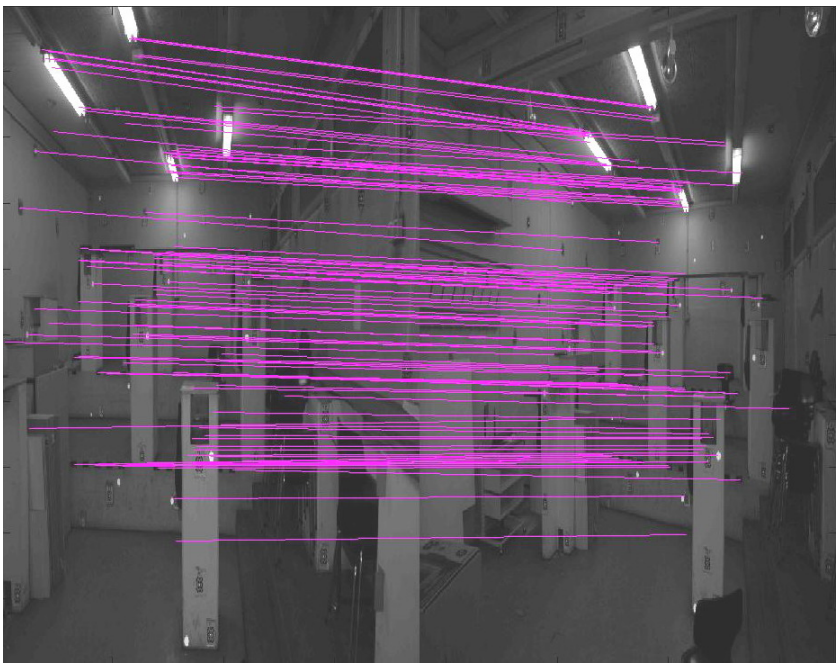


Abbildung 6.10: Homologe Punktepaare der Bilder DSC0004 und DSC0007.

DSC0005–DSC0006.

Größe	MATLAB	ORPHEUS
ω_{56}	$-0,8307$	$0,46014$
φ_{56}	$-8,4688$	$-6,7463$
κ_{56}	$0,3814$	$-0,061$
$\mathbf{b}_{56}$	$\begin{pmatrix} -0,373 \\ -0,260 \\ 0,890 \end{pmatrix}$	$\begin{pmatrix} -0,340 \\ 0,212 \\ 0,916 \end{pmatrix}$

DSC0005–DSC0007.

Größe	MATLAB	ORPHEUS
ω_{57}	$0,3675$	$0,4185$
φ_{57}	$20,235$	$20,867$
κ_{57}	$-0,801$	$0,0774$
$\mathbf{b}_{57}$	$\begin{pmatrix} 0,740 \\ 0,416 \\ 0,529 \end{pmatrix}$	$\begin{pmatrix} 0,733 \\ 0,406 \\ 0,546 \end{pmatrix}$

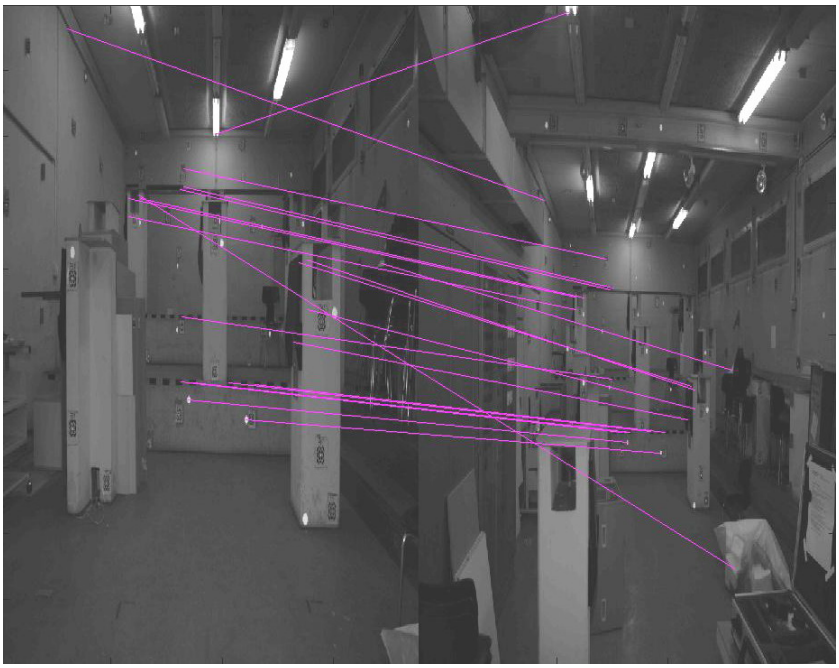


Abbildung 6.11: Homologe Punktepaare der Bilder DSC0005 und DSC0006.

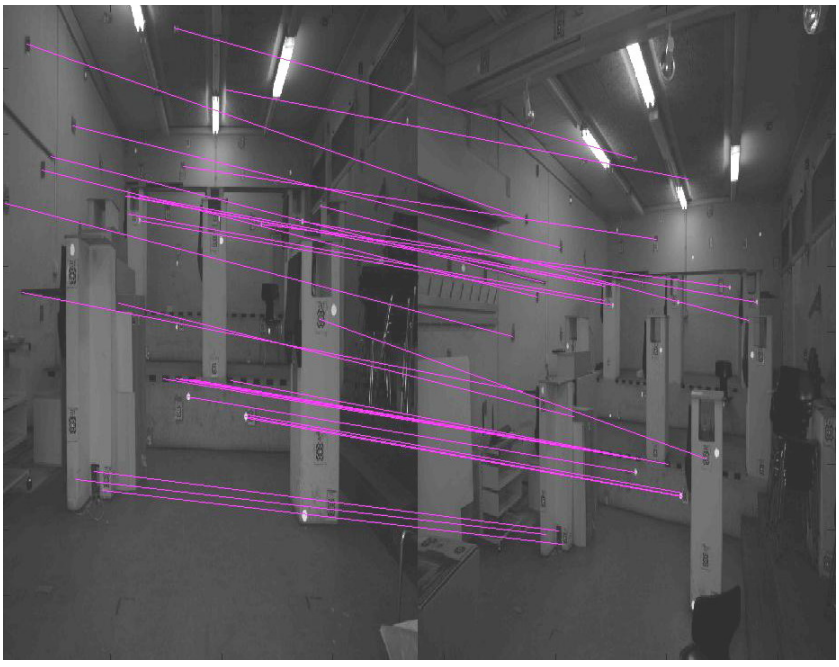


Abbildung 6.12: Homologe Punktepaare der Bilder DSC0005 und DSC0007.

