

# **Entwicklung eines Verfahrens zur Lagekorrektur von Kalibrierungsmustern in Bildsequenzen**

Studienarbeit  
im Rahmen des Diplomstudienganges Informatik

eingereicht am Institut für Informatik  
der Humboldt-Universität zu Berlin

von Sascha Stübing  
geb. am 22.03.1983  
in Berlin  
Betreuer: Dipl.-Inf. Roman Blaschek  
eingereicht am .....

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>3</b>
1.1 Aufbau der Arbeit . . . . .	3
1.2 Motivation: Projekt - LOGICAL . . . . .	4
<b>2 Grundlagen</b>	<b>6</b>
2.1 Kalibrierungsmuster . . . . .	6
2.2 Kamera Kalibrierung . . . . .	9
2.3 Camera Calibration Toolbox für Matlab . . . . .	13
<b>3 Fehlerhafte Positionsbestimmung</b>	<b>20</b>
3.1 Mögliche Ursachen einer fehlerhaften Positionsbestimmung . . . . .	20
3.2 Korrigierbarkeit fehlerhaft erkannter Positionen . . . . .	23
<b>4 Rekonstruktion der Tiefeninformation</b>	<b>25</b>
4.1 Rückprojektion der 2D-Bild- in 3D-Weltkoordinaten . . . . .	25
4.2 Grundlagen zur Ebene in einem Raum . . . . .	28
4.3 Die Singulärwertzerlegung . . . . .	31
4.4 Ausreißerbehandlung . . . . .	34
<b>5 Implementierung</b>	<b>43</b>
5.1 Berechnung der Bewegungsebene der Objekte . . . . .	43
5.2 Rückprojektion in 3D-Weltkoordinaten . . . . .	47
5.3 Projektion der 3D-Kamera in 2D-Bildkoordinaten . . . . .	48
5.4 Korrektur der fehlerhaften Lösungen . . . . .	49
5.5 Verschiebung der Suchmusterebene . . . . .	50
5.6 Datenerhebung und Auswertung . . . . .	52
<b>6 Zusammenfassung und Ausblick</b>	<b>55</b>
Literaturverzeichnis . . . . .	57

# 1 Einleitung

Die vorliegende Arbeit befasst sich mit einem Verfahren zur Analyse und Korrektur von fehlerhaft erhobenen Positionsdaten. Als Teil einer automatisierten Lagerlogistik erfassen intelligente Kameras die Positionsdaten von Gabelstaplern. Dabei kommt es vor, dass die erhobenen Positionsdaten fehlerhaft sind. Schwerpunkt der Arbeit ist die Korrektur der fehlerhaften Positionsdaten, damit diese für die Lagerlogistik verwendet werden können. Das Verfahren nutzt die Bewegungsebene der Gabelstapler um aus den 2D-Bildkoordinaten der Kameras die korrekten Positionsdaten zu rekonstruieren.

## 1.1 Aufbau der Arbeit

Nach dem Überblick wird zunächst beschrieben, in welchem Zusammenhang die Arbeit entstanden ist.

In Kapitel 2 werden einige wichtige Grundlagen für die Positionsbestimmung von Objekten in Bildern erläutert. Dazu zählen die Vorstellung von Kalibriermustern, die Beschreibung der Zentralprojektion, die Erläuterung der Kameraparameter und die Vorstellung eines Kalibrierverfahrens. Anschließend wird eine Kamera nach dem vorgestellten Kalibrierverfahren kalibriert.

Kapitel 3 beschäftigt sich mit der Klassifikation der Positionsdaten in korrekt und fehlerhaft erkannte Positionen. Dabei wird erklärt, wie sich fehlerhafte Positionsdaten erkennen lassen und ob man sie korrigieren kann.

Im vierten Kapitel wird die Rekonstruktion der Tiefeninformation aus den 2D-Bildsequenzen erläutert. Dazu werden Verfahren zur Rückprojektion der Bildkoordinaten in Weltkoordinaten vorgestellt. Dabei wird begründet, weshalb die Bewegungsebene des Kalibrierungsmusters dafür genutzt werden soll. Zusätzlich wird mit der Singulärwertzerlegung ein elegantes Verfahren zur Lösung von Ausgleichsproblemen eingeführt und erklärt, wie man es zur Berechnung einer Bewegungsebene der Objekte benutzen kann. Am Ende des Kapitels wird geprüft, welche Aussagen sich über die Qualität der Bewegungsebene machen lassen.

Im fünften Kapitel wird die Implementierung der vorgestellten Verfahrens beschrieben. Dabei wird genau auf die benötigten Funktionalitäten eingegangen. Am Ende des Kapitels wird das Verfahren anhand von aufgezeichneten Bildsequenzen getestet und ausgewertet.

Das sechste Kapitel beinhaltet die Zusammenfassung der Arbeit und einen Ausblick.

## 1.2 Motivation: Projekt - LOGICAL

Die Studienarbeit ist im Rahmen des Projekts LOGICAL (Logistik mit intelligenten Kameras und Labeling) entstanden. Dabei handelt es sich um ein Kooperationsprojekt zwischen der Humboldt-Universität zu Berlin und einem privaten Unternehmen, der Firma „LÜTH & DÜMCHEN - Automatisierungsprojekt GmbH“ aus Berlin.

Ziel war es, mittels intelligenter Kameratechnik und RFID-Technologie, eine automatisierte Lagerlogistik zu entwickeln. Dabei sollen Gabelstapler in den Kamerabildern (siehe Abbildung 1) identifiziert, verfolgt und ihre Position bestimmt werden. Wenn man in

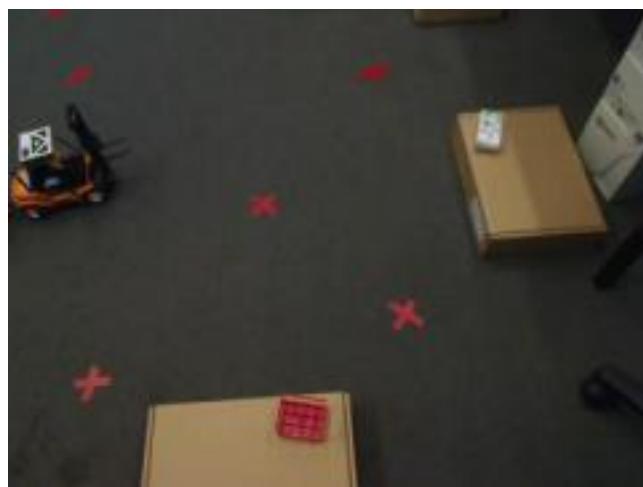


Abbildung 1: Kamerabild eines Lagermodells

der Lage ist den Gabelstapler im Bild zu erkennen, kann man seine Position zu *diesem* Zeitpunkt speichern. Somit lassen sich Aussagen treffen, zu welcher Zeit, sich welcher Gabelstapler, an welcher Position im Lager befunden hat. Über ein RFID-Lesegerät, welches am Gabelstapler angebracht wird, erhält man die Informationen, wann dieser Stapler welche Ware aufgenommen oder abgestellt hat. Eine, ebenfalls am Gabelstapler angebrachte, mobile Recheneinheit überträgt diese Daten über WLAN an eine zentrale Recheneinheit. In Abbildung 2 soll dieser Aufbau verdeutlicht werden.

Führt man die Daten der intelligenten Kamera und des RFID-Lesegerätes zusammen, kann man Aussagen treffen, zu welchem Zeitpunkt ein Gabelstapler welches Lagergut aufgenommen, transportiert oder abgestellt hat und wo er sich zu diesem Zeitpunkt befunden hat. Daraus lässt sich ableiten, wann ein Lagergut in welches Regal eingelagert oder aus ihm entfernt wurde. Diese Daten werden schließlich genutzt, um automatisch eine Logistikdatenbank aufzubauen, in der unter Anderem eingetragen ist, welches Lagergut, in

# 1 Einleitung

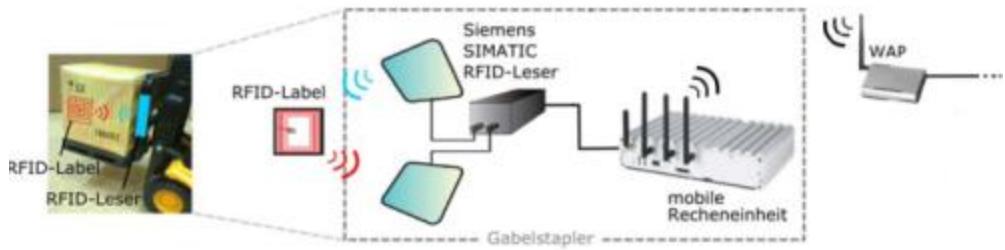


Abbildung 2: Über das RFID-Lesegerät werden die RFID-Tags der Lagergüter ausgelesen und an eine mobile Recheneinheit weitergeleitet. Von dieser gelangen sie über WLAN zu einem Zentralrechner.

welchem Regal abgestellt wurde.

