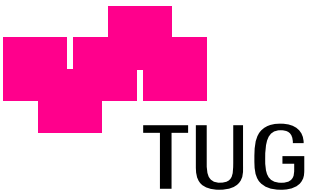
Magisterarbeit

# Dünne 3D Rekonstruktion eines Raumes

# Sparse 3D-reconstruction of a room

Gerald Schweighofer

Institut für Elektrische Meßtechnik und Meßsignalverarbeitung
Technische Universität Graz




Betreuer und Begutachter:
Ao.Univ.-Prof.Dipl.-Ing.Dr.techn. Axel Pinz

Graz, im November 2003

# Kurzfassung

Es sind viele Möglichkeiten bekannt, um komplette 3D-Modelle zu erstellen. Diese haben jedoch einen sehr hohen Rechenaufwand. In dieser Arbeit wird die Erzeugung eines "dünnen" 3D-Modelles eines Raumes erläutert, was in sehr kurzer Zeit möglich ist. Grundlage hierzu ist die Erweiterung der *Stereo* Rekonstruktion auf Panorama-Bilder. Zur Kalibrierung der Kamera dient ein Algorithmus, der die Eigenschaften der Bildaufnahme ausnutzt und somit das Kamerazentrum und das Rotationszentrum identisch sind. Zur Berechnung der äußeren Orientierung werden selbst entwickelte Sub-Pixel genaue Kalibrier-Targets verwendet. Rekonstruiert werden korrespondierende *Plessey-Corner*. Experimente zeigen die zu erreichende Genauigkeit des Systems.

# Abstract

There are many known ways in producing complete 3D models. But these have one disadvantage: the long computation time. In this work we describe the creation of a sparse 3D model of a room, which is possible in a short time. To do this we extended the stereo reconstruction on panoramic images. To calibrate the camera we used an algorithm which uses the properties of the image creation. Here the rotation center and the camera center are identical. To calculate the outer orientation we used self developed sub pixel accuracy calibration targets. To find correspondent corners in both images we used a plessey-corner detector. Experiments show us the obtained accuracy of the reconstructed 3D model.

# Inhaltsverzeichnis

# Abbildungsverzeichnis

# Kapitel 1

# Einleitung

In diesem ersten Kapitel soll zuerst das Problem erläutert, danach die grundlegenden mathematischen Zusammenhänge, die zur Lösung dieses Problems vonnöten sind, erklärt, sowie die bereits in der Literatur beschriebenen Möglichkeiten besprochen werden.

## 1.1 Problemstellung

Das Ziel dieser Arbeit war es ein System zu entwickeln, welches Punkte im gesamten Raum (Bildmerkmale) lokalisiert und die Koordinaten dieser ausgewählten Punkte berechnet.
Das System sollte weiters folgende Eigenschaften besitzen:

- Einfache Bedienbarkeit.
- Erfassung eines gesamten Raumes.
- Geringer Zeitaufwand zur Lokalisation der Punkte (max. 30 min.).
- Das System sollte autonom arbeiten.
- Generelle Einsetzbarkeit des Systems. Unabhängig von der Raumgröße.

## 1.2 Grundlagen

Die grundlegende mathematische Beschreibung der Abbildung eines Welt-Punktes $X$ mittels einer Digital-Kamera in ein entsprechendes Pixel mit den Koordinaten $p$ im Bild ist durch

$$\lambda p = K\ [R \mid t]\ X \tag{1.1}$$

gegeben. Hierbei wird $p = [x, y, 1]^T$ und $X = [x, y, z, 1]^T$ in normalisierten Koordinaten ausgedrückt. $t$ beschreibt die Position der Kamera. $R$ ist eine Rotationsmatrix,

welche die Orientierung (Blickrichtung) der Kamera wiedergibt.
Bei $K$ handelt es sich um die inneren Eigenschaften der Kamera:

$$K = \begin{bmatrix} f_x & s & x_0 \\ 0 & f_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1.2}$$

$f_x$,$f_y$ beschreiben die *focal-length* (Brennweite) der jeweiligen Achse.
$x_0$,$y_0$ stellen das Projektionszentrum dar. Bei $s$ handelt es sich um die Scherung, welche aber in der Literatur meist mit $s = 0$ angenommen, und auch in dieser Arbeit vernachlässigt wird.
Eine Sonderstellung nimmt der Parameter $\lambda$ ein. Dieser tritt durch die projektive Abbildung auf und entspricht der Entfernung des Punktes $X$ vom Zentrum der Kamera.

## 1.3 Bekannte Verfahren

Das Problem besteht also darin Gleichung 1.1 für Punkte $X$ im Raum durch Beobachten der Punkte $p$ in mehreren Bildern zu lösen. In der Literatur sind hierzu mehrere verschiedene Verfahren zu finden.
Diese Verfahren kann man zunächst einmal in die Anzahl der Bilder, die für eine Rekonstruktion benötigt werden, einteilen.

### Stereo

*Stereo* stellt hier das einfachste Verfahren dar. Hierbei werden zwei Kameras zuerst zueinander fix montiert. Dieser Aufbau wird kalibriert um danach 3D-Rekonstruktion betreiben zu können. Bei der Kalibrierung wird die *Fundamental*-Matrix berechnet ([LF96],[SM97]). Diese stellt einen elementaren Zusammenhang zwischen zwei Bildern dar, der sich aus Gleichung 1.1 ergibt (siehe dazu auch Kapitel 2.2). In [PZ98] wird dieses Verfahren dazu erweitert, dass sich die *Fundamental*-Matrix aus den zwei Bildern ohne vorherige Kalibrierung bestimmen lässt. Ein Problem stellt jedoch das Finden von gleichen (korrespondierenden) Punkten dar.
Da es sich hierbei jedoch nur um zwei Bilder handelt, ist damit der zu rekonstruierende Bereich sehr klein.

### Multiple Views

Um größere Objekte rekonstruieren zu können finden sich in der Literatur auch Verfahren, die mit mehreren Bildern arbeiten. Diese lassen sich in zwei Arten einteilen:

- *Erweitertes Stereo*
- *Faktorisierung*

**Erweitertes *Stereo*** wird meist bei Videoaufnahmen eingesetzt. Das zu rekonstruierende Objekt wird mit einer Videokamera gefilmt. Da sich beim Filmen die Orientierung der Kamera nur sehr langsam ändert, kann bei nahe beieinander liegenden *Frames* der *Stero*-Algorithmus angewendet werden. Korrespondierende Punkte werden hier durch Verfolgen (*tracken*) von Bild-Bereichen über die einzelnen *Frames* gewonnen. Die einzelnen Rekonstruktionen werden dann zu einer gesamten Rekonstruktion fusioniert ([Fit01],[RKG98],[AS98]). Das Problem hier ist jedoch, dass sehr viele *Frames* verarbeitet werden müssen. Dies führt zu einem sehr hohen Zeitaufwand bei der Rekonstruktion.

**Faktorisierung** nennt man die Zerlegung einer Matrix in zwei Faktoren. Bei der Aufnahme mehrerer Punkte von mehreren Kameras kann Gleichung 1.1 in Matrix-Schreibweise gebracht werden:

$$\begin{bmatrix} \lambda_1^1 x_1^1 & \lambda_2^1 x_2^1 & \cdots & \lambda_n^1 x_n^1 \\ \lambda_1^2 x_1^2 & \lambda_2^2 x_2^2 & \cdots & \lambda_n^2 x_n^2 \\ \vdots & \vdots & \ddots & \vdots \\ \lambda_1^m x_1^m & \lambda_2^m x_2^m & \cdots & \lambda_n^m x_n^m \end{bmatrix} = \begin{bmatrix} P^1 \\ P^2 \\ \vdots \\ P^m \end{bmatrix} \begin{bmatrix} X_1 & X_2 & \ldots & X_n \end{bmatrix} \tag{1.3}$$

Die Matrix links beschreibt die gemessenen Koordinaten der Punkte im Bild. $P^j = [R_j \mid t_j]$ beschreibt die Orientierung sowie Position der einzelnen Kameras. Durch Faktorisierung können die Parameter der Kamera, sowie die Position der Punkte im Raum bestimmt werden. Hier entsteht das Problem, dass man die $\lambda_i^j$ nicht messen kann. Diese werden durch iterative Verfahren gewonnen ([HC96],[HK00b],[HAC02] ,[SMP01],[CM99]). Weiter Probleme die hier auftreten sind:

- Alle Punkte müssen in allen Bildern sichtbar sein. Dies ist nur möglich, wenn ein Objekt von nur einer Seite aufgenommen wird.
- Die Rekonstruktion ist nur eine Projektive. D.h. aus der gewonnenen Rekonstruktion muss erst durch zusätzliche Eigenschaften eine euklidische Rekonstruktion erzeugt werden.

### Panorama

Unter Panorama versteht man eine Rundumsicht ($360^o$) von einem bestimmten Ort aus.
In [SP01] wird eine Kamera auf einem Arm montiert und durch einen Motor bewegt. Aus den aufgenommenen Bildern werden jeweils zwei Pixel-Spalten ausgewählt. Durch Drehen des Motors entstehen so zwei Panorama-Bilder. Dank der Analyse der Aufnahme-Geometrie ist es möglich eine Rekonstruktion zu berechnen. Da nur zwei Spalten der Bilder verwertet werden, sind sehr viele Bilder (1500) nötig um ein komplettes Panorama zu erhalten. Damit ist der Zeitaufwand zu hoch.
In [BK02] wurden hyperbolische Spiegel verwendet, die auf der Kamera montiert

waren. Dadurch bekam man nach einer Entzerrung des Bildes ein Panorama der Umgebung. Montiert wurde diese Kamera auf einem mobilen Roboter. Somit waren die Positionen der einzelnen Aufnahmen bekannt, und eine Rekonstruktion wured möglich. Bei Verwendung eines hyperbolischen Spiegels muss dieser jedoch genau an die Kamera angepasst werden, wodurch die Verwendung einer Kamera mit Zoom-Objektiv nicht möglich ist. Dies ist jedoch nötig um unabhängig von der Raumgröße eine Rekonstruktion zu erstellen.

Zusammenfassend ist zu bemerken, dass von den hier vorgestellten Verfahren nur ein Panorama die Rekonstruktion eines gesamten Raumes ermöglicht, die zwei Verfahren jedoch für die geforderte Problemstellung nicht anwendbar sind.

## 1.4 Meine Lösung

Da zur Rekonstruktion eines gesamten Raumes eine Rundumsicht vorhanden sein muss, zur Unabhängigkeit von der Raumgröße jedoch eine Kamera mit Zoom nötig ist, wurden beide Eigenschaften kombiniert.
Hierzu wurde eine Kamera (mit Zoom) auf einer *Pan-Tilt-Unit* (PTU) montiert. Durch Steuern der PTU ist es möglich eine Rundumsicht zu erzeugen.
Aufgrund des speziellen Aufbaus kann eine Kalibrierung der inneren Kamera-Parameter ohne weitere Hilfsmittel durchgeführt werden. Dies ist auch nötig, da der Zoom (und damit die inneren Kamera-Parameter) erst an Ort und Stelle eingestellt wird.
Danach werden an zwei verschiedenen Orten im Raum Rundumsichten (Panorama) erzeugt. Durch diese zwei Ansichten ist dann eine *Stereo*-Rekonstruktion möglich. Zur Berechnung der äußeren Orientierung werden selbst entwickelte Kalibrier-Targets verwendet.

# Kapitel 2

# Bildaufnahme

Ziel dieses Kapitels ist es an einem Standort eine Rundumsicht zu erhalten, mit der dann weiter gearbeitet werden kann. Im Gegensatz zu [SP01] werden hierbei ganze Bilder verwendet. Das hat den Vorteil, dass die Aufnahmen schneller gemacht werden, und die in der Literatur verwendeten Verfahren zur Rekonstruktion verwendet werden können.
Aus den inneren Kamera-Parametern lässt sich der Öffnungswinkel der Kamera $B_\alpha$ berechnen:

$$B_\alpha = 2 \arctan(\frac{2f_x}{x_{max}}) \tag{2.1}$$

$x_{max}$ entspricht hier der maximalen Auflösung der Kamera. Zu diesem Öffnungswinkel wird nun ein Überlapp definiert, welcher angibt, wie viel sich nebeneinander liegende Bilder überlappen sollen. Aus dem maximalen Rotationswinkel der PTU, dem Öffnungswinkel und dem Überlapp lassen sich nun die Anzahl der Bilder, die für die Aufnahme nötig sind, berechnen.

$$Bilder = \lceil \frac{B_\alpha - \alpha_{overlapp}}{PTU_{maxrot}} \rceil \tag{2.2}$$

In Gleichung 2.2 bedeutet $\lceil \rceil$ ein Aufrunden des Wertes innerhalb der Klammern. Dies ist nötig um die volle mögliche Auslenkung der PTU zu nützen. Anschließend kann der Auslenkungswinkel zwischen zwei Bildern durch

$$A_\alpha = \frac{Bilder}{PTU_{maxrot}} \tag{2.3}$$

berechnet werden.

Zur besseren Erklärung ist die Bildaufnahme in Abbildung 2.1 skizziert. Die PTU wird ganz nach links gefahren. Hier wird das erste Bild erstellt (Bild mit Nr. 1). Danach wird die Kamera durch die PTU immer um einen Winkel $A_\alpha$ nach rechts gedreht und eine neue Aufnahme gemacht. Dieser Vorgang wiederholt sich so lange

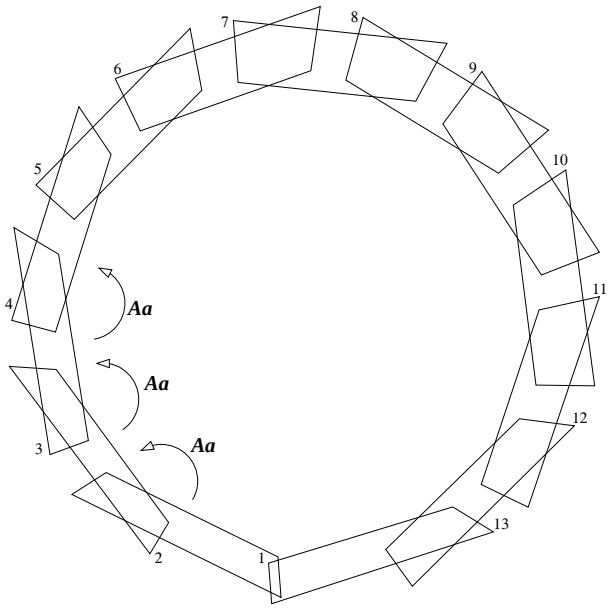



Abbildung 2.1: Bildaufnahme

bis alle Bilder aufgenommen sind. Im Beispiel in Abbildung 2.1 sind dies 13 Bilder. Wie man deutlich sehen kann ist der Überlapp zwischen Bild 1 und Bild 13 kleiner als zwischen den anderen Bildern. Das kommt dadurch, dass die PTU nicht in der Lage ist sich $360^o$ zu drehen.

## 2.1 Kalibrierung

Kalibrierung nennt man den Vorgang zur Bestimmung der inneren und äußeren Kamera-Parameter.
Die inneren Parameter beschreiben die Eigenschaften der Kamera. Diese sind abhängig von der Herstellung und von dem einstellbaren Zoom der Kamera.
Die äußeren Parameter beschreiben die Position und Orientierung der Kamera in einer Szene.
Benötigt werden diese Parameter bei der Aufnahme zur Bestimmung des Rotationswinkels der PTU, sowie zur Ermittlung der Epiploar-Geometrie um Punkte besser *Matchen* zu können.
Bekannte Kalibrier-Algorithmen benutzen ausgemessene Kalibrier-Targets (3D-Targets in [Tsa87], Schachbrettmuster in [Zha99]). Zwischen den gemessenen Punkten im Bild und den bekannten Koordinaten der Targets lassen sich die Kamera-Parameter berechnen. Je nachdem welches Target (3D oder 2D) benutzt wird, werden mindestens eines oder mehrere Bilder unter verschiedenen Blickwinkeln zur Kalibrierung benötigt.

### 2.1.1 Innere Orientierung

In diesem Kapitel wird die Kalibrierung, welche für die weiteren Berechnungen nötig ist, vorgestellt. Zuerst wird ein linearer Algorithmus zur Berechnung der inneren Kamera-Parameter ($f_x$,$f_y$,$x_0$ und $y_0$) vorgestellt, welche dann durch Linsenverzeichnungen ( $kc_0$,$kc_1$ ... $kc_5$ ) erweitert werden. Danach folgt eine nichtlineare Optimierung aller Parameter.