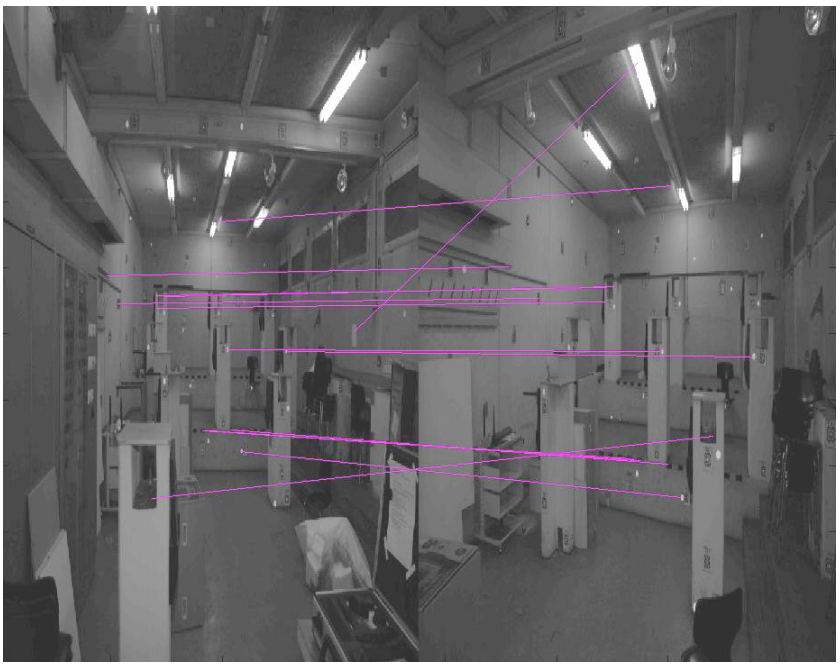


Abbildung 6.13: Homologe Punktepaare der Bilder DSC0006 und DSC0007.

DSC0006–DSC0007.

Größe	MATLAB	ORPHEUS
ω_{67}	$-1,0031$	$-0,0072$
φ_{67}	$19,166$	$27,614$
κ_{67}	$-0,520$	$0,149$
$\mathbf{b}_{67}$	$\begin{pmatrix} 0,460 \\ 0,007 \\ -0,883 \end{pmatrix}$	$\begin{pmatrix} 0,687 \\ 0,000 \\ -0,727 \end{pmatrix}$

6.3 Nahbereichsszene mit eben begrenzten Objekten

Hier wurde wiederum der EO Device eingesetzt. Als Objekte dienten einige eben begrenzte Objekte wie z. B. Bücher. Wieder wurde mit stark variierenden Orientierungen gearbeitet, der Aufnahmeabstand blieb aber in etwa bei allen Aufnahmen gleich. Insgesamt wurden fünf Aufnahmen von der Szene gemacht. Der Aufnahmestandpunkt des Bildes DSC0001 wurde als Referenz gewählt. Die von der robusten Schätzung befundenen Punktkorrespondenzen sind in den Abbildungen 6.14 bis 6.16 dargestellt.

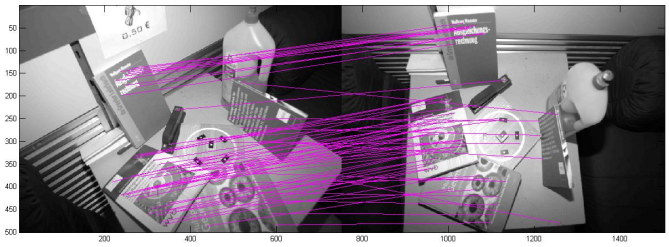


Abbildung 6.14: Homologe Punktepaare der Bilder DSC0001 und DSC0002 nach der robusten Schätzung.

DSC0001–DSC0002.

Größe	MATLAB	ORPHEUS
ω_{12}	$-32,823$	$-33,197$
φ_{12}	$-0,860$	$-1,504$
κ_{12}	$41,584$	$40,761$
$\mathbf{b}_{12}$	$\begin{pmatrix} 0,167 \\ 0,979 \\ -0,115 \end{pmatrix}$	$\begin{pmatrix} 0,172 \\ 0,976 \\ -0,135 \end{pmatrix}$

DSC0001–DSC0003. Die errechneten Werte wichen so stark von den Sollwerten der Bündelblockausgleichung ab, dass sie hier nicht angeführt werden. Dies könnte daran liegen, dass die als korrekt erkannten korrespondierenden Punkte fast in einer Ebene, also in einer für die Berechnung der Fundametalmatrix gefährlichen Fläche, lagen.

DSC0001–DSC0004.

Größe	MATLAB	ORPHEUS
ω_{14}	$-75,538$	$-77,068$
φ_{14}	$-7,9243$	$-4,9016$
κ_{14}	$98,588$	$101,150$
$\mathbf{b}_{14}$	$\begin{pmatrix} 0,189 \\ 0,731 \\ -0,656 \end{pmatrix}$	$\begin{pmatrix} 0,148 \\ 0,797 \\ -0,621 \end{pmatrix}$

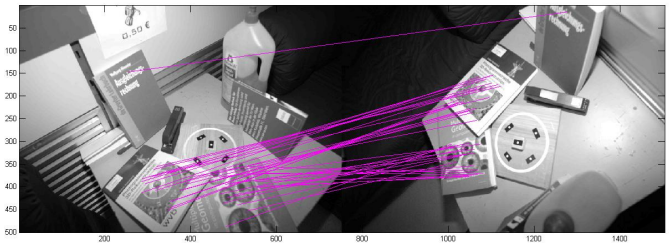


Abbildung 6.15: Homologe Punktepaare der Bilder DSC0001 und DSC0004 nach der robusten Schätzung.

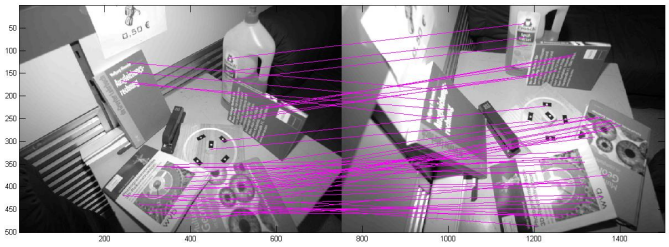


Abbildung 6.16: Homologe Punktepaare der Bilder DSC0001 und DSC0005 nach der robusten Schätzung.

DSC0001–DSC0005.

Größe	MATLAB	ORPHEUS
ω_{15}	$-24,709$	$-25,189$
φ_{15}	$-32,676$	$-32,587$
κ_{15}	$-45,029$	$-46,443$
$\mathbf{b}_{15}$	$\begin{pmatrix} -0,737 \\ 0,460 \\ -0,494 \end{pmatrix}$	$\begin{pmatrix} -0,711 \\ 0,458 \\ -0,534 \end{pmatrix}$

Kapitel 7

Zusammenfassung und Ausblick

Zusammendfassend lässt sich sagen, dass das in dieser Arbeit präsentierte Verfahren zwar geeignet ist, eine Relative Orientierung zwischen zwei oder mehreren Bildern in der Form herzustellen, dass entsprechende Näherungswerte für weitergehende Berechnungen geliefert werden, z. B. für eine Bündelblockausgleichung. Es waren jedoch einige kritische Konfigurationen zu erkennen:

- Große Bereiche mit sprunghaften Kontraständerungen in den einzelnen Bildern bzw. zwischen den Bildern, z. B. Schlagschatten,
- zu regelmäßige, sich wiederholende Muster, wie etwa eine regelmäßige Fassadenstruktur,
- einzelne Ebenen, die den Hauptanteil des Überlappungsbereiches der Bilder ausmachen: Dies ist zwar für das Matching der Keypoints optimal, jedoch ist eine einzelne Ebene auch eine gefährliche Fläche für die Berechnung der Fundamentalmatrix,
- zu hohe lokale Beleuchtungsunterschiede, wie etwa durch Scheinwerfer.

Die Keypoint Descriptors sind jedoch relativ unempfindlich gegenüber Rotationen und Maßstabsunterschieden.

Die aufgezeigte fehlende Stabilität ließe sich etwa durch die Verwendung von Bildtripeln oder Bildquadrupeln (*Trifokaler* bzw. *Quadrifokaler Tensor* [Ressl, 2003; Hartley und Zisserman, 2001]) statt von Bildpaaren erreichen. Auch bei diesen Verfahren ist eine abschließende Bündelblockausgleichung zweckmäßig [Rodehorst, 2004].