Bei der Positionsbestimmung aus den Bilddaten kommt es unter bestimmten Umständen vor, dass die Lösung, die der Mustererkennungsalgorismus liefert, fehlerhaft ist. Diese fehlerhaften Lösungen werden dann als solche erkannt und die Beobachtung als ungültig verworfen. Ziel dieser Arbeit ist es, ein Verfahren zu entwickeln, mit dem diese fehlerhaften Positionsdaten korrigiert werden können, so dass sie mit den tatsächlichen Positionen des Gabelstaplers übereinstimmen. Das ermöglicht eine bessere Verfolgung des Staplers, da mehr Positionen bekannt sind. Außerdem steigt die Sicherheit, dass die Auf-, und Abladevorgänge tatsächlich erfasst werden.

# 2 Grundlagen

Dieses Kapitel behandelt die grundlegenden Voraussetzungen zur Bestimmung der Position eines Objektes. Zunächst werden drei Kalibriermuster vorgestellt und eines davon als Suchmuster ausgewählt. Anschließend wird im Abschnitt Kamerakalibrierung erläutert, nach welchem Modell die Abbildung eines Punktes in der Welt auf einen Punkt im Kamerabild beschrieben werden kann. Dabei werden die einzelnen Kameraparameter genauer erläutert. Nach den theoretischen Grundlagen wird im dritten Abschnitt mit der *Camera Calibration Toolbox for Matlab* ein Werkzeug vorgestellt, das die Bestimmung der Kameraparameter stark vereinfacht.

## 2.1 Kalibrierungsmuster

Im folgenden Abschnitt werden verschiedene Kalibrierungsmuster vorgestellt und auf ihre Nutzungsmöglichkeit, vor allem im Hinblick auf ihre automatische Erkennbarkeit in Bildsequenzen, untersucht. Ziel soll es sein, aus einem Suchmuster mindestens fünf Punkte zu extrahieren, mit deren Hilfe eine 3D-Lageberechnung durchgeführt werden kann. Das Muster muss demnach so gewählt werden, dass die ausgewählten Punkte eindeutig dem Suchmuster zugeordnet werden können. Ein zufälliges Vorkommen des Kalibrierungsmusters in den Bildsequenzen sollte auszuschließen sein.

### **Das Schachbrettmuster:**

Das Schachbrettmuster, das in Abbildung 3 zu sehen ist, ist ein sehr beliebtes Kalibriermuster. Es besteht aus einem geometrisch sehr einfachen, binären Muster, sich abwechselnde schwarze und weiße Rechtecke. Aufgrund der Anordnung der Rechtecke ist es rotationsinvariant. Die extrahierten Punkte können also eindeutig identifiziert werden. Das einfachste Schachbrettmuster, das die Bedingung erfüllt, ist  $3 \times 4$  Felder groß. Aus diesem Muster lassen sich sechs Punkte extrahieren, die für die Positionsbestimmung genutzt werden können. Dabei werden zunächst die schwarzen Rechtecke erkannt und deren Berührungs punkte zurückgegeben. Diese Punkte lassen sich zwar sehr genau bestimmen, allerdings ist das Verhältnis von benötigter Fläche zu extrahierten Punkten nicht gut. Man benötigt zwölf Rechtecke, um sechs Punkte zu erhalten. Bei geringer Auflösung sind die

## 2 Grundlagen

einzelnen Rechtecke so schlecht zu erkennen, dass diese Punkte nicht zuverlässig gefunden, bzw. fehlerhafte Lösungen zurückgegeben werden.

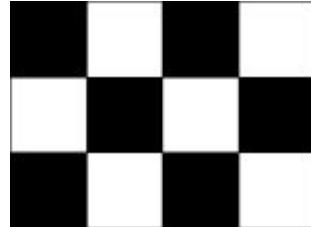


Abbildung 3: Das Schachbrettmuster besteht aus  $3 \times 4$  schwarzen oder weißen, gleich großen Kacheln.

### **Das Dreiecksmuster:**

Das von R. Schnabel und R. Blaschek entworfene Dreiecksmuster ist in Abbildung 4 dargestellt. Es besteht aus einer Reihe, zum Teil verschachtelter, Dreiecke. Das Besondere ist die Kombination aus weißem Dreieck auf schwarzem Hintergrund und schwarzen Dreiecken auf weißem Hintergrund. Die Erkennung des Musters funktioniert ähnlich wie beim Schachbrettmuster. Es werden die Konturen der schwarzen Dreiecke gesucht. Das weiße Dreieck wird dann gefunden, indem die Farben innerhalb des großen schwarzen Dreiecks invertiert werden. Im Unterschied zum Schachbrettmuster werden die Koordinaten der Eckpunkte extrahiert. Auf diese Weise lassen sich zwölf Punkte eindeutig bestimmen. Für



Abbildung 4: Das Dreiecksmuster besteht aus vier schwarzen oder weißen, zum Teil verschachtelten Dreiecken.

die Extraktion der zwölf Punkte werden lediglich vier geometrische Objekte benötigt, wovon zwei ineinander liegen. Das ist ein großer Vorteil, denn bei gleicher Grundfläche des Musters können die geometrischen Objekte wesentlich größer gestaltet werden, als beim Schachbrettmuster. Dadurch ist bei geringerer Auflösung, eine bessere Erkennungsrate als beim Schachbrett möglich. Ein weiterer Vorteil dieses Musters ist, dass es im Rauschen so gut wie nicht vorkommt, da diese Kombination der dunklen und hellen Dreiecke fast nie auftritt. Es wird demnach sehr selten ein Muster detektiert, obwohl keines vorhanden ist.

### Das CODE (AICON)-Muster:

Die CODE Muster der Firma Falcon bestehen aus Punktmarken, die von einem Code-Ring umgeben sind. Durch ihn ist die Nummer des Musters codiert (siehe Abbildung 5). Anhand der Anzahl und Größe der Teilstücke können mehrere Muster unterschieden werden, wie viele genau hängt vom Einsatzgebiet ab. Die Auflösung der Kamera wird hierfür der limitierende Faktor sein. Denn der Hersteller empfiehlt eine Mindestgröße des Musters im Bild von mehr als 15 Pixel. Die Firma Falcon bietet für ihre Muster ein Softwareumgebung zur automatischen Erkennung und Verfolgung der Muster in Grauwertbildern an. Mit dieser können die Trajektorien ausgewertet und Messdaten erhoben werden. Da jedoch Muster und Software lizenziert geschützt sind, werden sie hier nicht weiter untersucht.



Abbildung 5: Das CODE - Muster besteht aus mehreren AICON Markern der Firma Falcon.

### Entscheidung für ein Muster:

Da das CODE Muster der Firma Falcon aus lizenzierten Gründen nicht in Frage kommt, fällt die Entscheidung zwischen dem Schachbrett- und dem Dreiecksmuster. Im direkten Vergleich beider Muster hat sich, was die Erkennungsrate betrifft, das Dreiecksmuster als deutlich stabiler erwiesen. Auch das Auftreten von fehlerhaften Lösungen in mittleren bis großen Abständen zur Kamera war beim Dreiecksmuster geringer. Ein weiterer Vorteil des Dreiecksmuster ist seine besondere Anordnung der Dreiecke, was eine Reduzierung der falsch-positiven Detektionen zur Folge hat. Insgesamt hat sich das Dreiecksmuster gegenüber dem Schachbrettmuster durchgesetzt und wird als Suchmuster Verwendung finden.

Zur Kalibrierung der Kameras wird jedoch das Schachbrettmuster genutzt. Das hat den Vorteil, dass die Funktionen aus der *Camera Calibration Toolbox* für die Kalibrierung

genutzt werden können und keine Anpassungen an den Algorithmen nötig sind.

Ein Vergleich der Schachbrett- und Dreiecksmuster mit dem CODE Muster von Falcon ließ sich nicht realisieren, da weder die Muster noch die dazugehörige Software zur Verfügung standen.

## 2.2 Kamera Kalibrierung

Bei der Kalibrierung einer Kamera werden ihre internen und externen Parameter bestimmt. Mit deren Hilfe lässt sich die zentral-projektive Abbildung eines 3D-Punktes aus der Welt in einen 2D-Punkt im Bild beschreiben. Das Objekt im dreidimensionalen Raum wird auf einen zweidimensionalen Raum, das von der Kamera erzeugte Bild, abgebildet. Die Qualität der Kalibrierung ist dabei umso besser, je mehr Wissen in das Abbildungsmodell eingebracht wird.