#### Linearer Algorithmus

Unter der Annahme, dass mehrere Bilder mit einer Kamera gemacht werden, die sich zwar in ihrer Rotation, nicht aber in ihrer Position unterscheiden, kann Gleichung 1.1 vereinfacht werden zu:

$$\lambda p_{ij} = K R_j X_i \tag{2.4}$$

Diese Vereinfachung trifft z.B. bei einer Bewegung der Kamera durch eine PTU zu. In Gleichung 2.4 entspricht $p_{ij}$ der Abbildung des Weltpunktes $X_i$ im Bild $j$. D.h. der Punkt $p_{ij}$ entspricht einem Weltpunkt, der sich auf der Geraden $\lambda(KR_j)^{-1}p_{ij}$ befinden muss.Ein Zusammenhang zwischen zwei Bildern besteht durch ([RIHH99]):

$$\lambda p_{ik} = K R_k X_i = K R_k (K R_j)^{-1} p_{ij} = H_{kj} p_{ij} \tag{2.5}$$

Man sieht, dass Punkte in den beiden Bildern $k$ und $j$ durch eine projektive Transformation $H_{kj}$ verknüpft sind. Diese projektive Transformation ist in den Bildern durch Finden von 4 korrespondierenden Punkten berechenbar (siehe Homographie). Aus dem zweiten Teil der Gleichung 2.5 und der Annahme:
$R_j = [[1,0,0]^T, [0,1,0]^T, [0,0,1]^T]$ ergibt sich

$$K R_k K^{-1} p_{ij} = H_{kj} p_{ij} \tag{2.6}$$

$$K R_k = H_{kj} K \tag{2.7}$$

$$K R_k - H_{kj} K = 0 \tag{2.8}$$

Durch Vergleich der einzelnen Koeffizienten in Gleichung 2.8 und dem Wissen um die durchgeführte Rotation (PTU) ergibt sich ein lineares Gleichungssystem, welches nach den Parametern $f_x$,$f_y$,$x_0$ und $y_0$ aus $K$ gelöst werden kann. Bei einer Rotation um nur eine Achse kommen im linearen Gleichungssystem jedoch nicht alle Parameter von $K$ vor. Es sind also mindestens zwei verschiedene Rotationen nötig.

**Implementierung** Zu Beginn werden drei Bilder wie in Abbildung 2.2 erstellt. Zwischen den Bildern $\pi 1$ und $\pi 2$ fand eine Rotation um die y-Achse, zwischen den

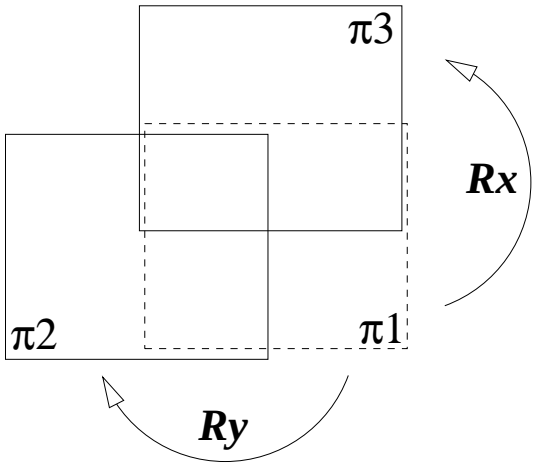



Abbildung 2.2: Bildaufnahme bei Kalibrierung

Bildern $\pi 1$ und $\pi 3$ eine Rotation um die x-Achse statt. Zwischen den jeweiligen Bildern werden nun korrespondierende Punkte gesucht. Dies geschieht wie folgt:

Da die Rotation nicht allzu groß ist (wenige Grad), ändern sich die Positionen der Punkte auch nicht allzu sehr. Deshalb wird für einen Punkt $p_{ij}$ sein korrespondierender Punkt $p_{ik}$ in einem anderen Bild an denselben Koordinaten gesucht. In der Umgebung dieser Koordinaten wird nun der beste *Match* (*SSD* siehe Kapitel 3.3.1) ermittelt. Die Position dieses besten *Matches* entspricht dem korrespondierenden Punkt.

Aus den gefundenen korrespondierenden Punkten wird nun für jede Rotationsrichtung eine Homographie und aus diesen dann die inneren Kamera-Parameter berechnet.

**Homographie** Beim genaueren Betrachten von Gleichung 2.5

$$\lambda \begin{pmatrix} x_{ik} \\ y_{ik} \\ 1 \end{pmatrix} = \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix} \begin{pmatrix} x_{ij} \\ y_{ij} \\ 1 \end{pmatrix} \tag{2.9}$$

sieht man, dass die Vektoren links und rechts vom Gleichheitszeichen sich nur in ihrer Länge $\lambda$ unterscheiden dürfen, so dass das Kreuzprodukt aus beiden 0 sein muss:

$$p_{ik} \times (Hp_{ij}) = 0 \tag{2.10}$$

Dies ergibt zwei Gleichungen pro Punktpaar:

$$[(h_4 x_{ij} + h_5 y_{ij} + h_6) - (h_7 x_{ij} + h_8 y_{ij} + h_9) y_{ik}] = 0 \tag{2.11}$$

$$[(h_7 x_{ij} + h_8 y_{ij} + h_9) x_{ik} - (h_1 x_{ij} + h_2 y_{ij} + h_3)] = 0 \tag{2.12}$$

Da $H$ nur bis auf einen Skalierungsfaktor bestimmt werden kann, besteht $H$ aus acht Unbekannten. Diese sind durch Lösen der Gleichungen 2.11 und 2.12 durch 4 Punktpaare möglich. Da das Bestimmen von Punktpaaren fehleranfällig ist, wurde mit dem *RANSAC*-Algorithmus Robustheit erzielt und diese gefundene Lösung danach noch optimiert (siehe Anhang B.1).

***RANSAC*** (RANdom SAmple Consensus) ist ein Algorithmus, der es ermöglicht robuste Lösungen zu finden. Robuste Lösungen sind dann nötig, wenn in den Daten, aus denen etwas berechnet werden soll, Fehler vorhanden sind. In unserem Fall, nämlich beim Berechnen der Homographie $H$ zwischen zwei Bildern, existieren in den Punktpaaren auch falsche Punktpaare, die eliminiert werden müssen. Würden diese nicht entdeckt werden, würde die nachfolgende Optimierung ein falsches Ergebnis liefern.
Algorithmus:

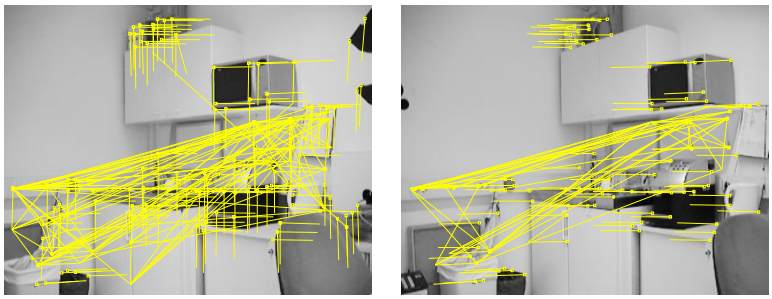
1. Wähle zufällig 4 Punktpaare aus allen Punktpaaren aus.
2. Berechne aus diesen 4 Paaren die Homographie $H$.
3. Teste nun wie gut alle Punktpaare mit dieser Homographie $H$ beschrieben werden können. Dazu berechne den Fehler $\varepsilon_i = dist(Hp_{i1}, p_{i2}) + dist(H^{-1}p_{i2}, p_{i1})$ für alle Paare $i$.
4. Setze alle Paare als *Inlier* deren Fehler $\varepsilon_i$ kleiner einem Schwellwert $T$ ist.
5. Berechne die benötigten Iterationen $N$ mit Gleichung 2.13 aus den *Inliern*.
6. Gehe zu 1 solange bis $N >$ Anzahl Wiederolungen.
7. Lösung für $H$ ist jenes $H$ welches die meisten *Inlier* hat.

Die Berechnung von $N$ geschieht durch:

$$N = \frac{\log(1-p)}{\log(1-w^s)} \tag{2.13}$$

$w$ gibt hier die Wahrscheinlichkeit dafür an, dass ein Punktpaar ein *Inlier* ist. $w = 1 -$ Anzahl Inlier/Anzahl Punktpaare. $N$ ergibt hierbei die Anzahl der Wiederholungen die notwendig sind, um mit einer Wahrscheinlichkeit von $p$ (wurde auf $p = 0.99$ gesetzt) die maximale Anzahl von *Inlier* zu finden. $s$ definiert die Dimension des Problems ($s = 4$ Punktpaare werden zur Berechnung von $H$ benötigt).

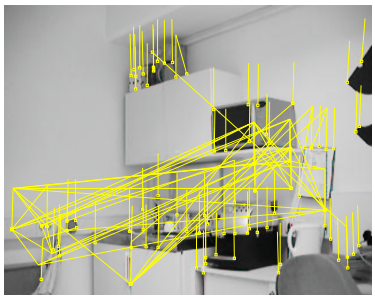
In Abbildung 2.3 ist ein Beispiel dargestellt. Hierzu wurde ein Bild (a) aufgenommen. Danach zwei weitere, bei denen die Kamera um die Y-Achse (b) bzw. um die X-Achse (c) um $5^o$ gedreht wurde. Zwischen (a) und (b) sowie (a) und (c) wurden dann korrespondierende Punkte gesucht. Diese sind als Linien dargestellt. Wie man sieht sind sehr viele fehlerhafte Verbindungen vorhanden. Um diese zu entfernen wurde nun mittels *RANSAC* die Homographie und deren *Inlier* berechnet. Das



(a)



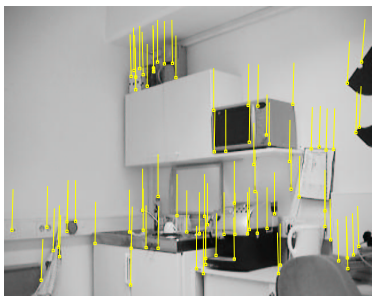
(b)



(c)

(d)

(e)



(f)

Abbildung 2.3: Homographie mit RANSAC:(a)-(c) Korrespondierende Punkte vor Ransac; (d)–(f) Korrespondierende Punkte nach Ransac

Ergebnis ist in (d)-(f) dargestellt.
Bei (d),(e) wurden aus 139 Punktpaaren 88, bei (d),(f) aus 133 Punktpaaren 81 als *Inlier* erkannt. In den Bildern ist nun die Bewegungsrichtung sehr deutlich zu erkennen.

**Simulation** Zum Testen des linearen Algorithmus wurden 100 Punkte ($p_0$) zufällig auf einem Bild ( 640 x 480 ) verteilt. Diese wurden dann mittels $p_i = KR_iK^{-1}p_0$ `für` $i = 1, 2$ in zwei andere Bilder umgerechnet, wobei beim ersten Bild eine Rotation um die X-Achse und beim zweiten Bild eine Rotation um die Y-Achse vorgenommen wurde.
Für K wurde

$$K = \begin{bmatrix} 1000 & 0 & 320 \\ 0 & 1000 & 240 \\ 0 & 0 & 1 \end{bmatrix} \tag{2.14}$$

gewählt.
Danach wurden diese $p_i$ durch Gauss'sches Rauschen gestört und mittels dem in Abschnitt 2.1.1 vorgestellten Algorithmus die inneren Kamera-Parameter berechnet.

Da der Algorithmus im wesentlichen von der Berechnung der Homographie zwischen zwei Bildern abhängt und um eine bessere Aussagekraft zu erhalten, wurden die Berechnungen mehrfach (20 mal) durchgeführt und jeweils die schlechtesten Ergebnisse gezeigt. Auf der Y-Achse wird jeweils das Verhältnis zwischen dem korrekten Wert und dem ermittelten Wert angezeigt.

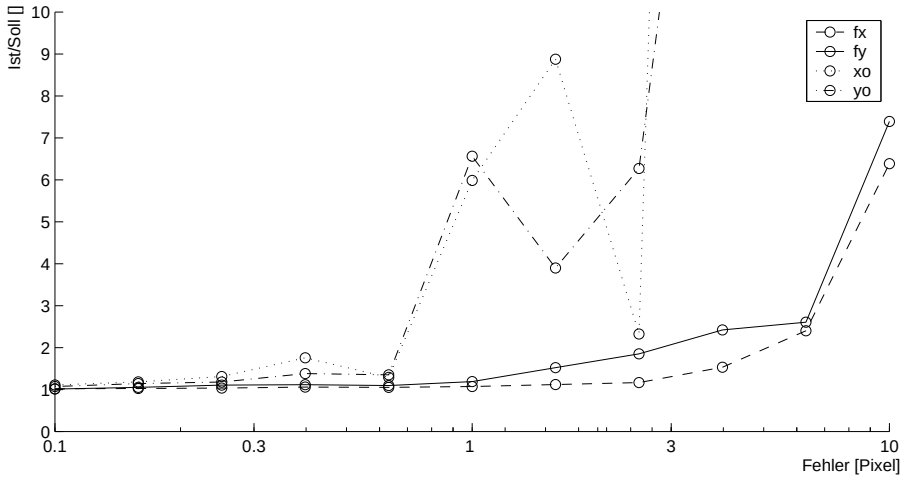



Abbildung 2.4: Innere Kamera-Parameter bei veränderlichem Rauschen

**Rauschen** Abbildung 2.4 zeigt die Ergebnisse für die inneren Kamera-Parameter ( $f_x$,$f_y$,$x_0$ und $y_0$ ) in Abhängigkeit vom Rauschen. Der Rotationswinkel betrug $\alpha = 5^o$. Es ist deutlich zu erkennen, dass bei steigendem Rauschen auch der Fehler für die einzelnen Parameter steigt. Ab einem Fehler von einem Pixel wird der ermittelte Wert für $x_0$ und $y_0$ jedoch unbrauchbar.

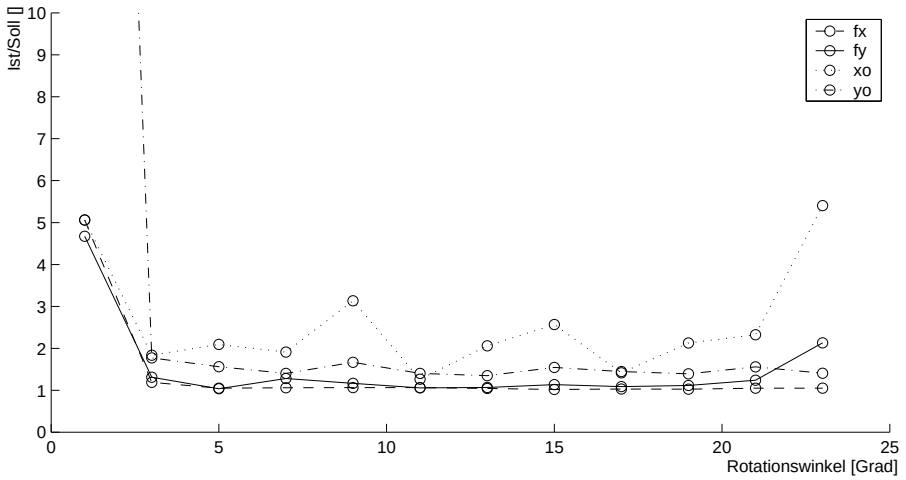



Abbildung 2.5: Innere Kamera-Parameter bei verändertem Winkel $\alpha$

**Rotationswinkel** Abbildung 2.5 zeigt die Auswirkungen von unterschiedlichen Rotationswinkel zwischen den Aufnahmen. Gut zu erkennen sind hier die schlechten Ergebnisse bei zu kleinem und zu großem $\alpha$. Letzteres ist hier aber auf zu wenige gefundene korrespondierende Punkte zurückzuführen.

**Falscher Rotationswinkel** Abbildung 2.6 zeigt die Auswirkungen von falschem Rotationswinkel zwischen den Aufnahmen. Hierbei wurden die Punkte auf den drei Bildern mit einem Winkel von $\alpha = 5^o$ berechnet, dem Kalibrier-Algorithmus jedoch ein davon abweichender Wert mitgeteilt. Es ist zu erkennen, dass bereits eine Abweichung um wenige Prozent ein schlechtes Ergebnis liefert.