Literaturverzeichnis

W. Förstner. Real-Time Photogrammetry. 10 S., 2005.

R. C. Gonzalez, R. E. Woods, und S. L. Eddins. *Digital Image Processing using MATLAB*. Pearson Education, Inc., Upper Saddle River, New Jersey, 2004.

C. Harris und M. Stephens. A combined Corner and Edge Detector. In *Fourth Alvey Vision Conference, Manchester, UK*, S. 147–151, 1988.

R. I. Hartley und A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, Cambridge, 2001.

H. Havlicek. Projektive Geometrie. Skriptum zur Vorlesung, Institut für Geometrie, Technische Universität Wien, 2003a.

H. Havlicek. Lineare Algebra. Skriptum zur Vorlesung, Institut für Geometrie, Technische Universität Wien, 2003b.

K. Kraus. *Photogrammetrie (Band 1) – Geometrische Informationen aus Photographien und Laserscanneraufnahmen*. Walter de Gruyter, Berlin – New York, 2004.

K. Kraus. *Photogrammetrie (Band 2) – Verfeinerte Methoden und Anwendungen*. Dümmler-Verlag, Bonn, 1997. (Mit Beiträgen von H. Kager und J. Jansa).

J. Liu. A Review of 3D Model Reconstruction from Images. `http://www.cs.man.ac.uk/~liuja/docs/recon.pdf` (Zugriff: 20. Jänner 2006), 2005.

D. G. Lowe. Distinctive Image Features from Scale-Invariant Keypoints. *International Journal of Computer Vision*, 60(2):91–110, 2004.

D. G. Lowe. Object Recognition from local Scale-Invariant Features. In *Proceedings of the International Conference on Computer Vision, Corfu, Greece*, S. 1150–1157, 1999.

W. Niemeier. *Ausgleichungsrechnung*. Walter de Gruyter, Berlin – New York, 2002.

M. Pollefeys, R. Koch, und L. Van Gool. Self-Calibration and metric Reconstruction inspite of varying and unknown intrinsic Camera Parameters. *International Journal of Computer Vision*, 32 (1):7–25, 1999.

C. Ressl. *Geometry, Constraints and Computation of the Trifocal Tensor*. Dissertation, Technische Universität Wien, Institut für Photogrammetrie und Fernerkundung, 2003.

V. Rodehorst. *Photogrammetrische 3D-Rekonstruktion im Nahbereich durch Auto-Kalibrierung mit projektiver Geometrie*. Dissertation, Technische Universität Berlin, Fachgebiet Photogrammetrie und Kartographie, 2004.

C. Zeller und O. Faugeras. Camera Self-Calibration from Video Sequences: the Kruppa Equations revisited. Technischer Bericht, INRIA, Sophia-Antipolis, France, 1996. Research Report 2793.

Anhang A

MATLAB-Sourcecodes

A.1 Hauptroutinen

`sift.m`. Extraktion der Keypoints und der Keypoint Descriptors. Dieses File ruft das Demo-Programm `siftWin32.exe` auf, das von David Lowe auf seiner Homepage zur freien Verfügung gestellt wird (URL: `http://www.cs.ubc.ca/spider/lowe/keypoints`, Zugriff: 7. Juli 2005).

```
% [image, descriptors, locs] = sift(imageFile)
%
% This function reads an image and returns its SIFT keypoints.
%   Input parameters:
%     imageFile: the file name for the image.
%
%   Returned:
%     image: the image array in double format
%     descriptors: a K-by-128 matrix, where each row gives
%  an invariant descriptor for one of the K
%  keypoints.  The descriptor is a vector
%           of 128 values normalized to unit length.
%     locs: K-by-4 matrix, in which each row has the 4 values
% for a keypoint location (row, column, scale,
% orientation).  The orientation is in the range
% [-PI, PI] radians.
%
% Credits: Thanks for initial version of this program to
% D. Alvaro and J.J. Guerrero, Universidad de Zaragoza
% (modified by D. Lowe)

function [image, descriptors, locs] = sift(imageFile)

% Load image
image = imread(imageFile);

% If you have the Image Processing Toolbox, you can
% uncomment the following lines to allow input of color images,
%  which will be converted to grayscale.
```

```
if isrgb(image)
   image = rgb2gray(image);
end

[rows, cols] = size(image);

% Convert into PGM imagefile, readable by "keypoints" executable
f = fopen('tmp.pgm', 'w');
if f == -1
    error('Could not create file tmp.pgm.');
end
fprintf(f, 'P5\n%d\n%d\n255\n', cols, rows);
fwrite(f, image', 'uint8');
fclose(f);

% Call keypoints executable
if isunix
    command = '!./sift ';
else
    command = '!X:\RONCAT_Andreas\dipl\siftDemoV4\siftWin32 ';
end
command = [command ' <tmp.pgm >tmp.key'];
eval(command);

% Open tmp.key and check its header
g = fopen('tmp.key', 'r');
if g == -1
    error('Could not open file tmp.key.');
end
[header, count] = fscanf(g, '%d %d', [1 2]);
if count ~= 2
    error('Invalid keypoint file beginning.');
end
num = header(1);
len = header(2);
if len ~= 128
    error('Keypoint descriptor length invalid (should be 128).');
end

% Creates the two output matrices (use known size for efficiency)
locs = double(zeros(num, 4));
descriptors = double(zeros(num, 128));

% Parse tmp.key
for i = 1:num
%row col scale ori
    [vector, count] = fscanf(g, '%f %f %f %f', [1 4]);
    if count ~= 4
        error('Invalid keypoint file format');
    end
    locs(i, :) = vector(1, :);

    [descrip, count] = fscanf(g, '%d', [1 len]);
```

```
    if (count ~= 128)
        error('Invalid keypoint file value.');
    end
    % Normalize each input vector to unit length
    descrip = descrip / sqrt(sum(descrip.^2));
    descriptors(i, :) = descrip(1, :);
end
fclose(g);
```