### Das Lochkameramodell

Das einfachste Modell, ist das Modell der Lochkamera (siehe Abbildung 6). Bei diesem Modell wird jeder Punkt  $P$  aus der Welt verzerrungsfrei auf den Punkt  $P'$  in der Bildebene abgebildet, indem von  $P$  ausgehend eine Gerade durch das optische Zentrum<sup>1</sup>  $O$  der Kamera gedacht wird.  $P'$  befindet sich an dem Schnittpunkt dieser Geraden mit der Bildebene. Der Abstand zwischen Bildebene und optischen Zentrum wird durch die Brennweite  $f$  beschrieben. Fällt man vom optischen Zentrum ein Lot auf die Bildebene dann erhält man die Position des Bildhauptpunktes  $C$ . Dieser befindet sich im Idealfall im Zentrum der Bildebene. Die Projektion eines 3D-Punktes  $(x_i, y_i, z_i)$  auf den 2D-Bildpunkt  $(u_i, v_i)$

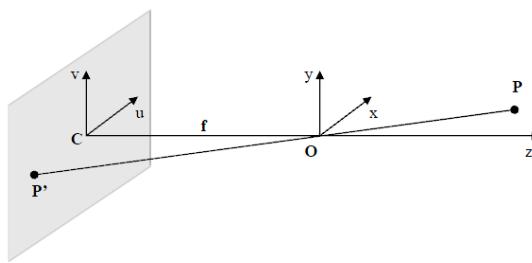


Abbildung 6: Modell der Lochkamera, jeder Punkt  $P$  aus der Welt wird über eine Gerade durch das optische Zentrum der Kamera  $O$  auf einen Punkt  $P'$  in die Bildebene abgebildet.

---

<sup>1</sup>Das optische Zentrum, auch Projektionszentrum genannt, ist in diesem Modell das Loch der Lochkamera.

kann durch (1) beschrieben werden.

$$\begin{bmatrix} \hat{u} \\ \hat{v} \end{bmatrix} = \frac{f}{z_i} \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (1)$$

### Erweiterung des Lochkameramodells zur Verzerrungskorrektur

Das Modell der Lochkamera ist nicht sehr präzise, da nur die Brennweite als Parameter für die Abbildung genutzt wird. Das Projektionszentrum bei einer realen Kamera ist jedoch kein einfaches Loch, sondern besteht aus einer oder mehreren Linsen. Durch diese kann es bei der Abbildung eines Weltpunktes in einen Bildpunkt zu Verzerrungen kommen. Dabei wird meist nur in zwei Arten von Verzerrungen unterschieden: eine radiale und eine tangentiale. Wie sich die beiden Verzerrungstypen im Bild äußern kann man in Abbildung 7 sehen.

Bei der radialen Verzerrung verändert sich der Abstand eines Bildpunktes zum optischen Zentrum, wobei seine Richtung erhalten bleibt (es entstehen Tonnen- oder Kisseneffekte bei rechteckigen Objekten). Bei der tangentialen Verzerrung entsteht eine seitliche Verschiebung des Bildpunktes. Wenn man aus den Bildpunkten Messdaten extrahieren

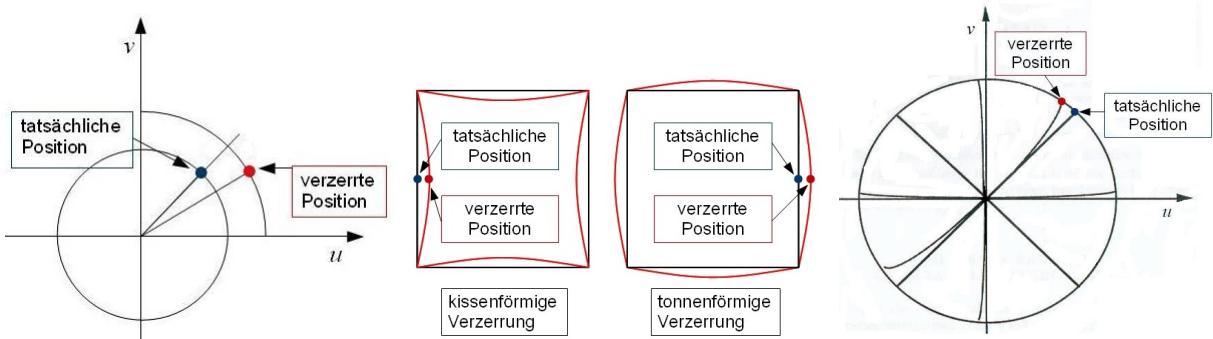


Abbildung 7: Links: Auswirkungen des Auftretens von radialer- und tangentialer Verzerrung. Mitte: Radiale Verzerrung, es treten Kissen- oder Tonneneffekte auf. Rechts: Tangentiale Verzerrung eines Punktes.

möchte, um Aussagen über Objekte in der Welt zu machen, können dabei bereits geringe Verzerrungen zu großen Abweichungen und damit zu großen Verfälschungen der Messdaten führen. Da sich die Eigenschaften der Linsen mathematisch beschreiben lassen, ist es möglich, das Bild so zu korrigieren, dass es einer nahezu idealen Abbildung entspricht. Die radiale Verzerrung kann mit folgender Formel (2) approximiert werden:

$$\begin{bmatrix} \theta u_i^{(r)} \\ \theta v_i^{(r)} \end{bmatrix} = \begin{bmatrix} \hat{u}_i(k_1 \cdot r_i^2 + k_2 \cdot r_i^4 + \dots) \\ \hat{v}_i(k_1 \cdot r_i^2 + k_2 \cdot r_i^4 + \dots) \end{bmatrix} \quad (2)$$

## 2 Grundlagen

wobei  $k_1, k_2, \dots$  Koeffizienten für die radiale Verzerrung sind,  $r_i = \sqrt{\hat{u}_i^2 + \hat{v}_i^2}$  und  $\hat{u}_i, \hat{v}_i$  die Bildkoordinaten aus Formel 1 sind. In der Regel genügen zwei Koeffizienten  $k_1$  und  $k_2$  um die radiale Verzerrung zu beschreiben.

Die tangentiale Verzerrung, die tangential zum Vektor vom optischen Zentrum aus auftritt, hat in der Regel einen geringeren Einfluss (verglichen mit der radialen Verzerrung) auf die gesamte Verzerrung im Bild. Mit Hilfe der Formel (3) kann die tangentiale Verzerrung approximiert werden:

$$\begin{bmatrix} \theta u_i^{(t)} \\ \theta v_i^{(t)} \end{bmatrix} = \begin{bmatrix} 2 \cdot p_1 \cdot \hat{u}_i \cdot \hat{v}_i + p_2 \cdot (r_i^2 + 2 \cdot \hat{u}_i^2) \\ p_1 \cdot (r_i^2 + 2 \cdot \hat{v}_i^2) + 2 \cdot p_2 \cdot \hat{u}_i \hat{v}_i \end{bmatrix} \quad (3)$$

$p_1$  und  $p_2$  sind Koeffizienten für die tangentiale Verzerrung,  $r_i, \hat{u}_i$  und  $\hat{v}_i$  sind wie in Formel 2 definiert. Erweitert man das Modell der Lochkamera um die Approximation der radialen und tangentialem Verzerrung (4), erhält man ein sehr viel genaueres Modell, um die Abbildung der 3D-Objekte in die 2D-Bildecke zu beschreiben.

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} D_u \cdot s_u \cdot (\hat{u}_i + \theta u_i^{(r)} + \theta u_i^{(t)}) \\ D_v \cdot (\hat{v}_i + \theta v_i^{(r)} + \theta v_i^{(t)}) \end{bmatrix} + \begin{bmatrix} c_u \\ c_v \end{bmatrix} \quad (4)$$

Die Koeffizienten  $D_u$  und  $D_v$  werden benötigt, um die Einheit der Bildkoordinaten in Pixel umzurechnen,  $s_u$  ist ein Skalierungsfaktor. Mit  $c_u$  und  $c_v$  werden die Koordinaten des Bildhauptpunktes bezeichnet.

Um die Korrektur der Verzerrung vornehmen zu können, ist es nötig, diese Abbildungsparameter (Kameraparameter) der Kamera zu bestimmen. Dieser Vorgang wird Kamerakalibrierung genannt.

## Kameraparameter

Die Kameraparameter untergliedern sich in intrinsische und extrinsische Kameraparameter, die die innere bzw. äußere Orientierung der Kamera charakterisieren. Durch sie kann eine Abbildung von 3D-Weltkoordinaten in 3D-Kamerakoordinaten (extrinsische Kameraparameter) und von 3D-Kamerakoordinaten in 2D-Bildkoordinaten (intrinsische Kameraparameter) realisiert werden.