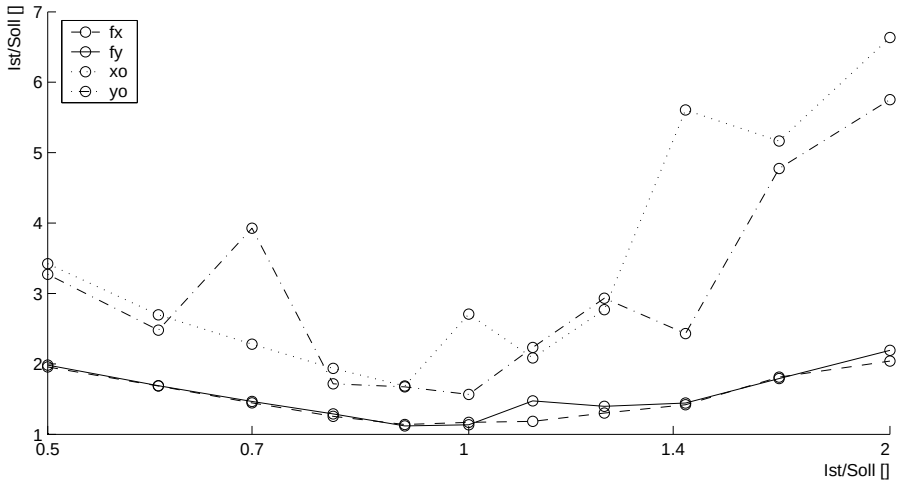
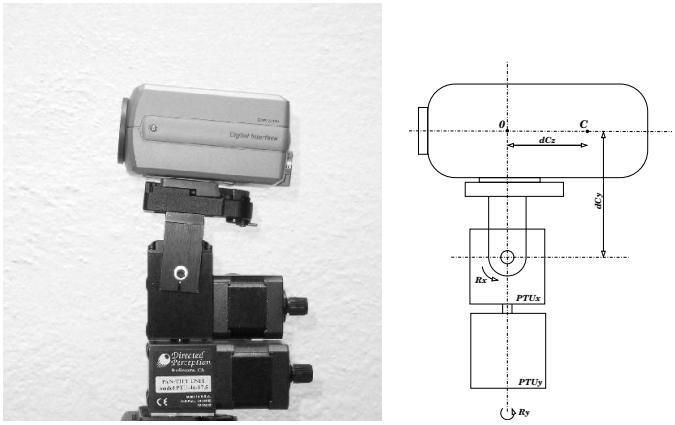



Abbildung 2.6: Innere Kamera-Parameter bei falschem Winkel $\alpha$

**Falsches Rotationszentrum** Die Vereinfachung von Gleichung 1.1 zu Gleichung 2.4 gilt nur, wenn das Rotationszentrum und das Kamerazentrum übereinstimmen. In Abbildung 2.7(a) ist die Kamera mit der PTU fotografiert. Daneben in Abbildung 2.7(b) sieht man die schematische Darstellung. In dieser sind die beiden Rotationsachsen um die die Kamera gedreht wird, sowie das Kamerazentrum eingezeichnet. Bei unserem Aufbau wurde ein $CdY = 9.2cm$ gemessen. Das Messen von $CdZ$ ist nicht möglich, da sich dieses innerhalb der Kamera befindet und auch vom eingestellten Zoom der Kamera abhängt.
Die Simulation wurde wie bei den vorherigen Fehlerquellen durchgeführt, jedoch mit der Ausnahme, dass hier die zur Kalibration verwendeten Punkte einen Abstand von zwei Meter zum Rotationszentrum haben und die Abbildung eines Punktes ins Bild mittels

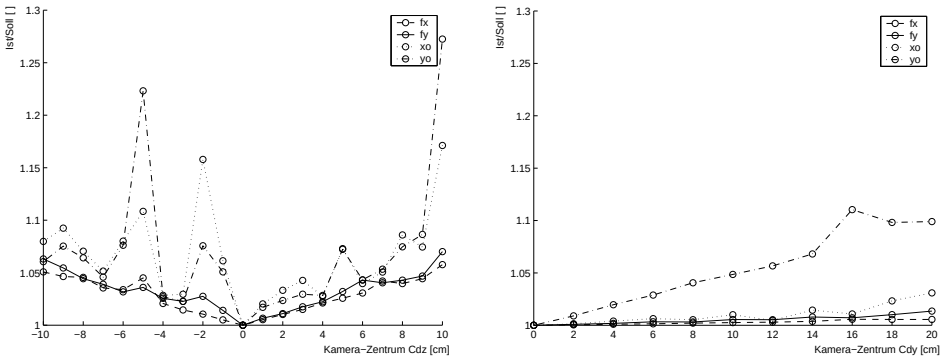
$$\lambda p = K\ R\ (X + CdY) - CdY + CdZ \tag{2.15}$$

berechnet wurde. In Abbildung 2.8 sind die Auswirkungen der beiden Größen $CdZ$ und $CdY$ auf die inneren Kamera-Parameter dargestellt.
Im Vergleich zu den anderen Fehlerquellen ist diese eine systematische. In Abbildung 2.8(a) ist zu sehen, dass sich eine Veränderung des Parameters $CdZ$ auf alle inneren Kamera-Parameter in ähnlicher Weise auswirkt. Ausgehend vom Null-Punkt steigen die Fehler annähernd linear an. In Abbildung 2.8(b) jedoch sieht man, dass die Änderung von $CdY$ nur eine Auswirkung auf $y_0$ hat und die anderen Parameter ($x_0$,$f_x$ und $f_y$) nicht beeinflusst.



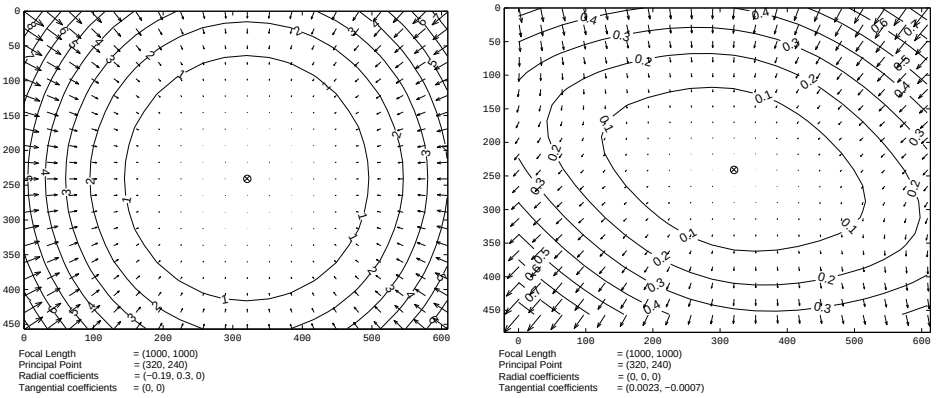

(a) Photo (b) Schematische Darstellung

Abbildung 2.7: Kamerazentrum, Rotationszentrum




(a) Variation von $CdZ$ (b) Variation von $CdY$

Abbildung 2.8: Innere Kamera-Parameter bei verändertem Kamerazentrum




(a) Radiale (b) Tangentiale

Abbildung 2.9: Linsenverzeichnungen nach [Bou03]

**Linsenverzeichnung**

Linsenverzeichnungen sind definiert durch:

$$p_{dist} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} r_r + r_t \tag{2.16}$$

Beschrieben werden die radialen und tangentialen Verzeichnungen durch:

$$r_r = (1 + k_1 r^2 + k_2 r^4 + k_5 r^6) \tag{2.17}$$

$$r_t = \begin{bmatrix} 2k_3 p_x p_y + k_4(r^2 + 2{p_x}^2) \\ 2k_4 p_x p_y + k_3(r^2 + 2{p_y}^2) \end{bmatrix} \tag{2.18}$$

In Gleichung 2.16 bis 2.18 entspricht $r = \sqrt{{p_x}^2 + {p_y}^2}$ dem Radius des Punktes vom Zentrum des Bildes ($x_0$,$y_0$) und $k_1$ bis $k_5$ den Parametern der Verzeichnung.

Diese Verzeichnungen werden verursacht durch:

- falscher Schliff der Linsen
- falsche Ausrichtung zwischen CCD-Chip und der Linse

In Abbildung 2.9 sind die Auswirkungen dieser Verzeichnungen dargestellt.
Da diese Verzeichnungen nicht linear sind, ist eine eindeutige mathematische Lösung in der Literatur nicht zu finden. Um dennoch mit diesen arbeiten zu können, wird ein nichtlineares Optimierungsverfahren verwendet.

### Optimierung

Ziel dieses Verfahrens ist es die Abbildung eines Weltpunktes $X_i$ in einen Bildpunkt $p_{ij}$ unter Berücksichtigung der inneren Kamera-Parametern ($f_x$,$f_y$,$x_0$ und $y_0$) und radialen sowie tangentialen Verzeichnungen ($k_1$ bis $k_5$) zu optimieren.
Wichtig ist hierbei die richtige Reihenfolge in der Abbildung:

1. Weltpunkt in Kamera Koordinatensystem transformieren: $p_c = [R|t] \cdot X$.
2. Perspektivische Abbildung: $\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} \frac{p_{c_x}}{p_{c_z}} \\ \frac{p_{c_y}}{p_{c_z}} \end{pmatrix}$
3. Radiale und tangentiale Verzeichnungen: $p_{dist} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} r_x + r_t$.
4. Abbildung ins Bild: $p = K \cdot p_{dist}$.

Das verwendete Optimierungsverfahren und die verwendete Fehlerfunktion ist im Detail in Anhang B.2 zu finden.

### Experimente

Zum Testen der Kalibrierung wurde folgendes Experiment gemacht:

Der hier implementierte Kalibrier-Algorithmus wurde getestet. Dazu wurde die Kamera, welche auf der PTU montiert war in eine Ecke des Raumes gerichtet und die Kalibrierung gestartet. Diese machte drei Bilder aus verschiedenen Blickrichtungen (Bewegung durch die PTU). Danach wurden korrespondierende Punkte gesucht (siehe Abbildung 2.3 ), daraus die inneren Kamera-Parameter berechnet und diese dann mit Hilfe der Linsenverzeichnungen optimiert.

Die Ergebnisse:

$$K = \begin{pmatrix} 783.62 & 0 & 318.05 \\ 0 & 791.90 & 263.58 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.19}$$
$$kc = [k_1, k_2, \ldots, k_5] = [-0.2589, 0.4614, 0.0009, -0.0047, 0]$$

Zum Vergleich wurde dieselbe Kamera mit dem in [Bou03] vorgestellten Kalibrier-Algorithmus kalibriert. Die Ergebnisse hieraus:

$$K = \begin{pmatrix} 793.80 & 0 & 347.23 \\ 0 & 789.22 & 253.07 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.20}$$
$$kc = [k_1, k_2, \ldots, k_5] = [-0.2398, 0.3788, 0.0014, 0.0019, 0]$$

Um die beiden Ergebnisse aus zwei verschiedenen Kalibrier-Algorithmen miteinander zu Vergleichen wurde folgende Vorgangsweise gewählt:

1. Für jeden möglichen Punkt im Bild wurde nach Abschnitt 2.1.2 ein Vektor mit den Ergebnissen aus 2.20 berechnet.

2. Diese wurden dann mit den Ergebnissen aus 2.19 ins Bild zurück projiziert.

3. Zwischen den Ausgangspunkten und den zurückprojizierten Punkten wird die Distanz ermittelt. Diese entspricht dem Fehler, der durch die anderen Kamera-Parameter entsteht.

Der durchschnittliche Fehler, der dabei entsteht beträgt 31.9 Pixel mit einer Standardabweichung von 2.5 Pixel. Da die zweite Kamera sich jedoch an einer anderen Position befinden könnte, wurde nun vor der Projektion die Position der Kamera so optimiert, dass sich der Fehler minimiert. Der durchschnittliche Fehler beträgt dann 2.8 Pixel mit einer Standardabweichung von 1.6 Pixel.

### 2.1.2 Panorama

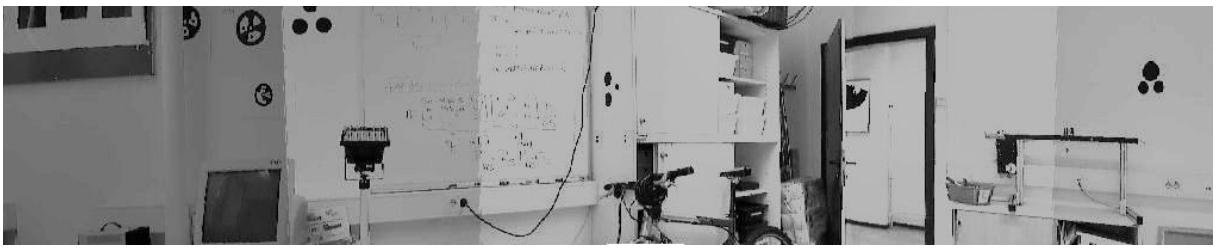


Abbildung 2.10: Panorama

Nachdem die Kamera kalibriert wurde, kann die Bildaufnahme, wie in Abbildung 2.1 skizziert, durchgeführt werden. In Abbildung 2.10 ist zur Veranschaulichung ein Panorama zu sehen, welches aus sechs nebeneinander aufgenommenen Bilder erstellt wurde. Es erfasst einen Blickwinkel von ca. $180^o$.

Die Kalibrierung in Kapitel 2.1.1 wurde mit relativ kleinen Winkeln durchgeführt um das Finden von gleichen Punkten so einfach wie möglich zu halten. Durch Vergleich von benachbarten Bildern merkte man aber, daß hier ein Fehler entstand. Deshalb wurde die Prozedur der Optimierung in Abschnitt 2.1.1 mit Punktkorrespondenzen von benachbarten Bildern erweitert und erneut durchgeführt.

Das Finden von Punktkorrespondenzen ist in diesem Fall (mit einer vorhandenen Kalibrierung) nicht sonderlich schwer. Da man den Rotationswinkel zwischen den

Aufnahmen kennt, kann zwischen zwei benachbarten Bilder mittels Gleichung 2.5 eine Homographie errechnet werden, mit der die Punkte von einem ins andere Bild umgerechnet werden können.

**Berechnung der Vektoren**

Zur weiteren Verarbeitung ist es notwendig, die Punktkoordinaten in den verschiedenen Bildern in ein einheitliches Format zu bringen. Bei der Aufnahme der Bilder, wird für jedes einzelne ein Rotationsvektor $Rv_i = \begin{pmatrix} 0 \\ i \cdot A_\alpha \\ 0 \end{pmatrix}$ gespeichert. Dieser gibt die relative Orientierung der einzelnen Bilder untereinander wieder.

Ein Punkt $p_{ij}$ wird nun folgendermaßen in einen Vektor umgerechnet:

1. Inverse der Linsenverzeichnungen
2. Inverse Abbildung vom Bild $K^{-1}p_{ij}$
3. Transformation vom Kamera-Koordinatensystem ins Aufnahme-Koordinatensystem

Da Punkt 1 mathematisch nicht lösbar ist, wurde eine Look-Up Tabelle erstellt, die zu jedem Bildpunkt $p_{dist_{ij}}$ mit Linsenverzeichnungen den dazugehörigen Bildpunkt ohne Linsenverzeichnungen $p_{ij}$ liefert.
Die Berechnung eines Vektors erfolgt nun mit:

$$v_i = {R_j}^{-1} \cdot K^{-1} \cdot p_{ij} \tag{2.21}$$

### 2.1.3 Äußere Orientierung

Die geometrischen Zusammenhänge zwischen zwei Aufnahmen an zwei unterschiedlichen Positionen stellt Abbildung 2.11 dar. Diese werden nun erläutert, und eine mathematische Lösung zur Berechnung der äußeren Orientierung wird angegeben.

Alle folgenden Angaben beziehen sich auf Abbildung 2.11 . Zur Vereinfachung besteht hier eine Aufnahme aus nur 5 Bildern, von denen jeweils nur eine Bildebene komplett eingezeichnet ist. Dies ist in Aufnahme 1, mit Projektionszentrum $c1$, Bild 4 und in Aufnahme 2, mit Projektionszentrum $c2$, Bild 3.

Aus den Punkten $p_I$ und $p_{II}$ werden, wie in Kapitel 2.1.2 erläutert, die Vektoren $V_1$ und $V_2$ erzeugt.

Da wir nur die relative Orientierung suchen, können wir $c1 = [0, 0, 0]^T$ in den Ursprung verschieben ohne die Allgemeinheit zu verletzen. Die zwei Projektionszentren $c1$ und $c2$ bilden mit einem Punkt $W$ eine Ebene $\pi$. Der Normalvektor zu dieser Ebene kann durch die Vektoren $t$ und $v_1$ durch ein Kreuzprodukt gebildet werden:

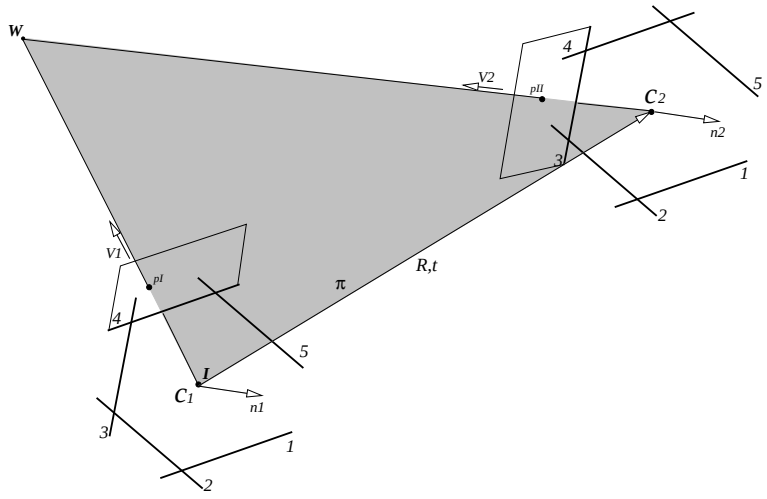



Abbildung 2.11: Äußere Orientierung

$$t \times v_1 = n_1 \tag{2.22}$$

Dieser Vektor $n_1$ wird im Kamerazentrum $c_2$ zu $n_2 = R(t \times v_1)$. Da nun aber $v_2$ und $n_2$ normal aufeinander stehen, muss gelten:

$$R(t \times v_1) \circ v_2 = 0 \tag{2.23}$$

Durch Vereinfachung von

$$E = R \cdot \text{skewsym}(t) \quad \text{und skewsym} \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix} = \begin{pmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{pmatrix} \tag{2.24}$$

sowie durch Umschreiben des inneren Produktes $\circ$ ergibt sich:

$$v_2{}^T E v_1 = 0 \tag{2.25}$$

In Gleichung 2.25 handelt sich bei $E$ um die *Essential* Matrix, welche die äußere Orientierung darstellt.