keypoints_lowe.m. Extraktion der Keypoints und der Keypoint Descriptors. Es wurde aufgrund zu langer Laufzeiten für die Ergebnisse dieser Arbeit nur begrenzt verwendet.

```
function [Pos, K] = keypoints_lowe(input);
% calculates the Lowe Keypoints of an input image input
%
% Pos ... (n x 2) matrix containing the positions of the
% keypoints (x, y)
% -> right handed coordinate system!!!
%
% K   ... (n x 128) matrix containing the keypoint descriptors


% convert input image to grayscale

I = imread(input);
I = rgb2gray(I);


% double sizing of the input image using bilinear interpolation

[rI cI] = size(I);

I = imresize(I,[2*r 2*c],'bilinear');

% transform pixel values in the range [0;1]

I = mat2gray(im2double(I));


% sigma, number of scales and scale factor as proposed in [Lowe, 2004];

sigma = 1.6;
s = 3; % number of scales for extrema search
k = 2^(1 / s); % scale factor


% create Gaußian Filters

for i=0:5
G{i} = fspecial('gaussian', [3 3], k^i*sigma);
end
```

```
% create convolved images of the 1st pyramid niveau

for i=0:5
L{1,i} = imfilter(I, G{i});
end

% create convolved images of the 2nd pyramid niveau

% take every second row

for i=1:floor(rI/2)
a(i,:) = I(2*i -1,:) % auxiliary variable
end

for j=1:floor(cI/2)
I2(:,j) = a(:,2*j - 1) % base image for second pyramid niveau
end

% convolved images of the 1st pyramid niveau

for i=0:5
L{2,i} = imfilter(I2, G{i});
end


% create convolved images of the 3rd pyramid niveau

% take every second row

[rI2 cI2] = size(I2);

for i=1:floor(rI2/2)
a(i,:) = I(2*i -1,:) % auxiliary variable
end

for j=1:floor(cI2/2)
I3(:,j) = a(:,2*j - 1) % base image for second pyramid niveau
end

% convolved images of the 1st pyramid niveau

for i=0:5
L{3,i} = imfilter(I3, G{i});
end


% create DoG-cell D

for i=1:3 % pyramid niveau
for j=0:4 % scale niveau k^j * sigma
D{i,j} = L{i,j+1} - L{i,j};
end
end
```

```
% search for putative keypoint candidates

l = 0; % number of found extrema

for i=1:3 % pyramid niveau
[rI cI] = size(D{i,1}); % image size of current pyramid niveau
for j=1:3 % scale niveau k^j * sigma
for y=2:(rI - 1) % current row
for x=2:(cI - 1) % current column

% 26-pixel neighbourhood of current pixel

nb(1:3) = [D{i,j-1}(y-1, x-1) D{i,j-1}(y-1, x) D{i,j-1}(y-1, x+1)];
nb(4:6) = [D{i,j-1}(y, x-1) D{i,j-1}(y, x) D{i,j-1}(y, x+1)];
nb(7:9) = [D{i,j-1}(y+1, x-1) D{i,j-1}(y+1, x) D{i,j-1}(y+1, x+1)];

nb(10:12) = [D{i,j}(y-1, x-1) D{i,j}(y-1, x) D{i,j}(y-1, x+1)];
nb(13:14) = [D{i,j}(y, x-1) D{i,j}(y, x+1)];
nb(15:17) = [D{i,j}(y+1, x-1) D{i,j}(y+1, x) D{i,j}(y+1, x+1)];

nb(18:20) = [D{i,j+1}(y-1, x-1) D{i,j+1}(y-1, x) D{i,j+1}(y-1, x+1)];
nb(21:23) = [D{i,j+1}(y, x-1) D{i,j+1}(y, x) D{i,j+1}(y, x+1)];
nb(24:26) = [D{i,j+1}(y+1, x-1) D{i,j+1}(y+1, x) D{i,j+1}(y+1, x+1)];

if D{i,j}(y, x) > max(nb) || D{i,j}(y, x) < min(nb)
l = l+1;
put_KP(l) = [i j y x];
end

end
end
end
end

% design matrix needed for subpixel localisation of key points

A = zeros(9,6); % inizialisation

A(1,:) = [1 1 1 -1 -1 1];
A(2,:) = [0 1 0 0 -1 1];
A(3,:) = [1 1 -1 1 -1 1];
A(4,:) = [1 0 0 -1 0 1];
A(5,:) = [0 0 0 0 0 1];
A(6,:) = [1 0 0 1 0 1];
A(7,:) = [1 1 -1 -1 1 1];
A(8,:) = [0 1 0 0 1 1];
A(9,:) = [1 1 1 1 1 1];

Ninv = inv(A'*A);


% sub-pixel localisation of key points
```

```
[r_put_KP c_put_KP] = size(put_KP);

r_curv = 10;     % maximum ratio between the principal curvatures

m = 0;

for l=1:r_put_KP
% 2nd derivatives at the current pixel location

i = put_KP(l,1);
j = put_KP(l,2);
y = put_KP(l,3);
x = put_KP(l,4);

d_yy = D{i,j}(y-1,x) - 2*D{i,j}(y,x) + D{i,j}(y+1,x);
d_xy = D{i,j}(y-1,x) - 2*D{i,j}(y,x) + D{i,j}(y,x+1);
d_xx = D{i,j}(y,x-1) - 2*D{i,j}(y,x) + D{i,j}(y,x+1);

H = [d_yy d_xy; d_xy d_xx]; % Hessian Matrix

        q = trace(H)^2 / det(H);

% accept current point only as key point if the ratio between the
% principal curvatures is below a certain value

        if ((r_curv + 1)^2 / r_curv) > q
              m = m+1;

                % sub-pixel localisation (as in Rodehorst04, pp. 25-26)

                % normalisation of gray values to the interval [0;1] ... maybe!!!

                w = D{i,j}((y-1):(x-1),(y+1):(x+1));

% w_max = max(w(:,1); w(:,2); w(:,3));

% w = w / w_max;

                bb = [w(1,1) w(1,2) w(1,3)...
                   w(2,1) w(2,2) w(2,3)...
                   w(3,1) w(3,2) w(3,3)];

                % paraboloid parameters

                [a b c d e f] = Ninv * A' * bb;

                % residual displacments

                u = (2*b*d - c*e) / (c^2 - 4*a*b); % column direction (x)
                v = (2*a*e - c*d) / (c^2 - 4*a*b); % row direction (y)

                Keys(m,:) = [i j (y+v) (x+u)]; % pyramid niveau, scale and
                 % position of key point
```

```
end
end

% orientation assignment

[rKeys cKeys] = size(Keys);

k = 2^(1/3);

num_keys_ori = 0;  % number of oriented keypoints

for m=1:rKeys
[oct sig_scale y x] = [Keys(m,1) k^Keys(m,2)*sigma Keys(m,3) Keys(m,4)];

sig = 1.5 * sig_scale; % 1.5 * sig_scale

mag_hist = zeros(1,36); % magnitude histogram (each bin is 10° wide)


for i=(y-4):(y+4)
     for j=(x-4):(x+4)

          % distance to Key Point
         d = sqrt((y-i)^2 + (x-j)^2);

         % weighted magnitude
         mag = sqrt((L{oct, scale}(i+1,j) - L{oct, scale}(i-1,j)^2 + ...
         (L{oct, scale}(i,j+1) - L{oct, scale}(i,j-1))^2) * 1 /...
         (sig^2 * 2 * pi) * exp(-(d^2) / (2* sig^2));

         % orientation (degrees)
         theta = atan2((L{oct, scale}(i,j+1) - L{oct, scale}(i,j-1)),...
         (L{oct, scale}(i+1,j) - L{oct, scale}(i-1,j))) * 180 /...
         pi + 180;

         t = floor(theta/10) + 1;    % histogram class of orientation

         % accumulation of weighted magnitudes
         mag_hist(1,t) = mag_hist(1,t) + mag;
     end
end

[mag_sort indices] = sort(mag_hist);

num_peaks = 1;

% how many peaks are in the histgram (threshold 80% of max. peak)?

while (mag_sort(1,num_peaks+1) / mag_sort(1,indices(1)) > 0.8)
     num_peaks = num_peaks + 1;
end

% calculate the best fitting parabola for each peak
```

```
for i=1:num_peaks
     % "measurements' vector"
     yy[1,1] = mag_hist(1,indices(i));
     yy[1,2] = mag_hist(1,mod(indices(i)-1-1,36)+1);
     yy[1,3] = mag_hist(1,mod(indices(i)+1-1,36)+1);

     % Model matrix
     A(1,:) = [(10*indices(i))^2 10*indices(i) 1];
     A(2,:) = [(10*(indices(i)-1))^2 10*(mod(indices(i)-1-1,36)+1) 1];
     A(3,:) = [(10*(indices(i)+1))^2 10*(mod(indices(i)+1-1,36)+1) 1];


     [a;b;c] = inv(A) * y; % unknowns

ori = b / (2*a);  % final orientation

     yy = a * ori^2 + b * ori + c;    % final magnitude

Keys_ori(k + num_peaks, :) = [oct scale y x ori];
end

num_keys_ori = num_keys_ori + num_peaks;

end

% calculation of key point descriptors

for n=1:num_keys_ori

for i=1:4
for j=1:4
local_descr{i,j} = zeros(1,8); % inizialisation of local descriptor
end
end

[oct scale yK xK ori] = Keys_ori(n,:);

for x = (-8):7
for y =(-8):7

% image coordinates
y_im = yK + y * cos(ori * pi / 180) + x * sin(ori * pi / 180);
x_im = xK - y * sin(ori * pi / 180) + x * cos(ori * pi / 180);

y_im = round(y_im);
x_im = round(x_im);

theta = atan2((L{oct, scale}(y_im,x_im+1) - L{oct, scale}(y_im,x_im-1)),...
(L{oct, scale}(y_im+1,x_im) - L{oct, scale}(y_im-1,x_im))) * 180 / pi...
+ 180 - ori;  % reduced orientation (degrees)

mag = sqrt((L{oct, scale}(y_im+1,x_im) - L{oct, scale}(y_im-1,x_im)^2 +...
(L{oct, scale}(y_im,x_im+1) - L{oct, scale}(y_im,x_im-1))^2); % magnitude
```

```
% the histogram in which the sample magnitude will be entered
i = floor((y + 8) / 4) + 1;
j = floor((x + 8) / 4) + 1;
ori_class = floor(theta / 45) + 1; % histogram class of the entry

dy = abs((y + 8) - ((i - 1)*4 + 2)) / 4; % weighting factors
dx = abs((x + 8) - ((j - 1)*4 + 2)) / 4;
dtheta = abs(theta - floor(theta / 45) * 45 + 22.5) / 45;

w = (1 - dy) * (1 - dx) * (1 - dtheta); % weighting factor

% descriptor entry
local_descr{i,j}(1,ori_class) = local_descr{i,j}(1,ori_class) +...
 mag * w * 1 / (2 * pi * 8) *...
 exp(- (x^2 + y^2)/ (2 * pi * 8^2));

end
end

% position of keypoint in original image;
Pos = [yK xK]*2^(oct - 1) / 2;


% make a (1,128)-vector out of the histograms
for i=1:16
K(n,(8*(i-1) + 1):(8*i)) = local_descr{(floor((i-1)/4) + 1),...
(mod(i-1,4) + 1)}(1,1:8);
end

K(n,:) = K(n,:) / norm(K(n,:)); % produce a unit vector


for i=1:128 % set entries > 0.2 to 0.2
if (K(n,i) > 0.2)
K(n,i) = 0.2
end
end

K(n,:) = K(n,:) / norm(K(n,:)); % produce a unit vector


end
```