### Intrinsische Kameraparameter

Die intrinsischen (inneren) Kameraparameter beschreiben optische, geometrische und digitale Eigenschaften einer Kamera. Sie sind von Kamera zu Kamera verschieden und unabhängig von der Position der Kamera. Durch sie wird die Abbildung vom 3D-Kamerakoordinatensystem in das 2D-Bildkoordinatensystem definiert. Zu den inneren Kameraparametern zählen die Brennweite  $f$ , die Position des Bildhauptpunktes  $c$ , die

## 2 Grundlagen

Koeffizienten der radialen und tangentialen Verzerrung ( $k_1, k_2, p_1, p_2$ ) und ein Verzerrungsparameter  $s$ , den man benötigt, um nicht orthogonale Achsen des CCD-Chips beschreiben zu können.

### Extrinsische Kameraparameter

Die extrinsischen (äußereren) Kameraparameter beschreiben die Position und Orientierung einer Kamera innerhalb des Weltkoordinatensystems, also wo befindet sich das optische Zentrum der Kamera und in welche Richtung zeigt die optische Achse. Wird die Kamera bewegt, so ändern sich diese Parameter entsprechend. Die Transformation vom Kamerakoordinatensystem ins Weltkoordinatensystem wird durch sechs Freiheitsgrade beschrieben (5). Drei Freiheitsgrade für die Rotation ( $r_{11} \dots r_{33}$ ) und drei Freiheitsgrade für die Translation der Kamera  $t_x, t_y$  und  $t_z$ . Die Spaltenvektoren der Transformationsmatrix stellen die Drehung um die Achsen, bzw. die Verschiebung ins Kamerazentrum (mittels  $t_x, t_y$  und  $t_z$ ), dar.

Damit man eine lineare Abbildung erhält, wird die Modellierung vom Euklidischen Vektorraum ( $P=[x, y, z]'$ ) in den projektiven Vektorraum transformiert ( $P=[x, y, z, 1]'$ ). Es werden also homogene Koordinaten für die Beschreibung eines Punktes in der Welt benutzt. Der Vorteil der Hinzunahme einer weiteren Dimension liegt in der Vereinfachung der weiteren Rechenschritte. So kann beispielsweise die Rotation und Translation der 3D-Weltkoordinaten ( $\widetilde{\mathbf{X}}^w$ ) in 3D-Kamerakoordinaten ( $\widetilde{\mathbf{X}}^c$ ) gleichzeitig durchgeführt werden.

$$\text{Transformationsmatrix: } \begin{bmatrix} x^c \\ y^c \\ z^c \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x^w \\ y^w \\ z^w \\ 1 \end{bmatrix} \quad (5)$$

### Kalibrierverfahren

Die Bestimmung der Kameraparameter erfolgt durch Kalibrierung. Dabei existieren im Wesentlichen drei verschiedene Verfahren. Die *Laborkalibrierung*, die *Testfeldkalibrierung* und die *Simultankalibrierung*. Bei der *Laborkalibrierung* werden die intrinsischen Kameraparameter mit einem Goniometer oder mit einem Kollimator bestimmt. Dabei wird der Winkel oder die Richtung der eintreffenden Strahlen durch das Objektiv hindurch gemessen, wodurch sich die Kameraparameter bestimmen lassen. Bei der *Testfeldkalibrierung* wird ein Testfeld mit bekannten Objektpositionen oder Abständen aus unterschiedlichen Winkeln aufgenommen und die Kameraparameter berechnet. Die *Simultankalibrierung* ist ähnlich der Testfeldkalibrierung, wobei hier das Objekt selbst, statt eines Testfeldes zur Kalibrierung genutzt wird. Der Vorteil liegt darin, dass die intrinsischen Kameraparamete-

## 2 Grundlagen

ter exakt für den Zeitpunkt der Objektaufnahmen ermittelt werden und somit höchste Genauigkeit bei der Auswertung erlauben.

In der vorliegenden Arbeit wurde die Kalibrierung der verwendeten USB-Kameras vom Typ *Quickcam Pro 9000* mit einem Testfeld durchgeführt. Die Kameraparameter wurden nach dem *Four-step Camera Calibration Procedure with Implicit Image Correction Algorithmus* von Heikkilä und Silvén [5] durchgeführt. Ihr Algorithmus wurde von Bouguet [2] in einer Matlab-Toolbox (*Camera Calibration Toolbox for Matlab*) implementiert. Der Algorithmus funktioniert folgendermaßen: Zunächst werden mit Hilfe einer *Direct Linear Transformation* (DLT), die bekannten Objektkoordinaten in Bildkoordinaten transformiert, ohne die Verzerrung zu berücksichtigen. Die Parameter der DLT Matrix werden nach der Methode der kleinsten Fehler Quadrate ermittelt, indem die beobachteten Bildpunkte mit den tatsächlichen Objektpunkten verglichen werden und die Parameter mit dem besten Resultat übernommen werden. Aus der DLT-Matrix können nun Kameraparameter abgeleitet werden. Diese sind allerdings noch ungenau. Zum einen wird keine Verzerrung korrigiert und zum anderen ist das Ergebnis stark vom Rauschen gestört. Die aus der DLT bestimmten Parameter werden nun als Ausgangswerte für einen iterativen Algorithmus zur Rauschreduzierung genutzt, bei dem mehrere Beobachtungen verwendet werden, um genauere Werte für die Kameraparameter zu erhalten. Im Anschluss wird über einen rekursiven Ansatz der Projektionsfehler korrigiert. Dabei werden die Objektpunkte mit Hilfe der bisher ermittelten Parametern erneut in die Bildebene abgebildet und die Kameraparameter neu berechnet. Der verbleibende Fehler ist nach einem Durchgang bereits so gering, dass ein weiterer Durchgang keine signifikante Verbesserung bringen würde. Im folgenden Abschnitt wird eine Kamera nach diesem Verfahren kalibriert.

### 2.3 Camera Calibration Toolbox für Matlab

Wie bereits erwähnt, ist in der frei verfügbaren *Camera Calibration Toolbox*, von Bouguet, das Kalibrierverfahren von Heikkilä und Silvén umgesetzt. Die Toolbox zeichnet sich durch eine gute Bedienbarkeit und hohe Genauigkeit bei der Bestimmung der Kameraparameter aus. Im folgenden sollen alle wesentlichen Schritte des Kalibriervorgangs, anhand der Kalibrierung einer, im Projekt verwendeten Kamera, der *Logitech Quickcam Pro 9000*, beschrieben werden. Für eine ausführlichere Beschreibung sei auf [2] verwiesen.

Zum Kalibrieren wird ein Testfeld (Abbildung 8) in Form eines Schachbrettmusters benötigt. Von diesem müssen, mit der zu kalibrierenden Kamera, mehrere Aufnahmen gemacht werden. Dabei sollte man darauf achten, dass das Testfeld in verschiedenen Entfernung und Winkeln positioniert wird. Diese Kalibrierbilder werden für die Bestimmung der Kameraparameter benutzt.

## 2 Grundlagen

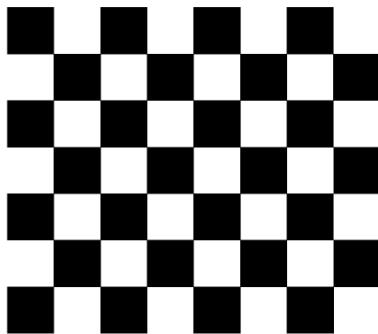


Abbildung 8: Das Testfeld ist ein Schachbrettmuster mit  $20 \times 20\text{mm}^2$  großen Flächen.

Nach dem Hinzufügen der Toolbox zum Matlab-Suchpfad kann man die grafische Oberfläche mit dem Befehl *calib\_gui* starten. In dieser GUI sind sämtliche benötigten Funktionen über einen Button aufrufbar. Im folgenden werden die Bezeichnungen der Buttons als Name der Funktion verwendet.

Zunächst benutzt man die Funktion *Image\_names*, um den Stammnamen und das Format der vorliegenden Kalibrierbilder anzugeben. Alle entsprechenden Bilder werden automatisch geladen und erscheinen zur Kontrolle in einer Übersicht (Abbildung 9).

Im ersten Kalibrierungsschritt müssen in jedem Bild die Eckpunkte des Kalibermusters markiert werden. Dafür wird die Funktion *Extract grid corners* verwendet. Diese verlangt, dass man eine Nachbarschaftsgröße angibt, sowie die Anzahl und Größe der Rechtecke in X- und Y-Richtung. Nun klickt man mit der Maus, in allen Bildern, immer in der gleichen



Abbildung 9: Miniaturansicht aller zur Kalibrierung der Kamera genutzten Bilder.