### Berechnen der *Essential* Matrix

Um die *Essential* Matrix zu berechnen verwenden wir einen linearen Algorithmus. Da $E$ nur bis auf einen Skalierungsfaktor bestimmt werden kann und aus 9 Unbekannten besteht, benötigt man mindestens 8 unabhängige Gleichungen. Diese werden

durch mindestens 8 Punktkorrespondenzen von Gleichung 2.25 gewonnen, die in ein lineares Gleichungssystem der Form

$$Ae = 0 \tag{2.26}$$

umgeschrieben werden. Wobei A eine $N \times 9$ Matrix mit den Werten von Gleichung 2.25 und $e$ ein $9 \times 1$ Vektor mit den Elementen von $E$ ist.

**Ermitteln der Parameter**

Da wir eigentlich an der Position und Ausrichtung der Kamera interessiert sind, müssen aus der *Essential* Matrix $E$ erst die Werte für die Translation $t$ und Rotation $R$ gewonnen werden. In [Har92] findet sich hierfür ein Algorithmus basierend auf *SVD*:

$$[u, d, v] = \text{svd}(E) \tag{2.27}$$

Die Lösungen für $R$ und $t$ aus Gleichung 2.23 ergeben sich zu:

$$\text{skewsym}(t) = vzv^T \tag{2.28}$$

$$R = uev^T \text{ oder } ue^Tv^T \tag{2.29}$$

wobei $z = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$ und $e = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$.

## 2.2 Korrespondenz mittels $F$

Ziel der Kalibrierung (innere und äußere Orientierung) ist die *Epipolar*-Geometrie. Diese stellt einen geometrischen Zusammenhang zwischen gleichen Punkten dar. Um in den Bildern damit arbeiten zu können, werden in Gleichung 2.25 die Vektoren in Punkte im Bild mittels Gleichung 2.21 umgerechnet. Dies ergibt

$$\begin{gathered} {v_2}^T E v_1 = 0 \\ ({R_{2i}}^{-1} K^{-1} p_{2i})^T E ({R_{1j}}^{-1} K^{-1} p_{1j}) = 0 \\ p_{2i}^T K^{-1T} {R_{2i}}^{-1T} E {R_{1j}}^{-1} K^{-1} p_{1j} = 0 \end{gathered}$$

Durch $F = K^{-1T} {R_{2i}}^{-1T} E {R_{1j}}^{-1} K^{-1}$ kommt man zu

$$p_{2i}^T F p_{1j} = 0 \tag{2.30}$$

Die in Gleichung 2.30 auftretende Matrix $F$ nennt man *Fundamental*-Matrix. Eigenschaften dieser sind:

- Punkte $p_{2i}$ können im Bild 1 nur auf der Linie $p_{2i}^T F$ liegen.

- Punkte $p_{1j}$ können im Bild 2 nur auf der Linie $Fp_{1j}$ liegen.

Beim Suchen eines korrespondierenden Punktes in einem zweiten Bild ist es mittels Gleichung 2.30 möglich den "Suchraum" von 2D (Vergleich aller Punkte im Bild) auf 1D (Suche nur auf der Linie) zu verringern.

# Kapitel 3

# Bildmerkmale

Um die Kalibrierung in Kapitel 2.1 durchführen zu können, benötigt man korrespondierende Punkte in mehreren Bildern. Dazu eignen sich künstliche (3.1) sowie natürliche (3.2) Bildmerkmale.

## 3.1 Kalibrier-Targets

Targets sind künstliche Punkte im Raum, welche Eigenschaften bieten, mit denen man sie leichter als reale Punkte vergleichen kann. Diese Eigenschaften sind:

- Punkte in den Targets müssen genau lokalisierbar sein.
- Targets müssen eine Wert besitzen um sie zuordnen zu können.

Da bei projektiven Abbildungen alle Figuren und Formen projektiv verzerrt werden, müssen die Punkte in den Targets projektiv invariant sein. Bekannte projektive Invariante sind das Doppelverhältnis zweier Geraden und die konvexe Hülle eines Objektes.

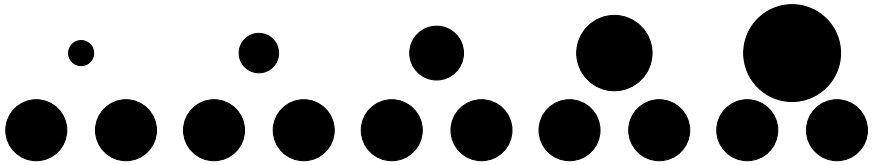
In dieser Arbeit wurde als Grundelement des Targets der Kreis gewählt, weil dieser einfach zu detektieren und seine Parameter einfach zu berechnen sind. Als projektive Invariante dient die konvexe Hülle zweier Kreise.

### 3.1.1 Beschreibung des Targets

Wie man in Abbildung 3.1 sehen kann bestehen die verwendeten Targets aus drei Kreisen. Die unteren zwei Kreise sind bei allen Targets identisch, die Größe des oberen Kreises stellt den kodierten Wert des Targets dar.

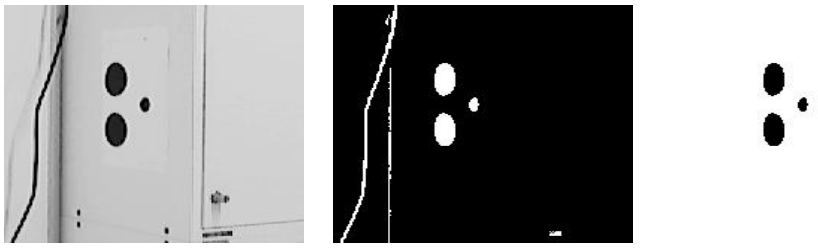
### 3.1.2 Detektion von Ellipsen

Die Detektion der Kreise, die sich bei einer projektiven Abbildung als Ellipse darstellen, funktioniert relativ einfach:



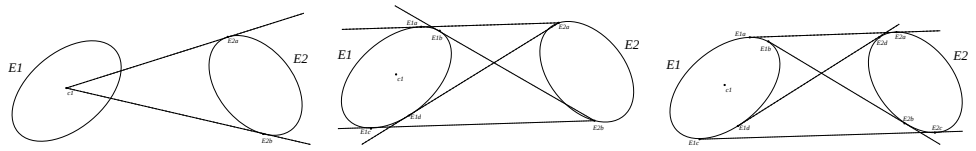
(a) (b) (c) (d) (e)

Abbildung 3.1: Verwendete Targets



(a) Bildausschnitt (b) Schwellwert,Label (c) Ellipsen

Abbildung 3.2: Detektion von Ellipsen




(a) Schritt 1 (b) Schritt 2 (c) Schritt 3.

Abbildung 3.3: Tangente an zwei Ellipsen

Das gesamte Bild wird mit einem globalen Schwellwert, der dem Median aller Pixel entspricht, binarisiert. Danach werden zusammenhängende Regionen mit *labelregion* gefunden (Abbildung 3.2(b) ).

Wie in [Mat00] erläutert, ist es mittels der Gausschen Parameter ($\mu = (\overline{x}, \overline{y})$ und $\sum = ((\sigma_x^2, \sigma_{xy}), (\sigma_y^2, \sigma_{xy}))$ möglich, die Parameter der Ellipsen zu bestimmen:

- Zentrum der Ellipse : $\mu$
- Haupt- und Nebenachse der Ellipse: erster und zweiter Eigenvektor von $\sum$
- Außen und Innenradius der Ellipse: $r_a = 2\sqrt{\sigma_{v1}^2}$ und $r_i = 2\sqrt{\sigma_{v2}^2}$ , wobei $\sigma_{v1}^2$ und $\sigma_{v2}^2$ der erste und zweite Eigenwert von $\sum$ sind.

Da durch die Binarisierung nicht nur Ellipsen als Regionen gefunden werden, wir aber nur an diesen interessiert sind, wurde ein Test verwendet, der nur Ellipsen detektiert. Die Fläche der Ellipse $A_{e_{pixel}}$ ist aus der Anzahl der Pixel in der Region bekannt. Man kann diese Fläche jedoch auch aus den Radien der Ellipse berechnen $A_{e_{radius}} = \pi r_i r_a = 4\sqrt{\sigma_{v1}^2 \sigma_{v2}^2} = 4\pi\sqrt{\sigma_x^2\sigma_y^2 - \sigma_{xy}^2}$. Als Entscheidungskriterium wird nun das Verhältnis dieser zwei Flächen verwendet:

$$|1 - \frac{A_{e_{pixel}}}{A_{e_{radius}}}| < 0.1 \tag{3.1}$$

Nur wenn Gleichung 3.1 zutrifft handelt es sich wirklich um eine Ellipse (Abbildung 3.2(c) ).

### 3.1.3 Projektive Invariante von Kreisen

Die projektiv invarianten Punkte in den Targets sind die Schnittpunkte der Tangenten an zwei Kreisen.

Zur Berechnung der Tangenten an zwei Ellipsen wurde ein iteratives Verfahren verwendet, welches jedoch sehr schnell (nach ca. 5 Iterationen) konvergiert.

In Abbildung 3.3 ist der Vorgang graphisch zu betrachten. In Schritt 1 (Abbildung 3.3(a) ) werden vom Zentrum $c_1$ der Ellipse $E1$ zwei Tangenten zu Ellipse

$E2$ berechnet. Dies ergibt 2 Schnittpunkte $E2_a$ und $E2_b$. Ausgehend von diesen 2 Punkten werden nun in Schritt 2 (Abbildung 3.3(b) ) die Tangenten zu Ellipse $E1$ berechnet. Hier ergeben sich nun 4 Schnittpunkte $E1_a$,$E1_b$,$E1_c$ und $E1_d$. Beim erneuten Berechnen der Tangente dieser Punkte zu $E2$ ist zu beachten, dass die jeweils richtigen (extremen) Punkte gewählt werden. So ergeben sich nach einigen Iterationen (Abbildung 3.3(c) ) 4 Tangenten, und damit 8 projektiv invariante Punkte, an jeweils zwei Ellipsen.

Die Berechnung einer Tangente an eine Ellipse durch einen Punkt $p$ ist im Anhang A zu finden.

### 3.1.4 Detektion des Targets

Da ein Target aus drei Ellipsen besteht wird nun für jeweils drei benachbarte Ellipsen getestet, ob es sich um ein Target handelt. Dazu nutzen wir die Eigenschaften des Targets:

1. Alle Ellipsen sind vor der projektiven Abbildung Kreise.
2. Zwei der drei Kreise sind gleich groß und deren Größe ist bekannt.
3. Alle projektiv invarianten Punkte des Targets sind in einer Ebene.

Durch Eigenschaft 3 ergibt sich die Möglichkeit eine Projektion $H$ zwischen den gemessenen Punkten $p_{bild} = [x, y, 1]^T$ und den originalen Punkten $p_o = [x, y, 1]^T$ herzustellen:

$$p_{bild} = H * p_o \tag{3.2}$$

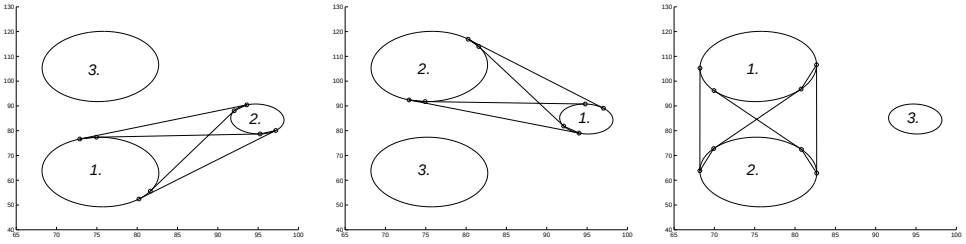
Hier ist $H$ eine 3 mal 3 Matrix, welche die projektive Abbildung zwischen zwei Ebenen (Bild und Original) beschreibt. Zum Berechnen dieser Matrix $H$ benötigt man 8 Punkt-Korrespondenzen, da man sie nur bis auf einen Skalierunsfaktor bestimmen kann.

Diese 8 Punkt-Korrespondenzen erhält man mit genau zwei Ellipsen. Durch Lösen des linearen Gleichungssystems 3.2 erhält man die projektive Abbildung des Targets.

Für $p_o$ werden die 8 Punkte verwendet, die durch die Tangenten zwischen den beiden gleich großen Kreisen enstehen.

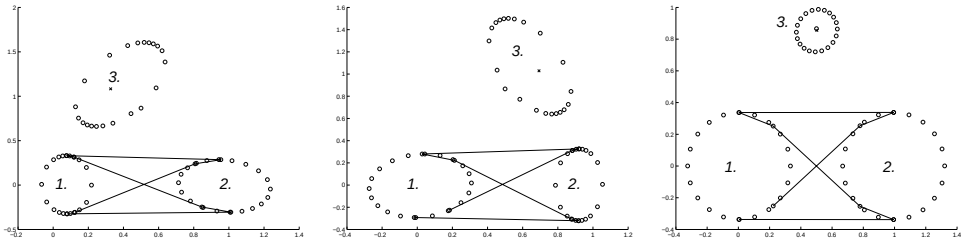
In Abbildung 3.4 sind jeweils die selektierten 8 Punkte zweier Ellipsen, und darunter in Abbildung 3.5 die Transformation der "Bildebene" in die "Originalebene" dargestellt. Wie deutlich zu erkennen ist, sind nur in Abbildung 3.5(c) die Ellipsen zu Kreisen geworden, d.h die Ellipsen 1 und 2 in Abbildung 3.4(c) sind jene, die dieselbe Größe haben. Damit entspricht der Wert des Targets der Größe des Kreises 3 in Abbildung 3.5(c) .

Als Entscheidungskriterium dient die "Kreisähnlichkeit". Dabei wird der Radius und die Abweichung davon für alle Kreise ermittelt. Ist die Abweichung unter einem




(a) (b) (c)

Abbildung 3.4: Auswahl von zwei Kreisen: Bildebene




(a) (b) (c)

Abbildung 3.5: Transformation in Originalebene

gewissen Wert ($\sigma_r < 0.2$ und $\sigma_{d\varphi} < 0.3$) bzw. ist die Position des dritten Kreises korrekt, so handelt es sich um ein gültiges Target.

Zum Kalibrieren werden von den Targets die in Abbildung 3.6 markierten Punkte verwendet. Diese sind die Außentangenten an allen drei Kreisen und sind somit projektiv invariant. Zusätzlich gilt, dass zwischen dem Punkt 1 und Punkt 2 in allen Targets der Abstand gleich groß ist. Dieses Maß wird später verwendet um die richtige Skalierung der Szene zu berechnen.

### 3.1.5 Experiment

Um zu Untersuchen wie gut sich die einzelnen Targets in Abbildung 3.1 unterscheiden lassen, wurden von jedem Target mit einer Kamera (jene aus Abschnitt 4.1) 20 Bilder aus verschiedenen Blickrichtungen und Entfernungen erstellt. In allen Bildern wurden dann die Targets lokalisiert.

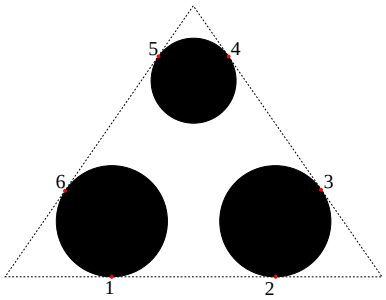



Abbildung 3.6: Kalibrier-Punkte der Targets

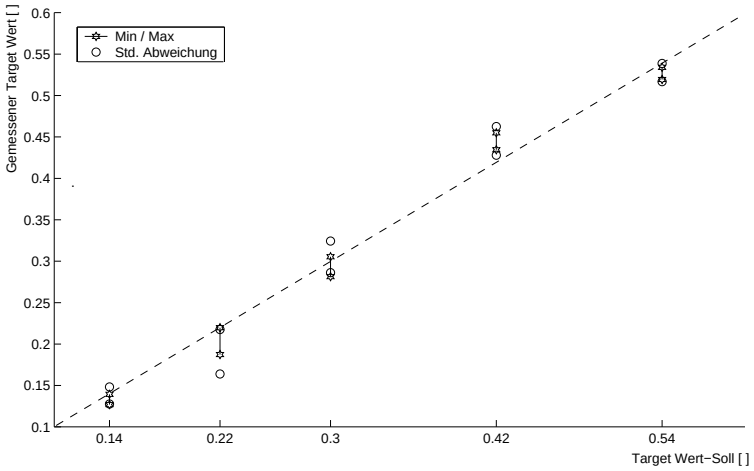



Abbildung 3.7: Ermittelte Werte bei gemessenen Targets.