`keypointmatch_lowe.m`. Bijektives Matching der Keypoints aus zwei Bildern mittels Nearest Neighbourhood.

```
function m = keypointmatch_lowe(Keys1, Keys2);
% returns the indices of the matched SIFT keyoint pairs
% of the keypoint matrices Keys1 and Keys2.
% A match (e.g. P1-P2) is only accepted P1 is the closest
% neighbour to P2 and vice versa.
```

```
[r1 c1] = size(Keys1);
[r2 c2] = size(Keys2);

% find nearest neighbour to all keypoints in the 1st image
for i=1:r1

    P = Keys1(i,:);           % current keypoint

    P_m = ones(r2,1) * P;    % build a matrix whose lines contain P

    D = P_m - Keys2;          % difference vectors

    DT = D';                 % transpose of D

    dist = sqrt(diag(D*DT));    % distances to P

    [sort1,IX] = sort(dist);     % sorted distances and resp. indices

% if sort1(2) > 1.1 * sort1(1)
    m1(i) = IX(1);
%else
%    m1(i) = -1;
%end
end


% find nearest neighbour to all kepoints in the 2nd image
for i=1:r2

    P = Keys2(i,:);           % current keypoint

    P_m = ones(r1,1) * P;    % build a matrix whose lines contain P

    D = P_m - Keys1;          % difference vectors

    DT = D';                 % transpose of D

    dist = sqrt(diag(D*DT));    % distances to P

    [sort1,IX] = sort(dist);     % sorted distances and resp. indices
%if sort1(2) > 1.1 * sort1(1)
     m2(i) = IX(1);
%else
%    m2(i) = -1;
%end
end


% find corresponding minima

num_eq = 0;
for i=1:r1
    index = find(m2==i);
    b = find(index == m1(i));
```

```
    x = size(b);
    if (x > 0) & (b(1) ~= -1)
        num_eq = num_eq+1;

        m(num_eq,:) = [i m1(i)];
    end
end


% remove double entries resulting from multiple keypoints

i = 1;

while i < size(m,1)
    if m(i,:) == m(i+1,:)
        m(i,:) = [];
    else
        i = i+1;
    end
end
```