## 2 Grundlagen

Reihenfolge die äußeren Eckpunkte an. Die Funktion sucht automatisch die Eckpunkte in der Nachbarschaft der markierten Positionen, so dass ein zu präzises Klicken unnötig ist. Die restlichen Eckpunkte des Schachbrettmusters werden dann vollautomatisch gefunden. Das Resultat wird anschließend angezeigt (Abbildung 10).

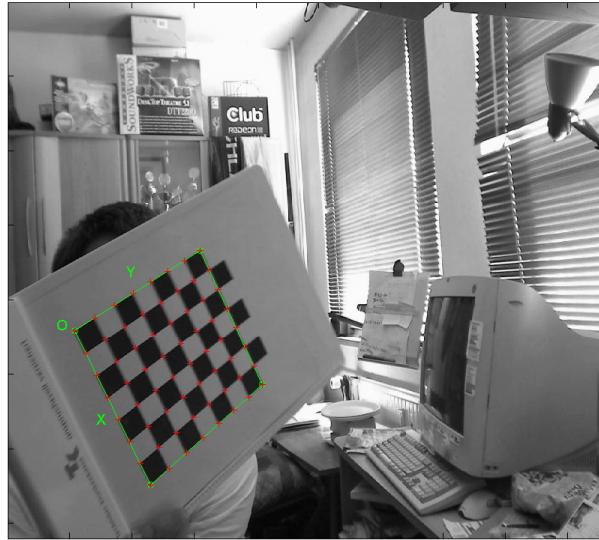


Abbildung 10: Nachdem die äußeren Eckpunkte des Musters per Hand markiert wurden, werden die inneren Ecken (rot) automatisch erkannt.

Nun können mit Hilfe der Funktion *Calibration* die ersten Kameraparameter geschätzt werden. Es werden zunächst, aus den Parametern der DLT Matrix, die Brennweite  $f$  und der Bildhauptpunkt  $c$  bestimmt (Tabelle 1). Diese bilden die Ausgangswerte für eine

Brennweite $f$ :	[ 790.89206    790.89206 ]
Bildhauptpunkt $c$ :	[ 479.50000    359.50000 ]
Verzerrungsparameter $s$ :	[ 0.00000 ] (Winkel zwischen Pixel beträgt 90°)
Verzerrungskoeffizienten $kc$ :	[ 0.00000    0.00000    0.00000    0.00000    0.00000 ]

Tabelle 1: Kalibrierungsparameter nach Schätzung aus DLT.

nichtlineare Optimierung zur Korrektur des Projektionsfehlers, welche genauere Werte für die Brennweite  $f$  und Bildhauptpunkt  $c$  liefert, sowie die Koeffizienten  $kc$ , zur Verzerrungskorrektur bestimmt. Dieser Schritt wird automatisch nach dem Schätzen der DLT durchgeführt und liefert in diesem Fall die in Tabelle 2 zusammengefassten Werte<sup>2</sup>.

<sup>2</sup>Zu den berechneten Werten wird jeweils noch eine maximale Abweichung mit angegeben. Diese werden, aus Gründen der Übersichtlichkeit, nicht mit in die folgenden Tabellen aufgenommen.

## 2 Grundlagen

Brennweite $f$ :	[ 795.82339    797.33150]
Bildhauptpunkt $c$ :	[ 454.37853    345.70229]
Verzerrungsparameter $s$ :	[ 0.00000 ] (Winkel zwischen Pixel beträgt 90°)
Verzerrungskoeffizienten $kc$ :	[ 0.06899    -0.16947    -0.00123    0.00023    0.00000 ]
Projektionsfehler:	[ 0.21003    0.17344 ]

Tabelle 2: Kalibrierungsparameter nach nichtlinearer Optimierung.

Um den Projektionsfehler noch weiter zu reduzieren, werden weitere Funktionen durch die Toolbox bereitgestellt. Mit Hilfe der Funktion *Reproject on Image* kann man sich die Koordinaten der Eckpunkte ins Bild zurück projizieren lassen, siehe Abbildung 11. Die Koordinaten der Anfangs gefundenen Eckpunkte sind als rote Kreuze dargestellt, die

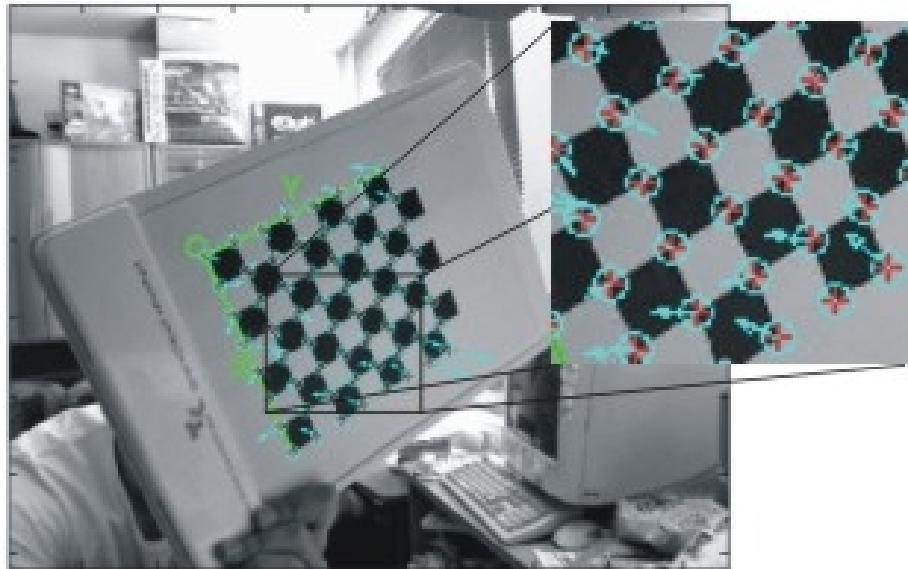


Abbildung 11: Ermittelte Eckpunkte (rote Kreuze) und rückprojizierte Eckpunkte (türkise Kreise)

Koordinaten der projizierten Eckpunkte als Kreise. Die Pfeile stellen die Richtung und Größe des Projektionsfehlers zwischen gefundenen und berechneten Eckpunkt dar. Der Projektionsfehler ist noch relativ groß. Ein Grund hierfür ist, dass die Verzerrung bei einigen Bildern etwas stärker ist und somit die Eckpunkte weiter von ihrer berechneten Position entfernt liegen.

Mit der Funktion *Recomp Corners* werden die Positionen der Eckpunkte automatisch bestimmt. Die Positionen der Eckpunkte aus dem vorherigen Durchgang dienen der Funktion dabei als Ausgangsposition. Nun wird in einer festzulegenden Umgebung dieser Positionen nach den Eckpunkten gesucht. Sind die Eckpunktkoordinaten aktualisiert, kann

## 2 Grundlagen

die Funktion *Calibration* erneut aufgerufen werden. Wie man Tabelle 3 entnehmen kann, konnte der Projektionsfehler um etwa  $\frac{3}{100}$  Pixel nur minimal verringert werden. Für eine

Brennweite $f$ :	[ 795.40776 797.00865 ]
Bildhauptpunkt $c$ :	[ 454.65605 345.11672 ]
Verzerrungsparameter $s$ :	[ 0.00000 ] (Winkel zwischen Pixel beträgt $90^\circ$ )
Verzerrungskoeffizienten $kc$ :	[ 0.06845 -0.16820 -0.00150 0.00037 0.00000 ]
Projektionsfehler:	[ 0.20781 0.17196 ] (in Pixel)

Tabelle 3: Der Projektionsfehler wurde weiter reduziert

weitere Verbesserung kann man sich mittels der Funktion *Analyse error*, die Projektionsfehler aller Eckpunkte der verwendeten Bilder in einer Grafik anzeigen lassen (Abbildung 12). Auf diese Weise ist es möglich, Bilder mit besonders hohem Projektionsfehler

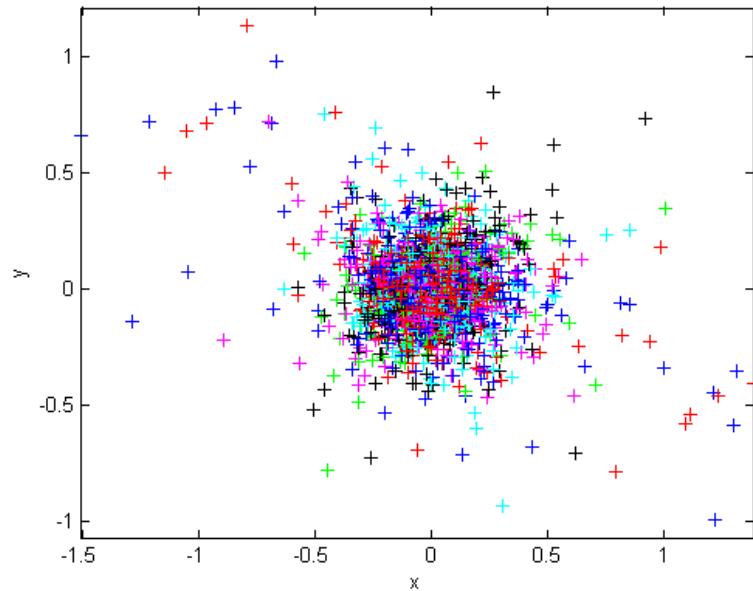


Abbildung 12: Mit Hilfe der Funktion *Analyse Error* wird der Projektionsfehler visualisiert. Je stärker die Abweichung vom Zentrum, desto größer der Projektionsfehler.

zu erkennen. Klickt man auf einen Punkt in der Darstellung, wird der korrespondierende Eckpunkt und die Nummer des Bildes ausgegeben. Findet man mehrere Punkte mit großem Projektionsfehler, die aus einem Bild stammen, ist es sinnvoll, dieses Bild beim Kalibrierungsvorgang nicht zu berücksichtigen.