In Abbildung 3.7 sind die Ergebnisse dieses Versuches dargestellt. Auf der Y-Achse sind die kodierten Werte der Targets, welche aus den Aufnahmen rekonstruiert wurden, dargestellt. Die einzelnen Targets überlappen sich auf der Y-Achse nicht, d.h. man kann sie richtig zuordnen, und damit als korrespondierende Punkte benützen. Noch zu erwähnen ist, dass bei diesen 100 gemachten Aufnahmen nur richtige Targets als solche erkannt wurden. Fehler wurden nur dann beobachtet, wenn das Target zu weit entfernt war, bzw. ein Kreis nicht als solcher detektiert wurde.

## 3.2 Natürliche Bildmerkmale

Betrachtet man die Matrix $M$

$$M = \left( \begin{pmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{pmatrix} \right) \tag{3.3}$$

in der $I(x, y)$ die Intensitätswerte des Graustufenbildes darstellen. Diese Matrix $M$ nennt man auch Momentenmatrix. Wenn nun die zwei Eigenwerte dieser Matrix an einem Punkt im Bild groß sind, bedeutet dies eine große Änderung des Intensitätswertes bei einer kleinen Änderung der Position. Dies deutet auf eine Ecke im Bild hin. Die Eckenstärke-Funktion (*response function*) ist definiert durch

$$R = \det(M) - k(\mathrm{tr}(M))^2 \tag{3.4}$$

wobei $k = 0.04$ gesetzt wird [HS88].
Um Ecken, die durch Rauschen im Bild entstehen, zu vermeiden werden die Ableitungen in Gleichung 3.3 zusätzlich mit einem Gauss-Kern geglättet.
Bei der Anwendung dieses Ecken-Detektors (*Plessey-Detector*) enstehen rund um eine wirkliche Ecke sehr viele Maxima, daher wird bei der Implementierung darauf geachtet, dass benachbarte Ecken einen Mindestabstand von ca. 10 Pixel aufweisen. Dazu wird für alle Punkte im Bild die Eckenstärke-Funktion berechnet. Dann wird jener Punkt ausgewählt, dessen Ergebnis am größten war. Dieser Punkt wird als Ecke berichtet. Im Umkreis von 10 Pixel um diesen Punkt werden nun alle Ergebnisse der Eckenstärke-Funktion auf null gesetzt. Dies wird solange wiederholt bis alle Ecken im Bild lokalisiert sind.
Beispiele von in Bildern detektierten Ecken sind in Abbildung 4.6 zu sehen.

## 3.3 Matching

Unter *Matching* bezeichnet man das Finden von gleichen Punkten in zwei verschiedenen Bildern derselben Szene. Bei künstlichen Merkmalen ist dies durch die kodierte Information in den Merkmalen einfach. Problematischer wird es jedoch bei natürlichen Merkmalen. In der Literatur hat sich *SSD* bewährt. Dieses Verfahren ist jedoch nur begrenzt einsetzbar und wurde deshalb erweitert.

### 3.3.1 SSD

*SSD* (*Sum of Squared Differences*) ist ein einfaches und schnelles Verfahren zum Vergleichen von Bildbereichen.

$$SSD = \sum_{w \in W} [IL(w) - IR(w)]^2 g(w) \tag{3.5}$$

Der Bildbereich $W$ wird in den Bildern $IL$ und $IR$ zusätzlich mit einer Gauss-Funktion $g(w)$ gewichtet.
Da die Kamera zwischen den Aufnahmen eine große Bewegung durchmacht, reicht dieser Vergleichswert jedoch nicht aus um Punkte einander eindeutig zuzuordnen. Deshalb wurde eine Verfeinerung dieses Verfahrens entwickelt.

### 3.3.2 Rotation und Skalierung

Aus zwei zu vergleichenden Punkten $p_i$ und $p_j$ wird der dazugehörige Welt-Punkt $X$ berechnet (siehe Kapitel 3.4). Mittels diesem wird dann der Skalierungsfaktor

$$scale = \frac{\text{dist}(X, c_i)}{\text{dist}(X, c_j)} \tag{3.6}$$

bestimmt. Wobei $\text{dist}(x, y)$ die euklidische Distance zwischen den Punkten $x$ und $y$ ist. Und $c_i$ und $c_j$ die beiden Kamerazentren sind.
Die Rotation wird anders als in der Literatur nicht durch Rektifizierung des gesamten Bildes eliminiert, sondern für jeden Punktvergleich gezielt berechnet. Die Winkeldifferenz berechnet sich aus den Epipolar-Linien $l_1 = p_2F$ und $l_2 = Fp_1$ mit

$$\alpha = \tan(k_{l_1}) - \tan(k_{l_2}) \tag{3.7}$$

wobei $k$ die Steigung der Epipolar-Linien darstellt.

Zum Vergleich der beiden Bildbereiche wird nun für jedes Pixel ($x_1$,$y_1$) in Bild 1 das dazugehörige Pixel in Bild 2 mit
$[x_2, y_2]^T = \text{scale} \cdot \begin{bmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{bmatrix} [x, y]^T$ berechnet. Dadurch wird der Bildbereich gedreht und skaliert um eine bessere Übereinstimmung zu erhalten.

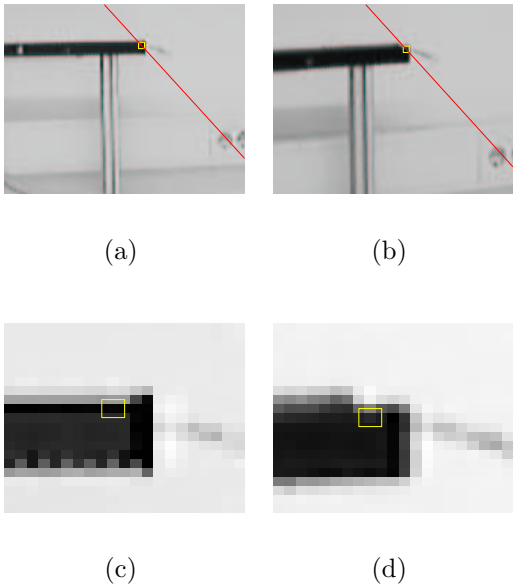
### 3.3.3 Intensitätsnormalisierung

Da die beiden zu vergleichenden Bildbereiche unter verschiedenen Winkeln betrachtet werden, ist die Helligkeit der Bilder unterschiedlich. Deshalb wurde eine Normalisierung durch

$$I_{norm}(x, y) = \frac{I(x, y) - \min(I)}{\max(I) - \min(I)} \tag{3.8}$$

durchgeführt.

In Abbildung 3.8 (a)-(b) sind zwei korrespondierende Bildbereiche dargestellt. Die eingezeichneten Linie stellen die Epipolar-Linien dar. Was hier sehr deutlich zu erkennen ist, ist die unterschiedliche Größe des Objektes in den zwei Bildern. Dies kommt durch den Distanzunterschied bei der Aufnahme. In Abbildung 3.8 (c)-(d) sind die beiden Bildbereiche durch Rotation, Skalierung und Intensitätsnormalisierung umgerechnet worden. Diese weisen nun dieselbe Größe, Ausrichtung und Helligkeit auf.



(a) (b)

(c) (d)

Abbildung 3.8: Matching von zwei Bildmerkmalen:(a),(b) Bild Features mit Epipolar-Linie; (c) Intensitätsnormalisiertes Merkmal von Bild a; (d) Rotiert,Skaliert u. Intensitätsnormalisiertes Merkmal von Bild b.

Nun wird das linke Bild in einer Umgebung des Punktes im zweiten Bild solange verschoben und der *SSD*-Wert berechnet bis sich ein Maximum ergibt. Dort wo das Maximum liegt, ist auch derselbe Punkt wie im linken Bild. In Abbildung 3.8 (d) ist die größte Übereinstimmung mit Abbildung 3.8 (c) mit einem gelben Rechteck gekennzeichnet.

### 3.3.4 Auswahl der Matches

Um korrespondierende Punkte besser auswählen zu können wird ein Unähnlichkeitsmaß eingeführt. Es sollen nur jene Punkpaare gewertet werden welche einen hohen Ähnlichkeitswert haben, aber mit den anderen möglichen Kandidaten so unähnlich wie möglich sind:

1. Wähle potentielle *Matches* aus: Das sind jene *Matches* bei denen für $p_i$ als bester *Match* $p_j$ auftritt, und für $p_j$ als bester *Match* $p_i$.

2. Berechne das Unähnlichkeitsmaß: UÄM $= 1 - \frac{m_{2nd}}{m_{ij}}$. Wobei $m_{ij}$ der Wert des *Matches* zwischen $p_i$ und $p_j$, und $m_{2nd}$ der Wert des nächstbesten *Matches* ist.

3. Sortiere die Liste aller potentiellen *Matches* nach ihrer Wertigkeit.

4. Sortiere die Liste der Unähnlichkeiten nach ihrer Wertigkeit.

5. Wähle jene *Matches* die in beiden Listen bei den besten $q$% dabei sind.

In der Implementierung wurde für $q$ ein Wert von 60 % gewählt. Dieser Wert stellt sicher, dass immer *Matches* ausgewählt werden.

## 3.4 3D-Koordinaten

In diesem Kapitel wird die Berechnung der 3D-Koordinaten von zwei korrespondierenden Punkten aus zwei Ansichten erläutert. Gesucht wird jener 3D Punkt $X$, welcher folgende Eigenschaften besitzt.

$$\lambda p_1 = P_1 X \quad \lambda p_2 = P_2 X \tag{3.9}$$

Und aus den Bildpunkten $p_1$ und $p_2$ und den Kamera-Matrizen $P_1$ sowie $P_2$ berechnet werden kann.

### 3.4.1 Lineare Triangulierung

Diese zwei Gleichungen können in ein lineares Gleichungsystem der Form $Ax = 0$ umgeschrieben werden, welches dann nach $X$ gelöst werden kann. Zuerst muss mittels Kreuzprodukt der Skalierungsfaktor $\lambda$ entfernt werden. $p \times (PX) = 0$ ausgeschrieben ergibt

$$\begin{aligned} x(p^{3T}X) - (p^{1T}X) &= 0 \\ y(p^{3T}X) - (p^{2T}X) &= 0 \\ x(p^{2T}X) - y(p^{1T}X) &= 0 \end{aligned} \tag{3.10}$$

Diese Gleichungen sind nun linear abhängig von $X$. Somit lässt sich $X$ durch Einsetzen beider Punkte $p_1$ und $p_2$ in die Gleichungen 3.10 lösen.

Die Lösung aus den Gleichungen 3.10 ist jedoch nur dann richtig wenn Gleichung 3.9 gilt. Da dies aber durch Fehler in der Lokalisierung der Punkte sowie der Kamera-Matrizen nie der Fall ist, werden die gemessenen Punkte zuerst so transformiert, dass Gleichung 3.9 gilt. Diese Transformation geschieht durch Finden der beiden Epipolar-Linien die den Punkten am nächsten sind und Gleichung 3.9 erfüllen. Die transformierten Punkte sind dann jene, die auf den Epipolar-Linien liegen. Näheres dazu in [HZ00] **TODO: [Kapitelangabe von multiview]** .

### 3.4.2 Optimierung

Da die lineare Triangulierung von der äußeren Orientierung abhängig ist und diese nur durch die Positionen weniger Targets ermittelt wurde, werden die 3D-Koordinaten optimiert.

Die hier verwendete Optimierung nennt sich in der Literatur *bundle-adjustment* (Bündel-Ausgleich). Hierbei werden nicht nur die Weltpunkte optimiert, sondern auch die äußere Orientierung. Ziel dieser Optimierung ist das Minimieren von

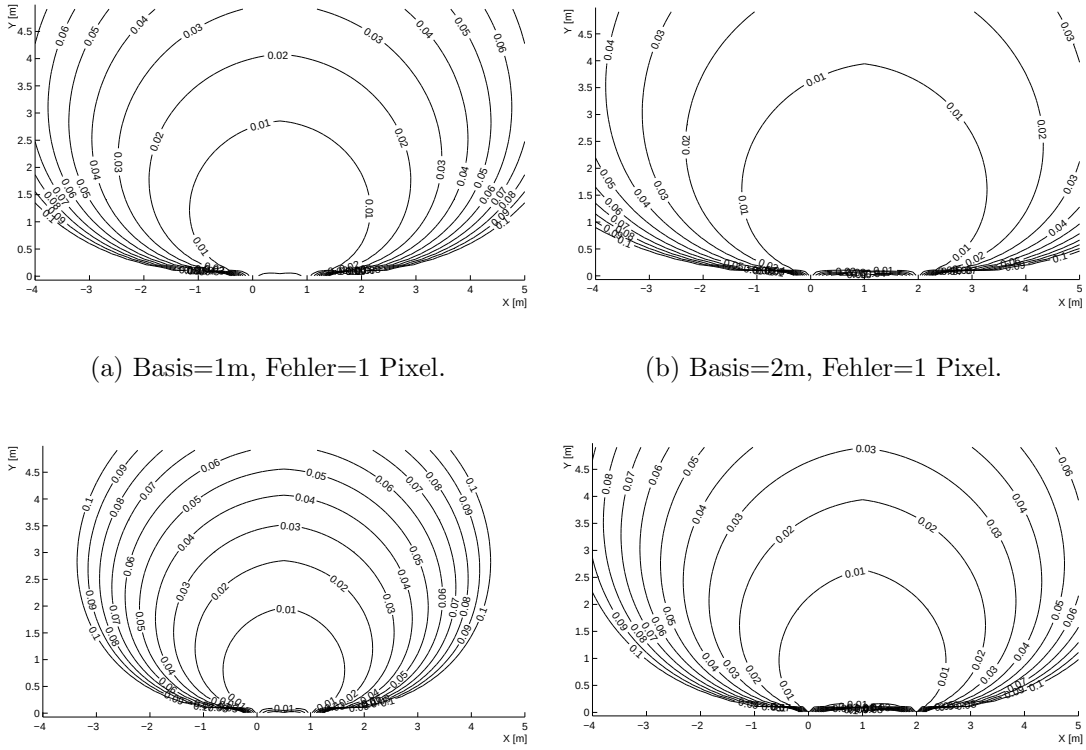
$$\min_{R,t,X_i} \sum_{X_i} (\text{dist}(K[R|t]X_i, p_{i2})^2 + \text{dist}(KX, p_{i1})^2) \tag{3.11}$$

wobei die genauere Berechnung in Anhang B.3 zu finden ist.

### 3.4.3 Simulation

Um zu analysieren von welchen Parametern die Genauigkeit einer Rekonstruktion durch diese Methode abhängt, wurde folgende Simulation durchgeführt:
Ausgehend von zwei Kamerastandorten mit einer Distanz (Basis) von einem und zwei Metern wurden die Punkte in der Umgebung mittels Triangulierung rekonstruiert. Als Fehlerquelle wurde in den Bildern (14 Bilder bei 40 % Überlapp) ein Fehler von einem und zwei Pixel verwendet.

In Abbildung 3.9 sind die Ergebnisse dieser Simulation dargestellt. Wie man deutlich sehen kann, ist der Fehler bei allen Versuchen in der Nähe der beiden Kamerazentren am kleinsten und wird bei zunehmender Entfernung größer. Besonders groß ist der Fehler in den zwei Bereichen links und rechts der beiden Kamerazentren. Dies hat damit zu tun, dass beim Blick der zwei Kameras in dieselbe Richtung keine *Stereo*-Information mehr vorliegt. Man kann von diesen Punkten nur die Richtung rekonstruieren, jedoch nicht die genaue Position bestimmen. Bei einer Verdoppelung der Basis (3.9(a)→3.9(b) und 3.9(c)→3.9(d)) kann man auch von einer Verdoppelung der Genauigkeit ausgehen. Ebenso wird bei einer Verdoppelung des Fehlers von einem auf zwei Pixel (3.9(a)→3.9(c) und 3.9(b)→3.9(d)) auch die Genauigkeit halbiert.




(a) Basis=1m, Fehler=1 Pixel.

(b) Basis=2m, Fehler=1 Pixel.

(c) Basis=1m, Fehler=2 Pixel.

(d) Basis=2m, Fehler=2 Pixel.

Abbildung 3.9: Fehler bei der Triangulierung.

# Kapitel 4

# Experimente

## 4.1 Aufbau

Abbildung 4.1: Aufbau

In Abbildung 4.1 wird der Aufbau des Versuches gezeigt. Hier handelt es sich um eine SONY-Fire-Wire Kamera, welche auf einer PTU montiert ist. Angesteuert wird die Kamera und die PTU von einem 2Ghz Pentium 4 Rechner mit Matlab 6.4.