rel_ori.m. Berechnung der relativen Orientierung zweier Bilder aus gegebenen Punktkorrespondenzen und Kalibrierungsmatrizen. Die Fundamentalmatrix wird mittels RANSAC geschätzt und daraus die Orientierungsparamater abgeleitet.

```
function [RR2, ZZ2, Pic1corr,Pic2corr] =...
 rel_ori(Pic1, Pic2, C1, C2, threshold,p)
%
% at first, the fundamental matrix F is calculated by
% the point correspondences given by the matrices
% Pic1 and Pic2.
% If less then 8 point correspondences are given,
% then the result is F = zeros(3,3).
%
% C1, C2 ... calibration matrices
%
% threshold ... defines the maximum distance a point
% might have from its epipolar line to be an inlier.
%
% p ... propability that an 8-point-sample contains
% only inliers (default: p = 0.99)
%
% If 8 or more correspondences are given, the
% (overdetermined) normalised linear 8-point-algorithm
% is used to estimate the F-matrix, which is
% de-normalised afterwards and then the best fitting
% F-matrix is calculated using RANSAC.
%
% After that, the essential matrix is calculated
% and the rotational matrix RR2 and the base vector
% ZZ2 are returned plus a list of the corresponding
```

```
% points from the first and the second picture.
%
% Pic1corr, Pic2corr ... corresponding point pairs
% of the frist resp. second image whose
% correspondence was accepted by the robust estimation


if nargin < 6       % set default values
    p = 0.99;
    if nargin <5
        threshold = sqrt(3.84) / 3;
    end
end




numbPoints=length(Pic1);
[r1 c1] = size(Pic1);
if c1 ~= 3
Pic1=[Pic1 ones(numbPoints,1)];
Pic2=[Pic2 ones(numbPoints,1)];
end

% Normalisation of image coordinates
CoG=mean(Pic1);
STD=Pic1-kron(ones(numbPoints,1),CoG);
m=sqrt(2/trace(STD'*STD)*numbPoints);
T1=m*[1 0 -CoG(1)
      0 1 -CoG(2)
      0 0      1/m];

CoG=mean(Pic2);
STD=Pic2-kron(ones(numbPoints,1),CoG);
m=sqrt(2/trace(STD'*STD)*numbPoints);
T2=m*[1 0 -CoG(1)
      0 1 -CoG(2)
      0 0      1/m];

%  H1=eye(3);H2=eye(3);

Pic1norm = [T1*Pic1']';
Pic2norm = [T2*Pic2']';



% normalisation complete
% ======================%



% inizialisation of F
F = zeros(3,3);
```

```
%if (r1 == r2) & (r1 > 6) % otherwise no F-matrix can be calculated

% use the normalised 8-point-algorithm and RANSAC
  if numbPoints < 180
            N = factorial(numbPoints) / ...
             (factorial(numbPoints - 8) * factorial(8));
        else
            N = 1;
        end

max_inliers = 8; % maximum number of inliers found by a sample
sample_count = 0;

        max_inliers = 8;

while N > sample_count
  sample_size = 0;
            PointSet = zeros(1,8);

            while sample_size < 8
            % select a point randomly
                newPoint = mod(floor(rand(1)*r1^2),r1)+1;

                % does the point set already contain this point?
                b = find(PointSet == newPoint);
                [rb cb] = size(b);

                if cb ==0
                    sample_size = sample_size + 1;
                    PointSet(1,sample_size) = newPoint;
                end
            end

            PointSet = sort(PointSet);

            if sample_count == 0
                best_inliers = PointSet;
            end


            % system matrix
            A=zeros(8,9);
            for i=1:8,
                for p1coord=1:3,
                    for p2coord=1:3,
                        A(i,(p2coord-1)*3+p1coord) =...
                        Pic1norm(PointSet(i),p1coord) *...
                          Pic2norm(PointSet(i),p2coord);
                    end
                end
            end

        % get f via the SVD
```

```
            [U, D, V] = svd(A);

f = V(:,9);

      Fnorm(1,:) = [f(1) f(4) f(7)]; % building Fnorm out of the elements of f
Fnorm(2,:) = [f(2) f(5) f(8)];
Fnorm(3,:) = [f(3) f(6) f(9)];

            % denormalisation of F
            F = T1' * Fnorm * T2;

[U, D, V] = svd(F); % bring up the singularity constraint using SVD

DD = D;
            DD(3,3)= 0;

F = U * DD * V';

n = 8; % number of inliers (the 8 points of which F was computed)

inliers = PointSet;

for i=1:r1
 % does the point set already contain this point?
                b = find(inliers == i);
                [rb cb] = size(b);

                if cb ==0
                    new_index = i;

                    x_new = Pic1(new_index,1) / Pic1(new_index,3);
                    y_new = Pic1(new_index,2) / Pic1(new_index,3);

                    xx_new = Pic2(new_index,1) / Pic2(new_index,3);
                    yy_new = Pic2(new_index,2) / Pic2(new_index,3);

                    % epipolar line to the new point
                    l = F' * [x_new;y_new;1] ;

                    P_curr = [Pic2(new_index,1);
                     Pic2(new_index,2);
                     Pic2(new_index,3)];

                    % distance of point in picture 2 to epipolar line (HNF)
                    dist = dist2D(P_curr, l);

                    if dist < threshold
                        n = n + 1; % number of inliers
                        inliers(n) = new_index;
                    end

                    inliers = sort(inliers);
                end
```

```
            end

sample_count = sample_count + 1;

if mod(sample_count,1000) == 0
                    sample_count
                    N
                    max_inliers
                end


if n >= max_inliers
                    max_inliers= length(inliers);
best_inliers = inliers;

                    epsilon = 1 - n / r1   % proportion of outliers

                    if epsilon ~=0
                    N = round(log(1 - p) / log(1 - (1 - epsilon)^8))

                    % set of points with most inliers so far
                    else
                   N =0;
end

                   % sample_count = sample_count + 1
        end




end
%end

% continue with LMS of F using all points in best_inliers
% if sample_count == max_sample_count
%     'Warning: sample_count '
[rbestInliers cbestInliers] = size(best_inliers);

'ratio of inliers'
cbestInliers / r1

for i=1:cbestInliers
    Pic1corr(i,:) = Pic1(best_inliers(i),:);
    Pic2corr(i,:) = Pic2(best_inliers(i),:);

   for p1coord=1:3,
      for p2coord=1:3,
         A(i,(p2coord-1)*3+p1coord) = ...
          Pic1norm(best_inliers(i),p1coord)...
          * Pic2norm(best_inliers(i),p2coord);
      end
   end
```

```
end


[U,D,V] = svd(A);

f = V(:,9); % the vector f is the right nullspace of A

F1norm=[f(1:3) f(4:6) f(7:9) ];

% denormalisation of F
F = T1' * F1norm * T2;

[U, D, V] = svd(F); % bring up the singularity constraint using SVD

DD = D;
DD(3,3)= 0;

F = U * DD * V';

% essential matrix
E12=C1'*F*C2;


[U,S,V]=svd(E12);


% relative rotational matrix and right sign for the base line
ZZ2=U(:,3);

W=[0 -1 0;
    1 0 0;
    0 0 1];

R2a=U*W*V';
R2a=R2a/det(R2a);
R2b=U*W'*V';
R2b=R2b/det(R2b);

x1=[Pic1(best_inliers(1),:)]';
x2=[Pic2(best_inliers(1),:)]';
sa=[sign(axi(ZZ2)*C1*x1)]'*sign(axi(ZZ2)*R2a*C2*x2);
sb=[sign(axi(ZZ2)*C1*x1)]'*sign(axi(ZZ2)*R2b*C2*x2);

if sign(sa)==1
   RR2=R2a;
else
   RR2=R2b;
end

% overdetermined system, scale for C1*x1 is set to +1
s=[C1*x1 -RR2*C2*x2]\ZZ2;
if s(1)<0
   ZZ2=-ZZ2;
end
```

```
% normalisation of the base vector
s=norm(ZZ2);
ZZ2=ZZ2/s;
```

proj_depth.m. Anpassung der Basisvektoren zwischen den Projektionszentren auf einen gemeinsamen Maßstab.

```
function lambda = proj_depth(b,R)
% returns the scale factors lambda of a projectively
% reconstructed scene where b is a double indexed cell
% array containing the base vectors and R is a cell containing
% the rotational matrices of the resp. projections centres.

l = 1; % number of equations

n = size(R,2)

for i = 3:n
    for j = 1:(i-2)
        for k = (j+1):(i-1)

            if ~(j== 1 & k ==2)
                A((3*(l-1)+1):3*l, (k-1)*(k-2)/2 + j - 1) =...
                  R{j}' * b{j,k};
                A((3*(l-1)+1):3*l, (i-1)*(i-2)/2 + k - 1) = ...
                  R{k}' * b{k,i};
                A((3*(l-1)+1):3*l,(i-1)*(i-2)/2 + j - 1) = ...
                 -R{j}' * b{j,i};
            else
                A((3*(l-1)+1):3*l, (i-1)*(i-2)/2 + k - 1) = ...
                 R{k}'  * b{k,i};
                A((3*(l-1)+1):3*l, (i-1)*(i-2)/2 + j - 1) = ...
                 -R{j}' * b{j,i};
                bb((3*(l-1)+1):3*l,1) = -b{1,2};
            end
               l = l+1;
        end
    end
end

X = inv(A'*A) *A' * bb;

numx = n*(n-1)/2 - 1; % number of unknowns

j = 1;
k = 3;
for i = 1:numx
    lambda{j,k} = X(i);

    if j < (k-1)
        j = j+1;
    else
```

```
        j = 1;
        k = k+1;
    end
end
```