Um Bilder von der Kalibrierung auszuschließen kann man die Funktion *Add/Suppress images* verwenden. Ihr teilt man die Bildnummer mit, dann wird ein Bild als inaktiv

## 2 Grundlagen

markiert und somit bei der Kalibrierung nicht verwendet. Andererseits kann ein inaktives Bild mit dieser Funktion auch wieder aktiviert werden.

Wenn Bilder als inaktiv markiert wurden, muss die Funktion *Calibration* ein weiteres Mal ausgeführt werden. Bei der Kalibrierung der *Quickcam 9000 pro* wurden in zwei Bildern mehrere Eckpunkte mit großen Projektionsfehlern ( $> 1$  Pixel) gefunden. Ohne diese Bilder bestimmte die Funktion *Calibration* die in Tabelle 4 dargestellten Kameraparameter. Der Projektionsfehler konnte um etwa  $\frac{3}{100}$  Pixel verringert werden. Eine weitere

Brennweite $f$ :	[ 795.64183    797.12639]
Bildhauptpunkt $c$ :	[ 454.94717    345.40055]
Verzerrungsparameter $s$ :	[ 0.00000 ] (Winkel zwischen Pixel beträgt $90^\circ$ )
Verzerrungskoeffizienten $kc$ :	[ 0.06786    -0.16703    -0.00160    0.00042    0.00000 ]
Projektionsfehler:	[ 0.17475    0.15772 ] (in Pixel)

Tabelle 4: Kameraparameter nach erneutem Kalibrierdurchlauf, ohne Bild 7 und 8.

Eliminierung von Bildern brachten keine signifikanten Verbesserungen, so dass die eigentliche Kalibrierung abgeschlossen ist und die Parameter gespeichert werden können. Das übernimmt die Funktion *save*.

Die beiden folgenden Funktionen sind für die Kalibrierung selbst nicht nötig, aber durchaus praktisch. Mit der Funktion *visualize\_distortions*, die nicht als Button in die GUI integriert ist, kann man sich die ermittelten Verzerrungen ansehen. Dabei werden drei Grafiken erzeugt. In der ersten (Abbildung 13) wird das komplette Verzerrungsmodell dargestellt (radiale und tangentiale Verzerrung), in der zweiten Grafik (Abbildung

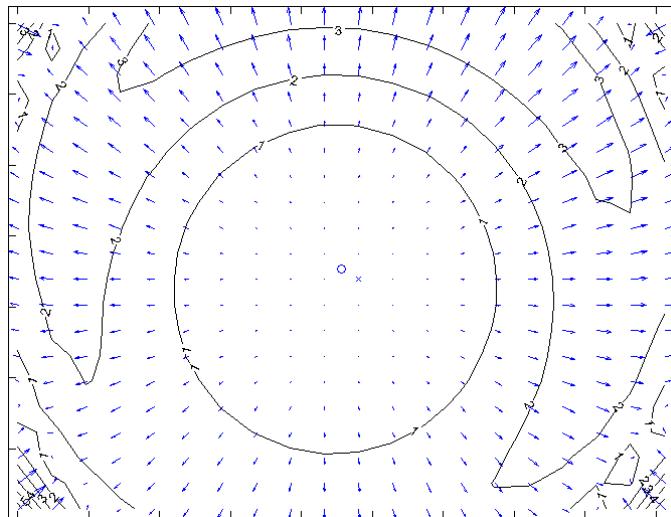


Abbildung 13: Visualisierung für die Verzerrung der kalibrierten Kamera

## 2 Grundlagen

14 links) wird nur der Einfluss der tangentialen Komponente und in der dritten Grafik (Abbildung 14 rechts) wird der Einfluss der radialen Komponente visualisiert. Die Pfeile

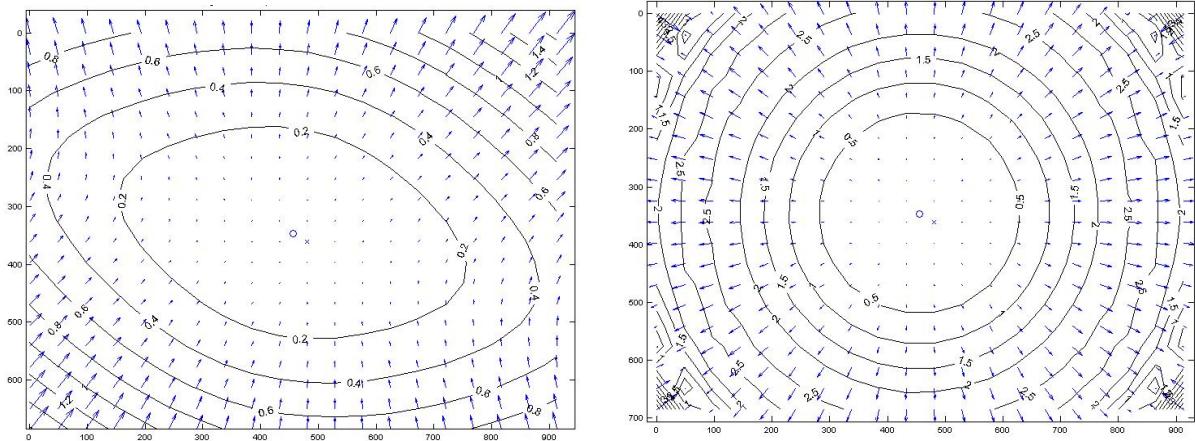


Abbildung 14: Links: Visualisierung der tangentialen Verzerrung. Rechts: Visualisierung der radialen Verzerrung, der Kamera.

in den Abbildungen geben an, wie stark ein einzelner Punkt in der Region durch die Linse verzerrt wird. Im Beispiel beträgt die Verzerrung im Randbereich 3-4 Pixel. Der Anteil der tangentialen Verzerrung ist dabei mit maximal 1.4 Pixel etwas geringer als der der radialen Verzerrung (maximal 4 Pixel). Das Kreuz in der Mitte der Abbildung steht für den Mittelpunkt des Bildes, der Kreis daneben kennzeichnet den Bildhauptpunkt.

Die Funktion *Comp. Extrinsic* ermittelt aus den berechneten intrinsischen Kameraparametern, die extrinsischen Parameter von Testmustern, die nicht zur Kalibrierung genutzt wurden. Sie liefert einen Translationsvektor und eine Rotationsmatrix, welche exemplarisch in Tabelle 5 dargestellt sind.

Damit sind alle wichtigen Funktionen der *Camera Calibration Toolbox* vorgestellt worden. Der Kalibriervorgang der Kamera ist abgeschlossen und die ermittelten Kameraparameter können für die weitere Bildverarbeitung genutzt werden.

Translations Vektor:	[ 56.855160    -18.738488    532.528123]
Rotations Matrix:	[ 0.267516    0.919316    -0.288606 ] [ 0.957444    -0.219939    0.186891 ] [ 0.108337    -0.326320    -0.939030 ]
Projektionsfehler:	[ 0.09848    0.12196] (in Pixel)

Tabelle 5: Ergebnisse der Berechnung der extrinsischen Parameter.

# **3 Fehlerhafte Positionsbestimmung**

In diesem Abschnitt geht es um die Klassifizierung von fehlerhaften Lösungen, die unter bestimmten Umständen bei der Positionsbestimmung des Kalibriermusters (im Folgenden auch Suchmuster genannt) auftreten können. Dabei werden mögliche Ursachen für diese fehlerhafte Lösung erklärt und es wird beschrieben, wie man sie erkennen kann. Anschließend wird untersucht, ob der Fehler behoben werden kann, so dass eine korrekte Lösung der Positionsbestimmung entsteht.

Für ein besseres Verständnis ist es sinnvoll, zunächst die Positionsbestimmung grob zu erläutern.