Die Eigenschaften der Kamera sind in Tabelle 4.1 und die der PTU in Tabelle 4.2 festgehalten.

| | |
|---|---|
| Hersteller | Sony |
| Typ | DFW-VL 500 |
| Interface | IEEE -1394-1995 Fire-Wire |
| Auflösung | VGA (640x480), Non-Compressed |
| Linse | 12-fach Zoom |
| Gewicht | 335 g |

Tabelle 4.1: Eigenschaften der Kamera

| | |
|---|---|
| Hersteller | Directed Perception, Burlingame,CA |
| Name | Pan-Tilt-Unit PTU-C64-17.5 |
| Auflösung | $0.051428^o$ |
| Max. Belastung | 2.72 Kg |
| Max. Geschwindigkeit | $300^o$/sec |
| Winkelbereich Horizontal | 6180 Schritte = $317.825^o$ |
| Winkelbereich Vertikal | 1511 Schritte = $77.708^o$ |

Tabelle 4.2: Eigenschaften der PTU

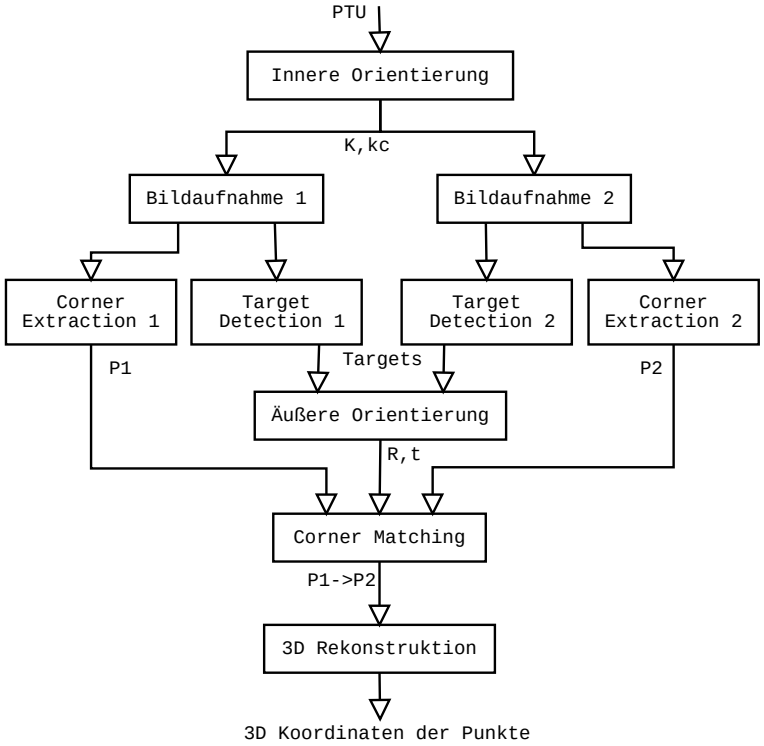



Abbildung 4.2: Ablaufdiagramm des Programmes

## 4.2 Ablauf

In Abbildung 4.2 ist das Ablaufdiagramm der Experimente dargestellt. Als Input dienen nur die Informationen der PTU. Mittels dieser wird nun wie in Kapitel 2.1 beschrieben die innere Orientierung ($K$ und $k_c$) ermittelt. Danach folgt die Bildaufnahme (Kapitel 2) welche für jeden Standort eine Bildfolge (Panorama) liefert. Aus diesen Aufnahmen werden danach die Targets extrahiert (Kapitel 3.1). Aus der Zuordnung der Targets zwischen den beiden Standorten wird dann die äußere Orientierung (Kapitel 2.1.3) $R$,$t$ berechnet. Durch diese Information lässt sich nun mittels der Epipolar-Linien (Kapitel 2.2) ein Vergleich (Kapitel 3.3) der extrahierten Punkte (3.2) durchführen. Und daraus die 3D-Koordinaten der Punkte berechnen (Kapitel 3.4).

## 4.3 Äußere Orientierung

Da das Messen der äußeren Orientierung nicht möglich ist (Kamerazentrum im inneren der Kamera kann nicht gemessen werden), wurde folgender Versuch durchgeführt:

Die Targets wurden auf die 4 Wände des Versuchs-Raumes (ein Target auf einen Kasten) geklebt. In Abbildung 4.3 sind die Aufnahmen der fünf für diesen Versuch verwendeten Targets (und ihre Lokalisation) dargestellt.

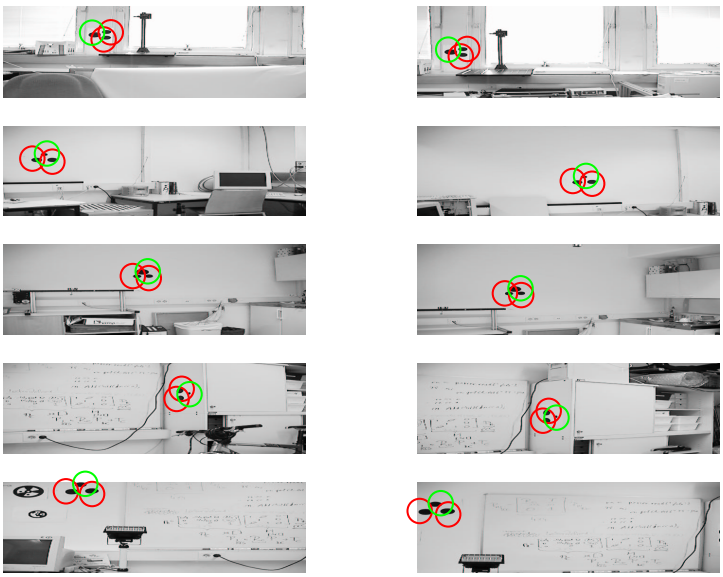


Abbildung 4.3: Aufnahmen der Targets

Danach wurden die Positionen und die Orientierung der Targets (mehrere Punkte in einem Target ergeben eine Ebene) rekonstruiert. Da die Wände des Raumes rechtwinklig aufeinander stehen, müssen dies auch die rekonstruierten Targets tun. Um einen Vergleich zu erhalten wurden die Koordinaten der Targets so transformiert, dass die Ebenen, in denen die Targets sich befinden mit der Ebene $x = 0$ und $y = 0$ übereinstimmen. Danach wurde der Fehler jedes einzelnen Targets ermittelt.

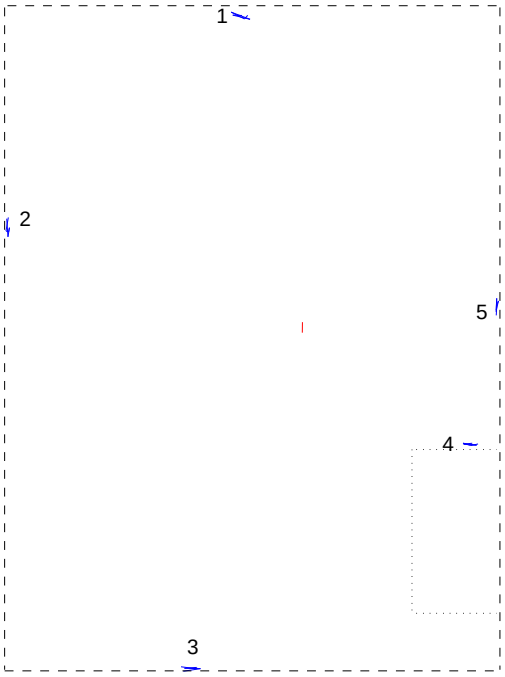



Abbildung 4.4: Rekonstruierte Targets

In Abbildung 4.4 ist die Visualisierung der rekonstruierten Positionen der Targets zu sehen. Zum besseren Verständnis wurden die Wände des Raumes (strichliert) und der Kasten (punktiert), auf denen sich die Targets befanden, eingezeichnet.
Die Abweichung der einzelnen Targets ergab

| Target | Winkel in Grad |
|---|---|
| 1 | 9.0540 |
| 2 | 0.1553 |
| 3 | 0.4029 |
| 4 | 9.3806 |
| 5 | 1.7072 |

was einem durchschnittlichen Fehler von $4.1400^o$ entspricht.

## 4.4 Natürliche Bildmerkmale

In der linken Aufnahme (13 Bilder) fanden sich 975 Merkmale und in der rechten 912 Merkmale. Daraus ergeben sich $975 * 912 = 88920$ Vergleiche, die durchgeführt werden müssten.
Ausgehend von der äußeren Orientierung wurden nun für jedes Merkmal in der linken Aufnahme mittels der Epipolar-Geometrie mögliche Merkmale in der rechten Aufnahme gesucht. Dazu wurde die Epipolar-Linie im rechten Bild berechnet und nur Markmale zugelassen, die sich in einem Bereich von 10 Pixel um diese befanden. Dadurch wurde die Anzahl der zu vergleichenden Merkmale auf 9213 (ein Faktor von 96.5) reduziert.
Aus den Ergebnissen des nach 3.3 durchgeführten Vergleiches wurden nun die besten (161) ausgewählt. Diese sind in Abbildung 4.6 dargestellt. Aus diesen wurden nun die 3d-Koordinaten nach 3.4 berechnet. Die Fehler der einzelnen Merkmale sind in Abbildung 4.5 dargestellt. Der Fehler der Bildmerkmale bewegt sich im Bereich von 0 bis 7 Pixel. Ganz rechts im Diagramm sind die Fehler der Targets dargestellt. Diese weisen einen maximalen Fehler von 0.8 Pixel auf. Die geringere Abweichung kommt von der Sub-Pixel genauen Lokalisation der Targets. In Abbildung 4.7 ist schließlich die 3D-Rekonstruktion der Bild-Merkmale zu sehen. Jene Merkmale, die einen größeren Fehler als $(5 \cdot median(allerFehler))$ aufweisen, wurden rot dargestellt und zum besseren Vergleich mit Nummern versehen. Die Fehler können in folgende Klassen unterteilt werden:

1. Fehler durch verdeckte Merkmale: 15 und 60
2. Fehler durch Kabel : 10, 63, 78 und 104
3. Fehler durch falsche Korrespondenz: 71, 91, 98 und 152
4. Andere Fehler: 57, 59,64 und 67

Die beiden Fehler (1) sind solche, in denen ein Bildmerkmal in einer Ansicht ganz zu sehen ist und in der zweiten durch einen Tisch oder eine Tür zur Hälfte verdeckt ist. Dies ergibt bei der Rekonstruktion natürlich einen Fehler.
Zu den Fehlern (2), die häufiger vorkommen, ist zu bemerken, dass es sich hier um Ecken handelt, die eigentlich keine sind: Ein Kabel mit einer dahinter liegenden Kante wird hier als Ecke detektiert. Diese ist jedoch keine und durch die unterschiedliche Tiefe des Kabels und der Kante ist die Position dieser virtuellen Ecke in jeder Ansicht verschieden. Ausnahmen bilden Kabel, die nahe an der Kante liegen (100).
Bei falschen Korrespondenzen (3) lieferte der *Matching*-Algorithmus falsche Bildmerkmale, die bei der anschließenden Optimierung aber nicht zu den anderen Punktpaaren passten. Solange es mehr richtige Paare gibt, werden diese optimiert, und die fehlerhaften durch ihren größeren Fehler erkannt.
Unter andere Fehler (4) sind solche gemeint, die bei Betrachten von Abbildung 4.5

keine Fehler erkennen lassen, jedoch als solche erkannt wurden.
Zusätzlich ist noch zu erwähnen dass es auch Paare gibt, die als korrekt erkannt wurden, aber falsch sind. Dies sind: 31, 49, 105, 125, 148, 151, 153, 154 und 156.

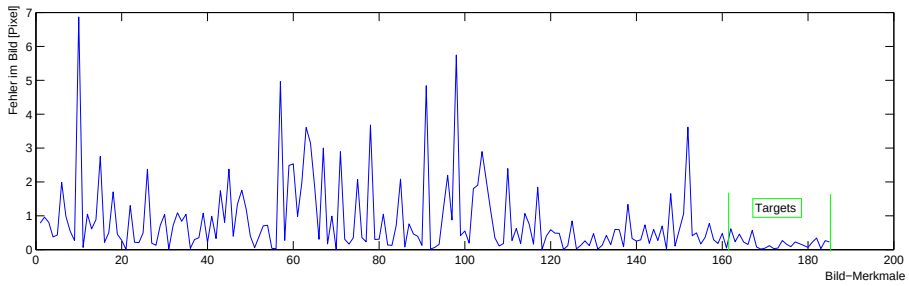



Abbildung 4.5: Fehler der Bildmerkmale

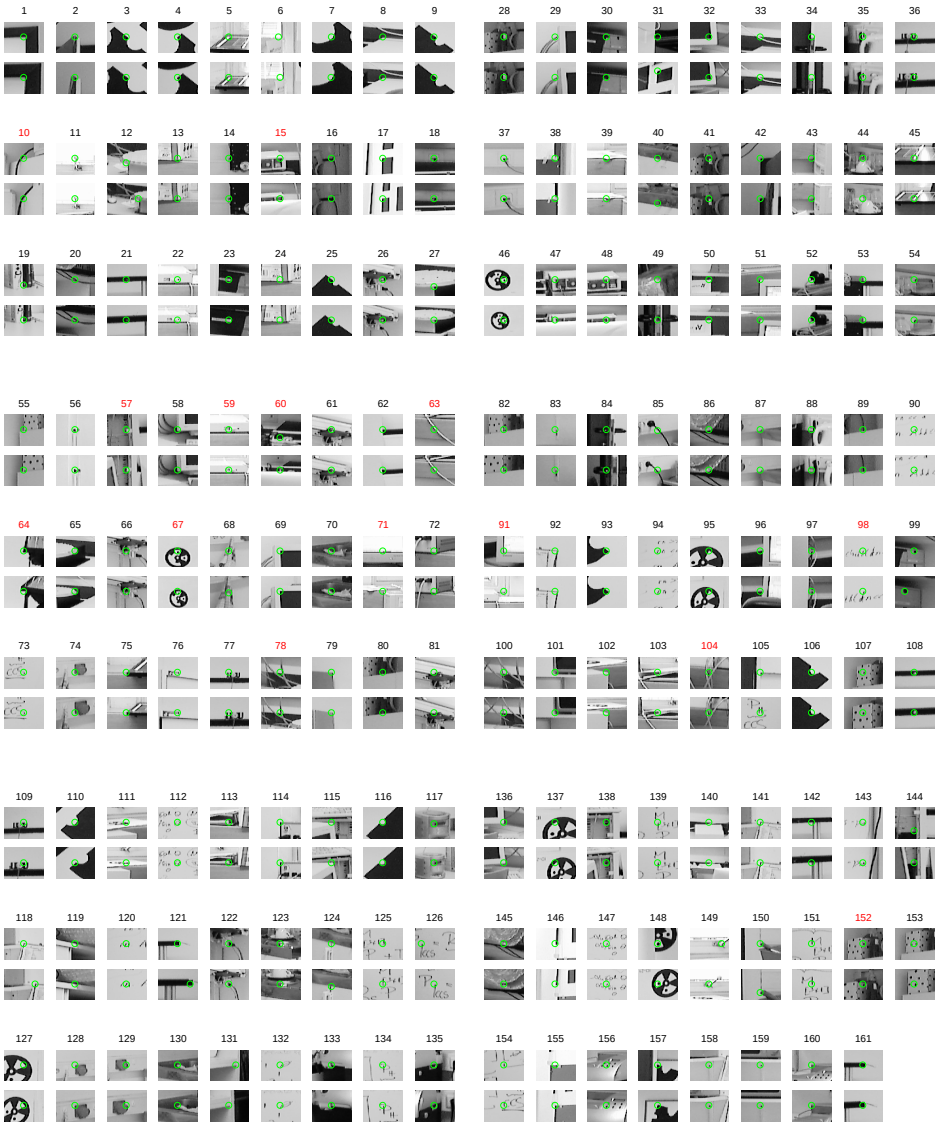


Abbildung 4.6: Korrespondierende Bildmerkmale

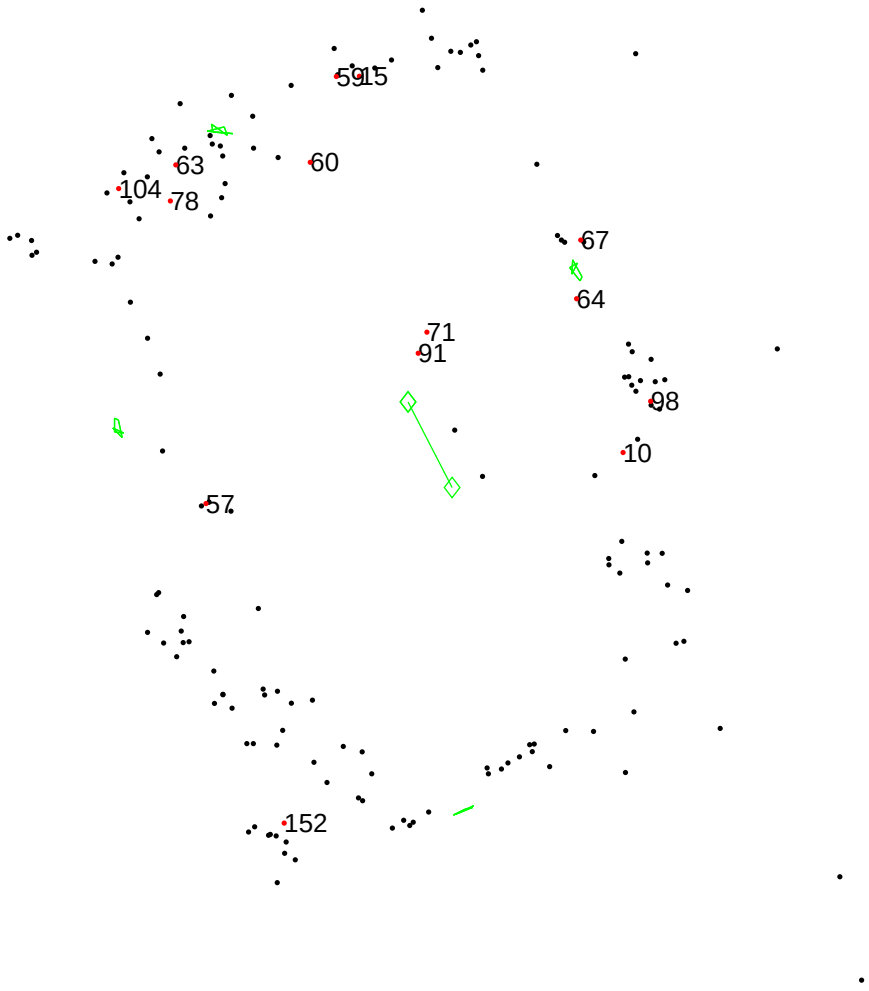



Abbildung 4.7: 3d-Rekonstruktion der Bildmerkmale

## 4.5 Genauigkeitsmessungen

Um die Genauigkeit diese Verfahrens zu ermitteln, wurde folgender Versuch durchgeführt.