A.2 Hilfsroutinen

thinout.m. Reduktion der Zeilen einer Matrix auf einen Faktor $\frac{1}{k}$.

```
function t = thinout(A, k)
% takes every k-th row of a matrix A and builds a matrix t from
% it with div(size(A,1)) rows


n = 1;

for i = 1:size(A,1)
    if mod(i-1,k) == 0
        t(n,:) = A(i,:);
        n = n+1;
    end
end
```

axi.m. Berechnung des Axiators eines (3×1)-Vektors.

```
function h=axi(a)
% returns a 3x3 matrix Sa such that cross(a,b)=Sa*b
% courtesy of CaR


h=zeros(3);
z= size(a);
if z(1)*z(2)==3
        h(3,2)=a(1);
        h(3,1)=-a(2);
        h(2,1)=a(3);
        h(1,2)=-h(2,1);
        h(1,3)=-h(3,1);
        h(2,3)=-h(3,2);
end
```

showmatch.m. Anzeigen der korrespondierenden Punkte zweier Bilder nach dem Matching; überarbeitete Version des Files auf der Homepage von David Lowe (URL siehe oben).

```
function num = showmatch(im1, des1, loc1, im2, des2, loc2)
%
% This function reads two images and SIFT features, and
```

```
%   displays lines connecting the matched keypoints.
% A match is accepted if it is a bijective minimum over
% the distances to all keypoints in the resp. other picture.

% It returns the number of matches displayed.
%
% Example: mymatch('scene.pgm','book.pgm');

% with courtesy to David G. Lowe who wrote the original
% version of this file


m = keypointmatch_lowe(des1, des2);

% Create a new image showing the two images side by side.
im3 = appendimages(im1,im2);

% Show a figure with lines joining the accepted matches.
figure('Position', [100 100 size(im3,2) size(im3,1)]);
colormap('gray');
imagesc(im3);
hold on;
cols1 = size(im1,2);

[rm cm]  = size(m);
for i = 1: rm
  if (m(i) > 0)
    line([loc1(m(i,1),2) loc2(m(i,2),2)+cols1], ...
         [loc1(m(i,1),1) loc2(m(i,2),1)], 'Color', 'c');
  end
end
hold off;
num = sum(m > 0);
fprintf('Found %d matches.\n', num);
```

`show_correctkeys.m`. Modifizierte Version von `showmatch.m` zur Anzeige der von der robusten Schätzung akzeptierten Korrespondenzen.

```
function num = show_correctkeys (im1,im2,Pic1,Pic2)
% displays the corresponding keypoint pairs of two images
% whose correspondence was by the orientation algroithm

% please note: this file works independently of the orientation
% algorithm!!!

im3 = appendimages(im1,im2);

% Show a figure with lines joining the accepted matches by RANSAC.
figure('Position', [100 100 size(im3,2) size(im3,1)]);
colormap('gray');
imagesc(im3);
hold on;
cols1 = size(im1,2);
```

```
for i = 1: size(Pic1,1)

    line([Pic1(i,1) Pic2(i,1)+cols1], ...
         [Pic1(i,2) Pic2(i,2)], 'Color', 'm');

end
hold off;
num = size(Pic1,1);
fprintf('Found %d matches.\n', num);
```

`preparematch4rel_ori.m`. Koordinatentransformation von MATLAB-Bildkoordinaten in ein Rechtskoordinatensystem.

```
function [Pic1, Pic2] = preparematch4rel_ori(m, locs1, locs2)
% takes matched point pairs and turns their MATLAB image
% coordinates to right handed ones (the origin is in the
% upper left corner of the image, the positive x-axis looks
% to the right hand side).
%
% the homolgous points are saved in Pic1 and Pic2, respectively.
%
%

n = size(m,1);  % number of matched points

for i=1:n

    % transform coordinates to a right handed system

    % transform x-coordinate of first point
    x = locs1(m(i,1),2) - 0.5;
    % transform y-coordinate of first point
    y = -locs1(m(i,1),1) - 0.5;

    Pic1(i,:) = [x y];

    % transform x-coordinate of second point
    xx = locs2(m(i,2),2) - 0.5;
    % transform y-coordinate of second point
    yy = -locs2(m(i,2),1) - 0.5;

    Pic2(i,:) = [xx yy];

end
```

`dist2D.m`. Distanz eines Punktes (gegeben durch seine homogenen ebenen Koordinaten) zu einer ebenfalls in homogenen Koordianten gegebenen Geraden in der Ebene.

```
function d = dist2D(P,l)
```

```
% calculates the euclidian distance of a projective point in a plane
% given by the column vector P and a projective line given by the
% column vector l

[rP cP] = size(P);

% make point and line inhomogenous

xP = P(1) / P(3);
yP = P(2) / P(3);

l = l / (sqrt(l(1)^2 + l(2)^2));

n = [l(1)
    l(2)];

d = abs([xP yP] * n + l(3));
```

`rot.m`. Berechnung einer Rotationsmatrix $\mathbf{R}$ aus einem Winkelsatz $(\omega, \varphi, \kappa)$ bzw. (α, ζ, κ).

```
function R = rot(om, phi, ka, option)
% R = rot(om, phi, ka, option) calculates the rotational matrix R
% from the given rotation angle sets (angle unit = gon!)
%
% - omega, phi and kappa (option == 'omfika') or
%
% - alpha, zeta and kappa (option == 'alzeka'), respectively.
%
% Please note: the 'alzeka' rotational matrix follows the
% ORIENT convention
% and does not correspond with the one presented in the
% photogrammetric text books of Karl Kraus!
%
% If no option is given, option == 'omfika' is supposed

if nargin == 3
% an OMFIKA rotational matrix is supposed
    option = 'omfika';
end


R= zeros(3,3);

if option == 'omfika'


    % Drehwinkel im Bogenmaß
    om = om * pi / 200;
    phi = phi * pi / 200;
    ka = ka * pi / 200;
```

```
    R(1,1) = cos(phi) * cos(ka);
    R(1,2) = - cos(phi) * sin(ka);
    R(1,3) = sin(phi);

    R(2,1) = cos(om) * sin(ka) + sin(om) * sin(phi) * cos(ka);
    R(2,2) = cos(om) * cos(ka) - sin(om) * sin(phi) * sin(ka);
    R(2,3) = - sin(om) * cos(phi);

    R(3,1) = sin(om) * sin(ka) - cos(om) * sin(phi) * cos(ka);
    R(3,2) = sin(om) * cos(ka) + cos(om) * sin(phi) * sin(ka);
    R(3,3) = cos(om) * cos(phi);

else
    if option == 'alzeka'

        % Drehwinkel im Bogenmaß
        al = om * pi / 200;
        ze = phi * pi / 200;
        ka = ka * pi / 200;

        R(1,1) =  cos(al) * cos(ze)* cos(ka) - sin(al) * sin(ka);
        R(1,2) = -cos(al) * cos(ze)* sin(ka) - sin(al) * cos(ka);
        R(1,3) =  cos(al) * sin(ze);

        R(2,1) =  sin(al) * cos(ze) * cos(ka) + cos(al) * sin(ka);
        R(2,2) = -sin(al) * cos(ze) * sin(ka) + cos(al) * cos(ka);
        R(2,3) =  sin(al) * sin(ze);

        R(3,1) =  -sin(ze) * cos(ka);
        R(3,2) =  sin(ze) * sin(ka);
        R(3,3) =  cos(ze);
    else
        % illegal option is given
        R = 0;
    end
end
```