## **Detektion des Kalibriermusters**

In den Bilddaten der Kamera wird nach dem Kalibriermuster gesucht. Abbildung 15 veranschaulicht das Prinzip. Eine Kamera beobachtet die Szene und sucht nach den verschachtelten Dreiecken des Suchmuster. Wird ein Muster gefunden, kann der Algorithmus, anhand der zwölf, aus dem Dreiecksmuster, extrahierten Punkte die Entfernung zur Kamera und die Ausrichtung des Musters berechnen. Dabei bestimmt er zu den ermittelten Werten die am Besten passende Position des Suchmusters. Die Positionsbeschreibung erfolgt durch einen Translations- und einen Rotationsvektor. Anhand des Identifikationsmusters (rechts in Abbildung 15 zu sehen), welches sich unterhalb des großen Dreiecks befindet, wird das erkannte Suchmuster eindeutig einem Gabelstapler zugeordnet. Das Identifikationsmuster ist ein einfacher Strichcode, der die Identifikationsnummer des Staplers darstellt.

## **3.1 Mögliche Ursachen einer fehlerhaften Positionsbestimmung**

Zunächst soll untersucht werden, welche Ursache eine fehlerhafte Positionsbestimmung haben kann und wie man diese erkennt. Im Allgemeinen kann man bei der Positionsbestimmung drei Fehlertypen unterscheiden:

- Typ 1: ein nicht vorhandenes Kalibriermuster wird erkannt
- Typ 2: ein im Bild vorhandenes Kalibriermuster wird nicht erkannt

### 3 Fehlerhafte Positionsbestimmung

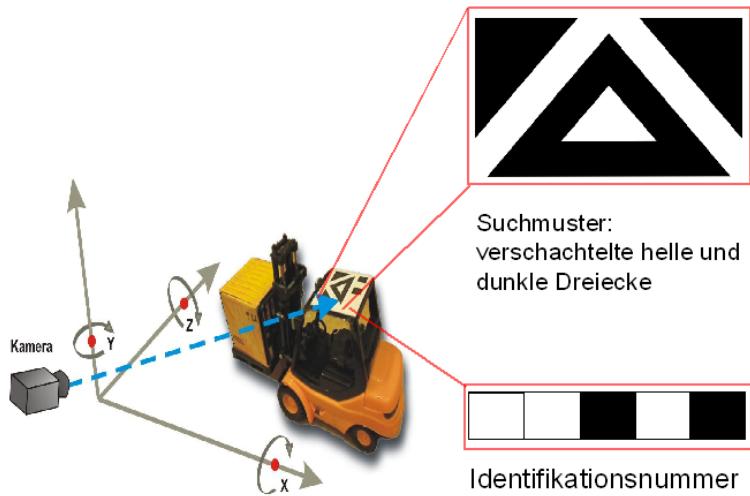


Abbildung 15: Kamera beobachtet die Szene. Auf dem Dach des Gabelstaplers sind das Such- und das Identifikationsmuster angebracht.

- Typ 3: ein im Bild vorhandenes Kalibermuster wird erkannt, die berechnete Position ist fehlerhaft

Das ein nicht vorhandenes Suchmuster vom Algorithmus im Bildrauschen erkannt wird, kommt aufgrund der Struktur des Dreiecksmusters sehr selten vor und soll hier auch nicht weiter untersucht werden. Ebenfalls nicht weiter behandelt wird die Tatsache, dass ein Suchmuster vom Algorithmus nicht erkannt wird, obwohl es im Sichtbereich der Kamera liegt. Die häufigste Ursache dafür ist, dass das abgebildete Muster im Bild zu klein ist und die Dreiecke von zu wenigen Bildpunkten repräsentiert werden. Das Objekt ist dann entweder zu weit von der Kamera entfernt oder die Auflösung der Kamera ist zu gering. Dieses Problem lässt sich also eher durch Anpassungen an der Hardware (mehr Kameras oder höhere Auflösung) als durch Anpassungen am Algorithmus lösen.

Gegenstand der Arbeit ist eine Untersuchung des dritten Fehlertyps. Anhand von Abbildung 16 soll das Problem dargestellt werden. Nach den vom Algorithmus ermittelten Positionsdaten wird eine Box um die Position des Suchmusters gezeichnet, die der Größe und Orientierung des Gabelstaplers entspricht. In Abbildung 16 (links) wird das Suchmuster im Bild gefunden und die Position korrekt berechnet. Der Gabelstapler befindet sich innerhalb der Box. Rechts in Abbildung 16 ist die Situation dargestellt, in der das Suchmuster ebenfalls im Bild korrekt gefunden wurde, die berechnete Position jedoch nicht zu der tatsächlichen Position des Gabelstaplers passt. Die Box liegt oberhalb des Suchmusters. Das bedeutet, die für den Algorithmus am Besten zu den extrahierten Punkten passende Orientierung des Suchmusters, ist die auf dem Kopf stehende. Anhand von Abbildung 17 soll dieses Problem noch einmal genauer verdeutlicht werden. Links im

### 3 Fehlerhafte Positionsbestimmung



Abbildung 16: Links: die Position des Suchmusters konnte korrekt bestimmt werden.  
Rechts: die Positionsbestimmung liefert eine fehlerhafte Lösung.

Bild sieht man die korrekt berechnete Position des Suchmusters, in der Mitte wird eine fehlerhafte Lösung der Positionsbestimmung dargestellt und im rechten Bild die zu der fehlerhaften Lösung passende Position des Gabelstaplers. Die Ursache für dieses kippeln<sup>3</sup>

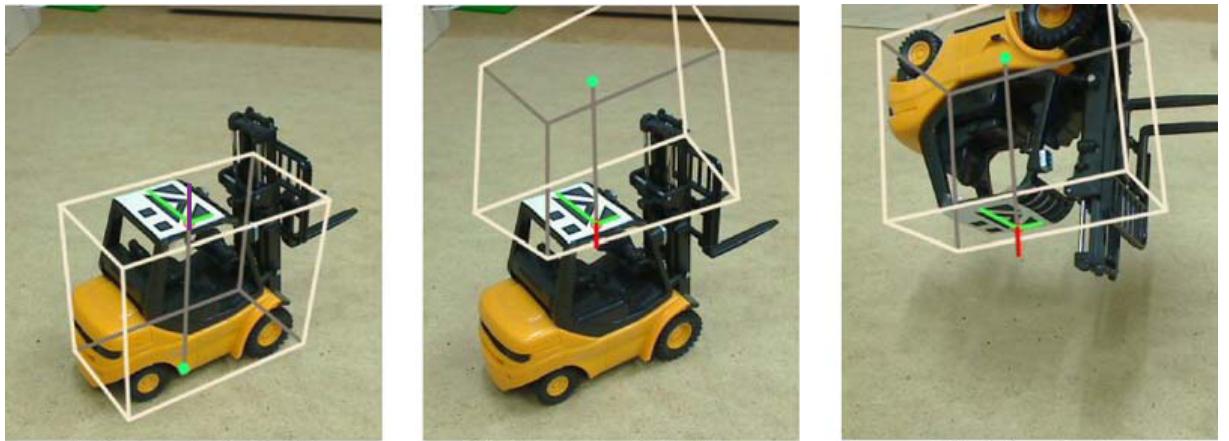


Abbildung 17: Linkes Bild: korrekte Positionsberechnung.  
Mittleres Bild: fehlerhafte Positionsberechnung.  
Rechtes Bild: die passende Position zu der berechneten Lösung.

der Lösung liegt im Bild und im Algorithmus zur Positionsbestimmung. Wenn im Bild nur einige Pixel des Suchmusters nicht korrekt dargestellt sind, sei es durch Rauschen, Verzerrung oder aufgrund der Quantisierung, passt eine andere Rotation des Suchmusters besser, als die tatsächliche. Der auf dem Kopf stehende Gabelstapler stellt also eine, für

<sup>3</sup>Man kann es als kippeln bezeichnen, denn die berechnete Position liegt mal oberhalb des Suchmusters und mal unterhalb des Suchmusters. Die Lösung des Positionsbestimmungsalgorithmus kippelt also zwischen zwei Lösungen.

den Algorithmus gültige Lösung dar. Erkennen lässt sich diese falsche Lösung mit Hilfe von a priori Wissen. Setzt man voraus, dass das Suchmuster ausschließlich von oben gesehen wird, dann zeigt der Normalenvektor der Suchmusterebene (in Abbildung 17 der violette, bzw. rote Strich) in Richtung positive Y-Achse. Verändert sich seine Richtung, liegt eine fehlerhafte Lösung vor. Im Datensatz wird zu jeder Position zusätzlich kodiert, ob es sich um eine korrekte oder fehlerhafte Lösung handelt.