An den Wänden wurden Marker angebracht, zwischen denen man die Distanz genau kennt. In Abbildung 4.8 sieht man zwei Marker zu je drei Quadraten. Da die Marker auf weißem Papier ausgedruckt wurden und dass auf der weißen Wand nicht gut zu erkennen war, wurden die Papier-Ränder strichliert nachgezeichnet. Zwischen den beiden äußeren Quadraten beträgt der Abstand genau einen Meter. Das mittlere dient nur zur Rekonstruktion der Ebene.

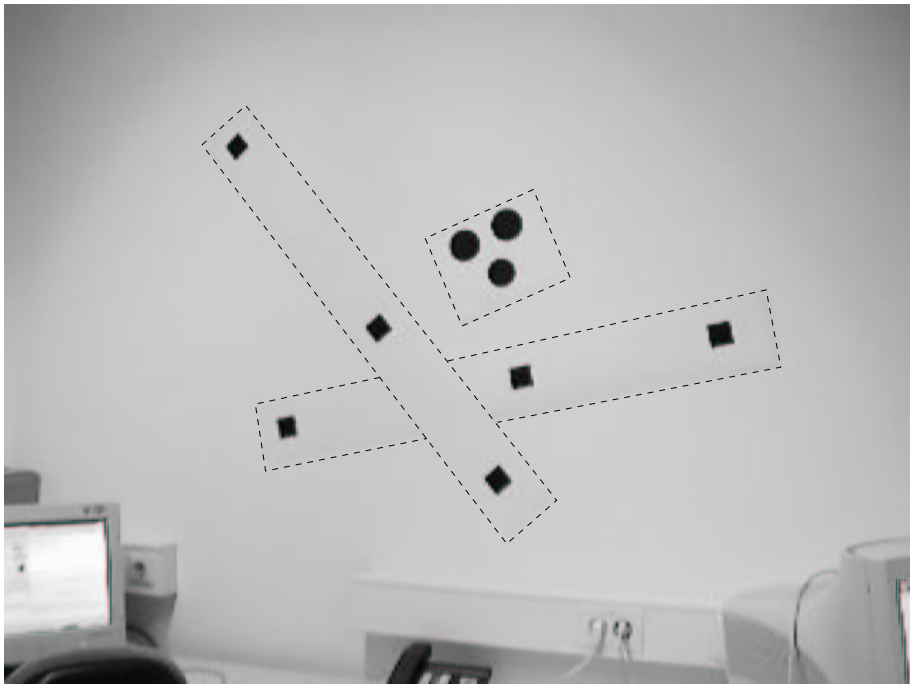


Abbildung 4.8: Vermessungs-Marken

Nach dem Anbringen der Marker wird die Rekonstruktion nach Abschnitt 4.2 erzeugt. Da nicht alle Quadrate als natürliche Markierung erkannt und dadurch rekonstruiert wurden, wurden alle auf den Wänden befindlichen Quadrate per Hand markiert. Aus der Rekonstruktion dieser (Abbildung 4.9 ) lässt sich nun die Genauigkeit ermitteln.

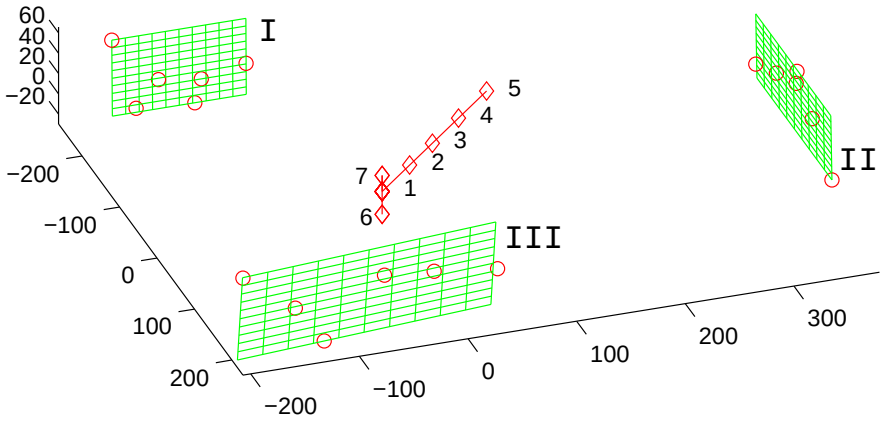



Abbildung 4.9: Rekonstruktion der Vermessungs-Marken (Einheiten in cm).

Dazu wurden folgende Werte berechnet:

1. Fehler der Abstände
2. Fehler zur Ebene (Je zwei Markierungen sind auf einer Wand befestigt. Diese sechs Quadrate liegen somit auf einer Ebene)

Um zu ermitteln, welchen Einfluß die Wahl der Basis (so nennt man die Distanz zwischen zwei Aufnahmepositionen die zur Rekonstruktion verwendet werden) auf die Genauigkeit der ermittelten Punkte hat, wurde dieser Versuch mehrmals durchgeführt.
Die Ergebnisse dieses Experimentes sind in Tabelle 4.3 bis Tabelle 4.5 dargestellt. Um eine bessere Aussagekraft zu erhalten, wurden die Ergebnisse der einzelnen Messungen zu ihren jeweiligen Flächen $(I, II, III)$ zusammengefasst.
In Tabelle 4.3 sind für alle sechs Messungen folgende Werte dargestellt:

- Aufnahme-Position: Beschreibt die zwei Positionen im Raum an denen die Kamera ihre Aufnahmen gemacht hat (siehe Abbildung 4.9 ).
- Basis: Die Distanz zwischen den zwei Aufnahme-Standorten.
- gefundene Punkte: Die Anzahl der von der Rekonstruktion erzeugten 3D-Punkte.
- Targets: die Anzahl der Targets, die zur Berechnung der äußeren Orientierung gefunden wurden.
- Matches: Anzahl der Punktvergleiche die durchgeführt wurden.

| Nr.: | Aufnahme-Position | Basis | gefundene Punkte | Targets | Matches |
|---|---|---|---|---|---|
| 1 | 1/2 | 54.7561 cm | 161 | 5 | 5968 |
| 2 | 1/3 | 99.6143 cm | 146 | 5 | 7291 |
| 3 | 1/4 | 151.6250 cm | 106 | 4 | 6930 |
| 4 | 1/5 | 207.2155 cm | 105 | 5 | 6765 |
| 5 | 1/6 | 22.4367 cm | 135 | 5 | 2068 |
| 6 | 6/7 | 38.3582 cm | 115 | 5 | 2545 |

Tabelle 4.3: Statistik zur Messung

Bei den Experimenten Nr.1 bis Nr.4 wurde die Kamera nur horizontal im Raum bewegt. Bei den Experimenten Nr.5 und Nr.6 war die Bewegung nur vertikal. Der Unterschied in der Art der Bewegung ist sehr gut an der Anzahl der durchgeführten Matches zu erkennen. Bei einer vertikalen Bewegung mussten nur 1/3 der Punk-Vergleiche wie bei einer horizontalen Bewegung durchgeführt werden.

| Nr.: | Durchschnittlicher Fehler in % | | | |
|---|---|---|---|---|
| | Fläche I | Fläche II | Fläche III | Alle Flächen |
| 1 | 5.6785 | 5.6468 | 3.7710 | 5.0321 |
| 2 | 5.1934 | 3.5408 | 4.4217 | 4.3853 |
| 3 | 1.9369 | 5.8575 | 3.6138 | 3.8028 |
| 4 | 5.5667 | 7.4065 | 12.9260 | 8.6330 |
| 5 | 10.5375 | 24.2520 | 3.7829 | 12.8575 |
| 6 | 4.0558 | 7.3708 | 2.4946 | 4.6404 |

Tabelle 4.4: Ergebnisse der Längenmessung (1m).

In Tabelle 4.4 sind die Ergebnisse der Längenmessungen dargestellt. Sehr gut zu erkennen ist die Tatsache, dass die Genauigkeit bei einer Zunahme der Basis steigt. Dies gilt jedoch nicht bei Experiment Nr.4 (Basis=207.2155cm). Hier war das Problem jenes, dass die Aufnahme der Quadrate von der Fläche $III$ im Bild nur mehr eine Größe von wenigen Pixel hatte. Dadurch war ein genaues *Matchen* zwischen den Aufnahmen nicht mehr so gut möglich.

| Nr.: | Durchschnittliche Distanz zur Ebene in cm | | | |
|---|---|---|---|---|
| | Fläche I | Fläche II | Fläche III | Alle Flächen |
| 1 | 0.1931 | 0.9951 | 6.2543 | 2.4808 |
| 2 | 1.0573 | 0.4456 | 1.6979 | 1.0669 |
| 3 | 1.2252 | 1.0079 | 2.2288 | 1.4873 |
| 4 | 0.2160 | 0.9728 | 4.3247 | 1.8478 |
| 5 | 0.7610 | 5.2756 | 0.9991 | 2.3452 |
| 6 | 1.4274 | 2.1704 | 0.4213 | 1.3397 |

Tabelle 4.5: Distanz zur Ebene.

In Tabelle 4.5 sind die Ergebnisse der Ebenenmessung dargestellt. Dabei wurde durch die sechs Punkte jeder Fläche eine Ebene gelegt und der Fehler der Punkte zu dieser Ebene berechnet. Deutlich zu erkennen ist hierbei die Tatsache, dass der Fehler bei der Fläche $III$ bei den Experimenten Nr.1 bis Nr.4 am größten ist. Das kommt daher, dass Punkte in der Fläche $III$ sehr nahe an der Verbindungsgeraden der beiden Kamerazentren liegen (siehe dazu Abschnitt 3.4.3). Die Fehlerverteilung bei den Experimenten Nr.5 und Nr.6 hängt nur von der Distanz der Kamerazentren von den zu rekonstruierenden Punkten ab. Die Fläche $II$ ist am weitesten davon entfernt.

# Kapitel 5

# Diskussion

Zu Beginn der Arbeit wurde eine kurze Literaturrecherche durchgeführt, welche auf die bereits vorhandenen Möglichkeiten einging. Es stellte sich jedoch heraus, dass diese Verfahren für die Problemstellung nicht einsetzbar sind.

Deshalb wurden die Eigenschaften die entstehen, wenn man eine Kamera rotiert und das Rotationszentrum mit dem optischen Zentrum der Kamera zusammenfällt, ausgenutzt. Dies entsteht bei der Montage der Kamera auf einer PTU. In Kapitel 2.1.1 wurde dann ein Kalibrier-Algorithmus vorgestellt, der genau diese Eigenschaften ausnutzt , um die inneren Kamera-Parameter zu berechnen. Da die Kamera aber auch Fehler durch Linsenverzeichnungen aufweist, wurden diese durch eine nichtlinear Optimierung gewonnen. Danach wurde zum Vergleich dieselbe Kamera mit einem am Institut vorhandenen Kalibrier-Algorithmus verglichen. Die Ergebnisse beider waren nahezu gleich gut.

Um eine 3D-Rekonstruktion durchführen zu können, muss zwischen zwei Aufnahmen die äußere Orientierung bekannt sein. Diese erhielten wir durch neu entwickelte Kalibrier-Targets. Diese "Kreismuster" wurden an den Wänden unseres "Testraumes" angebracht und danach in den aufgenommenen Bildern detektiert. Diese Kalibrier-Targets haben den Vorteil, dass sie Sub-Pixel genaue Koordinaten liefern. Jedes der fünf hier verwendeten Targets besitzt einen kodierten Wert damit man sie einander zuordnen kann. Die Ermittlung dieses Wertes funktionierte in allen Tests einwandfrei.
Aus den einander zugeordneten Targets wurde danach die äußere Orientierung bestimmt. Zur Berechnung dieser benötigt man mindestens drei Targets. In unseren Tests wurden immer mindestens vier der fünf Targets gefunden. Nicht gefunden wurden sie dann wenn sie nicht auf einem Bild abgebildet waren oder die Auflösung zu schlecht war, um sie als Targets zu identifizieren. Da mehrmals andere Objekte als Targets erkannt wurden (Strukturen im Bild, die den Targets ähneln) wurde zur Berechnung der äußeren Orientierung auch der *Ransac*-Algorithmus mit nachfolgender Optimierung verwendet.

In Kapitel 3.2 wurde der Ecken-Detektor erläutert der, die natürlichen Bildmerkmale in den Bildern findet. Um diese einander zuordnen zu können, wurde eine Methode gezeigt die die äußere Orientierung ausnutzt.

1. Ein Bildmerkmal wird transformieren, um einen besseren Vergleich zu erhalten.

2. Durch die Epipolar-Geometrie werden nur solche Punktpaare miteinander verglichen, die in Wirklichkeit auch möglich sind.

Es zeigte sich, dass dadurch die Laufzeit des Programmes erheblich verkürzt wurde. Da jedoch die Eindeutigkeit der Zuordnung noch nicht so gut war, wurden die Punktpaare zusätzlich nach ihrer Eindeutigkeit sortiert. Erst dadurch war eine relativ gute Zuordnung der Punkte untereinander möglich.

Zuletzt wurden aus den korrespondierenden Bildmerkmalen noch die 3D-Koordinate berechnet und, da diese nicht sonderlich genau waren, alle Parameter noch mittels Bündel-Ausgleich optimiert.

Den Experimenten ist zu entnehmen, dass der Fehler der einzelnen Punktpaare bei durchschnittlich 2 Pixel, und der Fehler der Targets bei durchschnittlich 0.3 Pixel liegt. Der Unterschied entsteht durch die Sub-Pixel genaue lokalisierung der Targets.

Es hat sich gezeigt, dass die hier vorgestellte Methode, nämlich einen Raum dünn zu rekonstruieren, mit vertretbarem Zeitaufwand und ausreichender Genauigkeit möglich ist.

# Anhang A

# Tangente an einer Ellipse

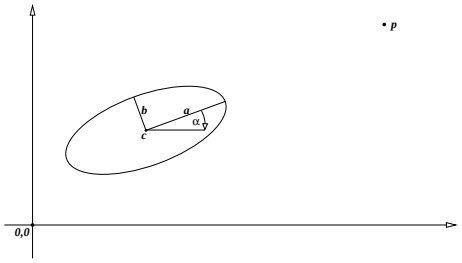



Abbildung A.1: Ellipse mit einem Punkt

Es gibt zwei Tangenten an einer Ellipse, die durch einen Punkt $p$ gehen. Um diese zu Berechnen benötigt man folgende Parameter der Ellipse:

- Zentrum der Ellipse $c = \frac{c_x}{c_y}$
- Koordinaten des Punktes $P = \frac{P_x}{P_y}$
- Orientierung der Ellipse $\alpha$

Die Parameter der Ellipse ($c$, $p$ und $\alpha$) wurden aus den Gauß'schen Parametern berechnet. Die Ellipse als Mittelpunktsgleichung ($c = 0$) ist definiert durch:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1 \tag{A.1}$$

Die Tangente an diese Ellipse ist dann:

$$\frac{xx_0}{a^2} + \frac{yy_0}{b^2} = 1 \tag{A.2}$$

Gesucht ist jener Punkt auf der Ellipse $(\frac{x_0}{y_0})$, dessen Tangente Gleichung A.2 erfüllt und durch Punkt $p$ geht. Dazu muss das Koordinatensystem erst so transformiert werden, dass die Gleichungen anwendbar sind.

$$p = \begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix} (P - c) \tag{A.3}$$

Gleichung A.3 transformiert $p$ so, dass die Ellipse sich im Zentrum des Koordinatensystems befindet und die beiden Hauptachsen der Ellipse mit den Achsen des Koordinatensystems übereinstimmen. Lösen von Gleichung A.2 nach $x_0$, Einsetzen in Gleichung A.3 und Lösen nach $y_0$ ergibt:

$$\begin{aligned} p &= \frac{-2a^2b^2p_y}{a^2{p_y}^2 + b^2{p_x}^2} \\ q &= \frac{b^4(a^2 - {p_x}^2)}{a^2{p_y}^2 + b^2{p_x}^2} \\ y_{0_{1,2}} &= -\frac{p}{2} \pm \sqrt{\frac{p^2}{2} - q} \end{aligned} \tag{A.4}$$

Die Werte für $x_{0_{1,2}}$ erhält man durch Einsetzen von $y_{0_{1,2}}$ in Gleichung A.2. Durch Rücktransformation der erhaltenen Punkte mit

$$\begin{pmatrix} \cos(-\alpha) & -\sin(-\alpha) \\ \sin(-\alpha) & \cos(-\alpha) \end{pmatrix} \begin{pmatrix} x_{0_{1,2}} \\ y_{0_{1,2}} \end{pmatrix} + c \tag{A.5}$$

erhält man die zwei Schnittpunkte der Tangenten.