`omfika.m`. Berechnung der zwei Winkelsätze $(\omega_1, \varphi_1, \kappa_1)$ bzw. $(\omega_2, \varphi_2, \kappa_2)$ aus einer gegebenen Rotationsmatrix $\mathbf{R}$.

```
function [omega1, phi1, kappa1, omega2, phi2, kappa2] = omfika(R)
% This function returns the two possible rotation
% angles sets consisting of omega, phi and kappa
% from a given rotational matrix R.


% phi1, phi2
phi1 = asin(R(1,3));
phi2 = pi - phi1;

if phi2 < 0
    phi2 = phi2 + 2*pi;
end
```

```
% kappa1
kappa1 = atan2(-R(1,2),R(1,1));


% kappa2
if kappa1 < 0
    kappa2 = kappa1 + pi;
else
    kappa2 = kappa1 - pi;
end

% omega1
omega1 = atan2(-R(2,3),R(3,3));



% omega2
if omega1 < 0
    omega2 = omega1 + pi;
else
    omega2 = omega1 - pi;
end


% Umrechnung nach gon
omega1 = omega1 * 200 / pi;
omega2 = omega2 * 200 / pi;

phi1 = phi1 * 200 / pi;
phi2 = phi2 * 200 / pi;

kappa1 = kappa1 * 200 / pi;
kappa2 = kappa2 * 200 / pi;
```

`alzeka.m`. Berechnung der zwei Winkelsätze $(\alpha_1, \zeta_1, \kappa_1)$ bzw. $(\alpha_2, \zeta_2, \kappa_2)$ aus einer gegebenen Rotationsmatrix $\mathbf{R}$.

```
function [alpha1, zeta1, kappa1, alpha2, zeta2, kappa2] = alzeka(R)
%
% This function returns the two possible rotation
% angles sets consisting of alpha, zeta and kappa
% from a given rotational matrix R.


% zeta1, zeta2
zeta1 = acos(R(3,3));
zeta2 = 2 * pi - zeta1;

if zeta2 < 0
    zeta2 = zeta2  + 2*pi;
end
```

```
% kappa1
kappa1 = atan2(R(3,2),-R(3,1));


% kappa2
if kappa1 < 0
    kappa2 = kappa1 + pi;
else
    kappa2 = kappa1 - pi;
end

% alpha1
alpha1 = atan2(R(2,3),R(1,3));



% alpha2
if alpha1 < 0
    alpha2 = alpha1 + pi;
else
    alpha2 = alpha1 - pi;
end


% Umrechnung nach gon
alpha1 = alpha1 * 200 / pi;
alpha2 = alpha2 * 200 / pi;

zeta1 = zeta1 * 200 / pi;
zeta2 = zeta2 * 200 / pi;

kappa1 = kappa1 * 200 / pi;
kappa2 = kappa2 * 200 / pi;
```

Lebenslauf

Persönliche Daten

Name:	Andreas Roncat
Geburtsdatum:	4. Oktober 1981
Geburtsort:	Kufstein (Tirol)
Staatsbürgerschaft:	Österreich
Eltern:	Helga Roncat, geb. Walch (VS-Oberlehrerin), Alois Roncat (HS-Oberlehrer i. R.)
Schwester:	Mag.phil. Eva Roncat

Bildungsweg

1987-1991	Volksschule Ebbs
1991-1999	Bundesgymnasium Kufstein (Neusprachlicher Zweig)
26. Juni 1999	Reifeprüfung mit Auszeichnung bestanden
seit WS 2000/01	Diplomstudium Vermessung und Geoinformation an der TU Wien
28. 1. 2003	1. Diplomprüfung abgelegt
seit WS 2001/02	Lehramtsstudium Informatik und Darstellende Geometrie an der TU Wien und der Universität Wien

Tabellenverzeichnis

3.1 Klasseneinteilung der Antwortfunktion $R(x, y)$ des Harris-Operators 10

4.1 Hyperebenen λ in projektiven Räumen $\mathcal{P}$ verschiedener Dimension n. 21

5.1 Adaptive Bestimmung der RANSAC-Parameter (in Pseudocode), nach [Hartley und Zisserman, 2001]. 34

5.2 Verschiedene Varianten der Selbstkalibrierung [Pollefeys et al., 1999]. 36

Abbildungsverzeichnis

2.1 Digitales Bild-Koordinatensystem. 4
2.2 Schematische Darstellung einer Bildpyramide. 7

3.1 Der Scale-Space eines Bildes [Lowe, 2004]. 11
3.2 26er-Nachbarschaft für lokale Extremwert-Ermittlung [Lowe, 2004]. 12
3.3 Orientierungshistogramm und Parabelanpassung der Orientierung(en) eines Keypoints. In diesem Beispiel entstehen an der gleichen Stelle im Bild drei Keypoints mit unterschiedlicher Orientierung. . . . 14
3.4 Keypoints mit Magnitude und Orientierung. 15
3.5 Grafik zur Visualisierung der Berechnung von $\overline{m_i}$ für ein Pixel aus der (16×16)-Umgebung eines Keypoints K. 16
3.6 Beispiel für einen Keypoint Descriptor. Der Einfachheit halber wird ein (8×8)-Fenster um den Keypoint betrachtet, das in vier (4×4) Bereiche unterteilt wird, für die je ein Orientierungshistogramm mit $45°$ Klassenbreite erstellt wird. Für den tatsächlichen Keypoint Descriptor wird ein (16×16)-Fenster herangezogen, das in 16 (4×4)-Bereiche unterteilt wird. [Lowe, 2004]. 16

4.1 Projektive Minimalebene als Variante des Wiener U-Bahnnetzes (© by Hans Havlicek) 19
4.2 Die projektiv abgeschlossene Anschauungsebene $\mathcal{P}^2$ in den $\mathbb{R}^3$ eingebettet [Ressl, 2003]. 21
4.3 Kollineation (Homographie) $\kappa : \varepsilon \to \varepsilon'$ in der Ebene. Kollineare Punkte gehen unter κ in kollineare Punkte über. 22
4.4 Korrelation (duale Kollineation) $\kappa^* : \varepsilon \to \varepsilon^*$ in der Ebene. Kollineare Punkte gehen dabei in kopunktale Geraden über. 23
4.5 Wechsel des Bildkoordinatensystems vom Linkssystem (x, y) auf das Rechtssystem (x', y'). 24
4.6 Abbildung von Punkten und Geraden unter einer Perspektive mit Projektionszentrum Z. Die Rotationsmatrix $\mathbf{R}$ transformiert vom Objekt- in das Kamerakoordinatensystem. 25

5.1 Relative Orientierung und Epipolarbedingung für zwei Bilder. . . 28

6.1 Genäherter Normalfall. 41
6.2 Aufnahmen vom EO Device mit 10° konvergenten Hauptstrahlen. 42
6.3 Aufnahmen mit vom EO Device 30° konvergenten Hauptstrahlen. 43
6.4 Homologe Punktepaare der Bilder DSC0002 und DSC0003. 44
6.5 Homologe Punktepaare der Bilder DSC0003 und DSC0005. 45
6.6 Homologe Punktepaare der Bilder DSC0003 und DSC0006. 46
6.7 Homologe Punktepaare der Bilder DSC0003 und DSC0007. 47
6.8 Homologe Punktepaare der Bilder DSC0004 und DSC0005. 47
6.9 Homologe Punktepaare der Bilder DSC0004 und DSC0006. 48
6.10 Homologe Punktepaare der Bilder DSC0004 und DSC0007. 49
6.11 Homologe Punktepaare der Bilder DSC0005 und DSC0006. 50
6.12 Homologe Punktepaare der Bilder DSC0005 und DSC0007. 50
6.13 Homologe Punktepaare der Bilder DSC0006 und DSC0007. 51
6.14 Homologe Punktepaare der Bilder DSC0001 und DSC0002 nach der robusten Schätzung. 52
6.15 Homologe Punktepaare der Bilder DSC0001 und DSC0004 nach der robusten Schätzung. 53
6.16 Homologe Punktepaare der Bilder DSC0001 und DSC0005 nach der robusten Schätzung. 53