# Anhang B

# Nichtlineare Optimierung

Als Optimierung von nichtlinearen Funktionen hat sich das Gauss-Newton Verfahren und seine Erweiterung als Levenberg-Marquardt bewährt. Hierbei wird eine Fehlerfunktion $f(x)$ mit einem Startwert $x_i$ beginnend iterativ verkleinert, bis ein lokales Optimum erreicht wird. Die Funktion $f(x)$ wird dazu im Startpunkt $x_i$ durch $g(x)$ linearisiert.

$$g(x) = f(x_i) + \frac{\partial f(x_i)}{\partial x}(x - x_i)$$

Nun wird für $x_i$ die Nullstelle von $g(x)$ gewählt, also

$$\begin{aligned} f(x_i) + \tfrac{\partial f(x_i)}{\partial x}(x_{i+1} - x_i) = 0 \Leftrightarrow \tfrac{\partial f(x_i)}{\partial x} x_{i+1} = \tfrac{\partial f(x_i)}{\partial x} x_i - f(x_i) \Leftrightarrow \\ \Leftrightarrow x_{i+1} = x_i - \tfrac{\partial f(x_i)}{\partial x}^{-1} f(x_i) \end{aligned} \tag{B.1}$$

In Gleichung B.1 wird $\frac{\partial f(x_i)}{\partial x}$ auch als Jacobian $J$ bezeichnet.
Vorgangsweise bei der Optimierung:

1. Berechnen eines Startwert $x_0$.
2. Berechne $x_{i+1} = x_i - J^{-1} f(x_i)$.
3. Falls $\|x_{i+1} - x_i\| < \epsilon$, beende Optimierung, ansonsten setze $i = i + 1$ und gehe zu Punkt 2.

Um nun mit diesem Verfahren Fehlerfunktionen optimieren zu können, muss für jede Fehlerfunktion die Jacobian-Matrix berechnet werden. Die Schrittweite beim Levenberg-Marquardt Verfahren wurde nicht selbst berechnet, sondern dem MATLAB-Befehl *lsqnonlin* überlassen.
In den nachfolgenden Abschnitten B.1 bis B.3 findet man für die Implementierung der verwendeten Optimierungsschritte die Fehlerfunktionen, sowie die dazu gehörigen Jacobian.

## B.1 Homographie

$\Phi$ Die Fehlerfunktion $\Phi$ ergibt sich bei Punktepaaren $p_i$ und $p_j$ in Gleichung 2.9 zu:

$$\begin{pmatrix} x_{ic} \\ y_{ic} \\ 1 \end{pmatrix} = \left\| \begin{pmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & h_9 \end{pmatrix} \begin{pmatrix} x_j \\ y_j \\ 1 \end{pmatrix} \right\|$$

$$\Phi = \begin{pmatrix} \Phi_x \\ \Phi_y \end{pmatrix} = \begin{pmatrix} x_{ic} - x_i \\ y_{ic} - y_i \end{pmatrix} = \begin{pmatrix} -x_i + \frac{(h1*x_j+h2*y_j+h3)}{(h7*x_j+h8*y_j+h9)} \\ -y_i + \frac{(h4*x_j+h5*y_j+h6)}{(h7*x_j+h8*y_j+h9)} \end{pmatrix} = \begin{pmatrix} -x_i + \frac{Z_1}{N} \\ -y_i + \frac{Z_2}{N} \end{pmatrix} \quad \text{(B.2)}$$

Die Ableitung dieser Fehlerfunktion nach den Parametern $h_1$ bis $h_9$ ergibt dann folgende Jacobian:

$$\frac{\partial \Phi}{\partial H} = \begin{pmatrix} \frac{\partial \Phi_x}{\partial h_1} & \frac{\partial \Phi_x}{\partial h_2} & \cdots & \frac{\partial \Phi_x}{\partial h_9} \\ \frac{\partial \Phi_y}{\partial h_1} & \frac{\partial \Phi_y}{\partial h_2} & \cdots & \frac{\partial \Phi_y}{\partial h_9} \end{pmatrix} = \begin{pmatrix} \frac{x_j}{N} & 0 \\ \frac{y_j}{N} & 0 \\ \frac{1}{N} & 0 \\ 0 & \frac{x_j}{N} \\ 0 & \frac{y_j}{N} \\ 0 & \frac{1}{N} \\ -\frac{Z_1}{N^2 * x_j} & -\frac{Z_2}{N^2 * x_j} \\ -\frac{Z_1}{N^2 * y_j} & -\frac{Z_2}{N^2 * y_j} \\ -\frac{Z_1}{N^2} & -\frac{Z_2}{N^2} \end{pmatrix}^T \quad \text{(B.3)}$$

## B.2 Kalibrierung

Zur Kalibrierung werden zu allen Punktpaaren die Weltpunkte $W_i$ aus ihren Vektoren $v_i$ (siehe Gleichung 2.21) geschätzt. Geschätzt deshalb, da die Entfernung vom Kamerazentrum nicht berechnet werden kann und auch keinen Einfluss auf die Optimierung hat.
Die Fehlerfunktion $\Phi$ ergibt sich als Fehler der Abbildung des Weltpunktes $W_i$ in das Kamerabild $p_{ci}$ und des gemessenen Punktes $p_{mi}$ im Bild.
Parameter zur Optimierung sind:

- innere Kamera-Parameter: $f_x$,$f_y$,$x_0$ und $y_0$
- Linsenverzeichnung: $k_1$ bis $k_5$
- sowie für jedes Punktpaar: $W_i$

Die Fehlerfunktion ist somit:

$$\Phi = p_{ci}(W_i, f_x, f_y, x_0, y_0, k_1...k_5) - p_{mi} = \begin{pmatrix} \Phi_x \\ \Phi_y \end{pmatrix} \tag{B.4}$$

Die Ableitungen für $\Phi$ ergeben sich zu:

$$\frac{\partial \Phi}{\partial(W_i, f_x, f_y, x_0, y_0, k_1...k_5)} = \begin{pmatrix} \frac{\partial \Phi_x}{\partial W_i} & \frac{\partial \Phi_x}{\partial f_x} & \frac{\partial \Phi_x}{\partial f_y} & \frac{\partial \Phi_x}{\partial x_0} & \frac{\partial \Phi_x}{\partial y_0} & \frac{\partial \Phi_x}{\partial k_1} & \cdots & \frac{\partial \Phi_x}{\partial k_5} \\ \frac{\partial \Phi_y}{\partial W_i} & \frac{\partial \Phi_y}{\partial f_x} & \frac{\partial \Phi_y}{\partial f_y} & \frac{\partial \Phi_y}{\partial x_0} & \frac{\partial \Phi_y}{\partial y_0} & \frac{\partial \Phi_y}{\partial k_1} & \cdots & \frac{\partial \Phi_y}{\partial k_5} \end{pmatrix} \tag{B.5}$$

## B.3 3D-Koordinaten (Bündel-Ausgleich)

Alle gefundenen Punktpaare in zwei Ansichten und die detektierten Targets werden zur Optimierung herangezogen.
Die Fehlerfunktion $\Phi$ ergibt sich als Fehler der Abbildung des Weltpunktes $W_i$ in das Kamera-Bild $p_{ci}$ und des gemessenen Punktes $p_{mi}$ im Bild.
Im Unterschied zur Kalibrierung werden hier Punktpaare aus zwei verschiedenen Ansichten zu Optimierung der äußeren Orientierung verwendet. Zusätzlich wurde ein neuer Parameter $C_z$ eingeführt. Dieser beschreibt den Fehler, der entsteht, wenn das Kamerazentrum nicht mit dem Rotationszentrum zusammen fällt.
Parameter zur Optimierung sind:

- innere Kamera-Parameter: $f_x$,$f_y$,$x_0$ und $y_0$
- Linsenverzeichnung: $k_1$ bis $k_5$
- sowie für jedes Punktpaar: $W_i$
- äußere Orientierung: $R$ und $t$.
- $C_z$: Fehler durch Kamerazentrum $\neq$ Rotationszentrum

Die Fehlerfunktion ist somit:

$$\Phi = p_{ci}(W_i, R, t, C_z, f_x, f_y, x_0, y_0, k_1 ... k_5) - p_{mi} = \begin{pmatrix} \Phi_x \\ \Phi_y \end{pmatrix} \tag{B.6}$$

Wobei die Fehlerfunktion pro Weltpunkt für zwei Ansichten ausgewertet wird. In der ersten Ansicht gilt $R = I$ und $t = (0, 0, 0)^T$.

Die Ableitungen für $\Phi$ ergeben sich zu:

$$\frac{\partial \Phi}{\partial (W_i, R, t, C_z, f_x, f_y, x_0, y_0, k_1 ... k_5)} = \begin{pmatrix} \frac{\partial \Phi_x}{\partial R} & \frac{\partial \Phi_x}{\partial t} & \frac{\partial \Phi_x}{\partial C_z} & \frac{\partial \Phi_x}{\partial W_i} & \frac{\partial \Phi_x}{\partial f_x} & \frac{\partial \Phi_x}{\partial f_y} & \frac{\partial \Phi_x}{\partial x_0} & \frac{\partial \Phi_x}{\partial y_0} & \frac{\partial \Phi_x}{\partial k_1} & \cdots & \frac{\partial \Phi_x}{\partial k_5} \\ \frac{\partial \Phi_y}{\partial R} & \frac{\partial \Phi_y}{\partial t} & \frac{\partial \Phi_y}{\partial C_z} & \frac{\partial \Phi_y}{\partial W_i} & \frac{\partial \Phi_y}{\partial f_x} & \frac{\partial \Phi_y}{\partial f_y} & \frac{\partial \Phi_y}{\partial x_0} & \frac{\partial \Phi_y}{\partial y_0} & \frac{\partial \Phi_y}{\partial k_1} & \cdots & \frac{\partial \Phi_y}{\partial k_5} \end{pmatrix} \tag{B.7}$$

# Literaturverzeichnis


[AS98] Shai Avidan and Amnon Shashua. Threading fundamental matrices. *Lecture Notes in Computer Science*, 1998.

[BK02] Roland Bunschoten and Ben Kröse. 3d scene reconstruction from cylindrical panoramic images. *Robotics and Autonomous Systems*, Vol.41:111–118, 2002.

[Bou03] Jean-Yves Bouget. Camera calibration toolbox for matlab, Feb. 2003.

[CM99] Qin Chen and Gerard Medioni. Efficient iterative solution to m-view projective reconstruction problem. *IEEE Conf. Computer Vision and Pattern Recognition, Fort Collins,Colorado*, pages 55–61, 1999.

[DJ01] Songtao Dai and Qiang Ji. A new technique for camera self-calibration. *Robotics and Automation*, Vol.3:2165–2170, 2001.

[EHM00] Ian D. Reid Eric Hayman, Lourdes de Agapito and David Murray. The role of self-calibration in euclidean reconstruction from two rotating and zooming camers, 2000.

[Fit01] A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2001.

[HAC02] Kalle Astroem Henrik Aanaes, Rune Fisker and Jens Michael Carstensen. Robust factorization. *Pattern Analysis and Machine Intelligence*, 24(9):1215–1225, Sept. 2002.

[Har92] Richard I. Hartley. *Estimation of Relative Camera Positions for Uncalibrated Cameras*. Springer-Verlag, Jun. 1992.

[HC96] Radu Horaud and Stephane Christy. Euclidian shape and motion from multiple perspective views by affine iterations. *IEEE Transactions on Pattern Analysis and Maschine Intelligence*, 18(11):1098–1104, Nov. 1996.

[HK99] Kyeongtae Hwang and Moon Gi Kang. Correction of lens distortion using point correspondence. *TENCON 99. Proceedings of the IEEE Region 10 Conference*, Vol.1, Sept. 1999.

[HK00a] Jong-Eun Ha and In-So Kweon. Calibration algorithm using known angles. *Electronic Letters*, Vol.36(1):20–22, Jan. 2000.

[HK00b] Mei Han and Takeo Kanade. Creating 3d models with uncalibrated cameras. *Applications of Computer Vision*, Fifth IEEE Workshop:178–185, Dez. 2000.

[HS88] C.G. Harris and M.J. Stephens. A combined corner and edge detector. *Proceedings Fourth Alvey Vision Conference, Manchester*, pages 147–151, 1988.

[HS97] Janne Heikkilä and Olli Silvén. A four-step camera calibration procedure with implicit image correction. *Computer Vision and Pattern Recognition*, pages 1106–1112, Jun. 1997.

[HZ00] Richard I. Hartley and Andrew Zissermann. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[JEHK00] Kuk-Jin Yoon Jong-Eun Ha, Jin-Young Yang and In-So Kweon. Self-calibration using the linear projective reconstruction. *Robotics and Automation*, Vol.1:885–890, Apr. 2000.

[LF96] Q.T. Luong and O.D. Faugeras. The fundamental matrix: theory, algorithms, and stability analysis. *International Journal of Computer Vision*, Vol.17(1):43–75, 1996.

[Mat00] Ginés García Mateos. A camera calibration technique using targets of circular features, Sept. 2000.

[PZ98] Philip Pritchett and Andrew Zisserman. Wide baseline stereo matching. In *Int'l Conf. on Computer Vision*, pages 754–760, Sept. 1998.

[RIHH99] Lourdes de Agapito Richard I. Hartley and Eric Hayman. Linear self-calbration of a rotating and zooming camera. *Computer Vision and Pattern Recognition*, 1, Jun. 1999.

[RIHR99] Lourdes de Agapito Richard I. Hartley, Eric Hayman and Ian Reid. Camera calibration and the search for infinity. *Computer Vision. The Proceedings of the Seventh IEEE International Conference*, pages 510–517, 1999.

[RKG98] Marc Polleyfeys Reinhard Koch and Luc Van Gool. Automatic 3d model acquisition from uncalibrated image sequences. *Proceedings Computer Graphics International*, pages 597–604, 1998.

[SM97] P.H. Storr and D.W. Murray. A review of robust methods to estimate the fundamental matrix. *International Journal of Computer Vision*, Vol.24(3):271–300, 1997.

[SMP01] Y. Omori S. Mahamud, M. Hebert and J. Ponce. Provably-convergent iterative methods for projective structure from motion. *Computer Vision and Pattern Recognition*, 1:1018–1025, Dez. 2001.

[SP01] Franc Solina and Peter Peer. Panoramic depth imaging with a single standard camera. *Image and Signal Processing and Analysis*, pages 170–175, 2001.

[Ste97] Gideon P. Stein. Lens distortion calibration using point correspondences. *Computer Vision and Pattern Recognition*, pages 602–608, 1997.

[TM00] B. Tordoff and D.W. Murray. Violating rotating camera geometry: The effect of radial distortion on self-calibration. *Pattern Recognition.15th International Conference*, Vol.1:423–427, Sept. 2000.

[Tsa87] R Y Tsai. A versatile camera calibration technique for high-accurancy 3d maschine vision metrology using off-the-shelf cameras and lenses. *IEEE Journal of Robotics and Automation*, RA-3(4):323–344, Aug. 1987.

[WM94] Guo-Qing Wei and Song De Ma. Implicit and explicit camera calibration: Theory and experiments. *Pattern Analysis and Machine Intelligence*, Vol.16(5), Mai 1994.

[YHS00] Yuan Bo Yu Hongchuan, Wu Fuchao and Wei Sui. Self-calibration algorithm of rotation cameras. *Intelligent Control and Automation*, 2:1321–1327, Jul. 2000.

[Zha99] Zhengyou Zhang. Flexible camera calibration by viewing a plane from unknown orientations. *Computer Vision. The Proceedings of the Seventh IEEE International Conference*, pages 666–673, Sept. 1999.