Im Zusammenhang mit dem Rotationsvektor existiert noch ein weiterer Fehler. Es kann durchaus vorkommen, dass die beste Lösung des Algorithmus Rotationen mit Drehwinkeln von weit mehr als  $360^\circ$  enthält. In diesem Fall wird ein Fehler angenommen und die Beobachtung verworfen. Das Resultat erinnert an Fehlertyp 2, bei dem das Suchmuster nicht erkannt wird, obwohl es im Bild enthalten ist.

## 3.2 Korrigierbarkeit fehlerhaft erkannter Positionen

In diesem Abschnitt soll untersucht werden, ob und unter welchen Umständen eine fehlerhafte Positionsbestimmung korrigiert werden kann.

Wie bereits im vorigen Abschnitt erwähnt, liegt der Fehler in einem falsch gewählten Rotationsvektor. Enthält der Vektor viel zu große Werte, werden die Daten sofort als fehlerhaft erkannt und verworfen<sup>4</sup>. In diesem Fall ist keine Korrektur möglich. Wird ein auf dem Kopf stehendes Muster erkannt, so ist zwar der Rotationsvektor ungültig aber das Ergebnis lässt sich prinzipiell korrigieren. In diesem Fall kann anhand der Richtung des Normalenvektors, der Suchmusterebene, die ermittelte Position als fehlerhaft klassifiziert werden. Die im Bild gefundenen Richtungsvektoren des Suchmuster sind trotz des fehlerhaften Normalenvektors gültig. Der korrekte Normalenvektor lässt sich über das Vektorprodukt der Richtungsvektoren rekonstruieren. In Abbildung 17 sind die Richtungsvektoren grün dargestellt. Der Ankerpunkt ist der Schnittpunkt der Richtungsvektoren und bildet den Ursprung der Suchmusterebene. Eine wichtige Eigenschaft dieses Punktes ist die Rotationsinvarianz. Dadurch liegt er auch bei fehlerhaften Lösungen sehr dicht an seiner tatsächlichen Position. Dieser Ankerpunkt repräsentiert die Position des Gabelstaplers. Wenn im späteren Verlauf der Arbeit von Positionsdaten gesprochen wird, sind immer die Koordinaten des Ankerpunktes gemeint.

Um bei einer fehlerhaften Lösung die korrekte Position des Ankerpunktes zu berechnen genügt die Korrektur des Normalenvektors jedoch nicht. Denn die Tiefeninformationen sind bei der Projektion in das 2D-Bildkoordinatensystem verloren gegangen. Ohne diese Information lässt sich die genaue Position des Ankerpunktes nicht bestimmen. Denn es gibt

---

<sup>4</sup>Der Positionsbestimmungsalgorithmus liefert normalerweise Werte aus einem sinnvollen Wertebereich.

Liefert er Werte, die weit außerhalb des Wertebereichs liegen zurück, wie es beispielsweise bei Fehldetections auftritt, ist dies ein Indiz, dass etwas mit dem erkannten Suchmuster nicht stimmt.

### 3 Fehlerhafte Positionsbestimmung

unendlich viele Lösungen, die, aus der Sicht der Kamera, alle entlang einer *Lösungsgeraden* liegen, wie es in Abbildung 18 dargestellt ist. Die verlorenen Tiefeninformationen lassen

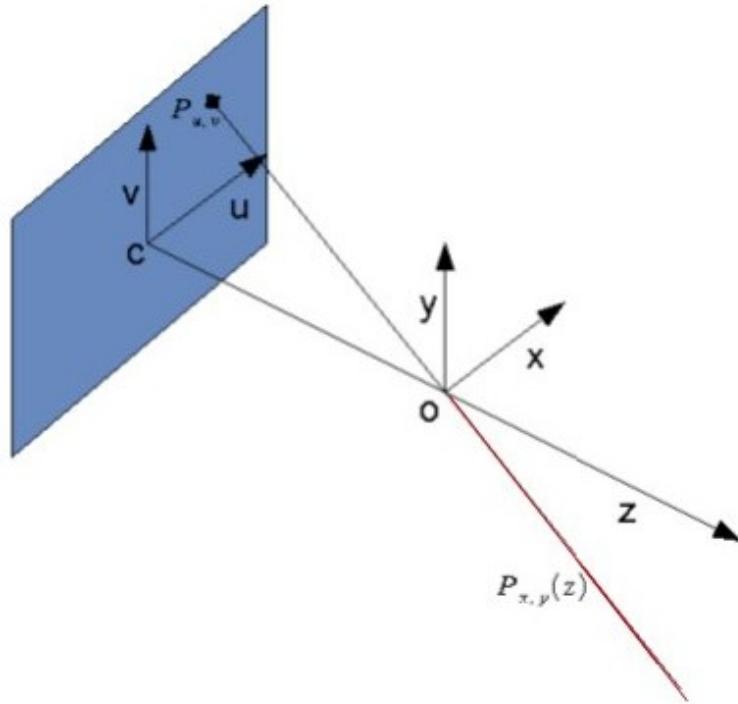


Abbildung 18: Der zum Bildpunkt  $P_{u,v}$  korrespondierende Punkt  $P_{x,y,z}$  in der Welt lässt sich, aufgrund fehlender Tiefeninformationen, nicht eindeutig bestimmen. Es gibt unendlich viele Lösungen, die alle auf einer Lösungsgeraden  $P_{x,y}(z)$ , in der Abbildung rot eingezeichnet, liegen.

sich rekonstruieren, wenn man bestimmte Annahmen voraussetzt:

1. Die Objekte bewegen sich auf einem ebenen Untergrund.
2. Das Suchmuster ist an allen Objekten auf der gleichen Höhe angebracht.

Aus den Annahmen folgt, dass die Ankerpunkte aller gefundenen Suchmuster in einer Ebene liegen. Wenn es gelingt, diese Ebene zu berechnen, dann lassen sich die Tiefeninformationen wiederherstellen. Der Schnittpunkt der Lösungsgeraden mit der Ebene, in der sich die Suchmuster befinden, stellt die korrekte Position der zuvor fehlerhaft bestimmten Lösung dar. Wie genau diese Suchmusterebene berechnet wird, wird im nächsten Kapitel beschrieben.

# 4 Rekonstruktion der Tiefeninformation

In diesem Kapitel wird erklärt, wie die bei der Projektion ins Bild verlorengegangene Tiefeninformation der Raumkoordinaten rekonstruiert werden kann. Zunächst wird beschrieben, wie die Rückprojektion der 2D-Bildkoordinaten in 3D-Weltkoordinaten theoretisch funktioniert. Dabei wird angenommen, dass die Suchmusterebene bekannt ist. Als nächstes soll dann die Suchmusterebene bestimmt werden. Dafür werden kurz einige Darstellungsformen von Ebenen und ihre Überführung ineinander vorgestellt. Anschließend erfolgt eine Einführung in die Singulärwertzerlegung. Dabei wird erklärt, wie sie bei der Bestimmung der Ebenengleichung eingesetzt wird. Am Ende des Kapitels werden einige Ausreißertests vorgestellt und geprüft welches Testverfahren sich am Besten zur Qualitätsverbesserung der Ebene eignet. Diese Qualitätsverbesserung wird erzielt, indem potentielle Ausreißer erkannt und aus der Ebenenberechnung entfernt werden.

## 4.1 Rückprojektion der 2D-Bild- in 3D-Weltkoordinaten

In Kapitel 2 wurde bereits beschrieben, wie ein Objektpunkt aus der Welt in die Bildebene abgebildet wird. Nun soll aus einem Punkt im Bild seine ursprüngliche Position in der Welt bestimmt werden. Dieser Vorgang wird allgemein als Rückprojektion bezeichnet. Es existieren mehrere Verfahren, um die Rückprojektion durchzuführen. Weit verbreitet ist beispielsweise das Verfahren der statischen Stereoanalyse. Wie in Abbildung 19 ange deutet, wird bei der Stereoanalyse eine Szene mit zwei kalibrierten Kameras betrachtet. Die Position des betrachteten Objektpunkts in der Welt erhält man, indem für den Punkt  $(u_1, u_2)$  der einen Kamera der korrespondierende Punkt  $(\bar{u}_1, \bar{u}_2)$  der anderen Kamera, gesucht wird. Hat man den gesuchten Punkt in beiden Bildern gefunden, bestimmt man die Geraden  $S$  und  $\bar{S}$ , die vom Punkt aus durch das optische Zentrum der Kamera verlaufen. Der Schnittpunkt der beiden Geraden stellt die Position des Punktes in Kamerakoordinaten dar. Kann kein Schnittpunkt ermittelt werden (weil  $S$  und  $\bar{S}$  windschief im Raum liegen), dann wird der entsprechende Punkt über Triangulation bestimmt. Die somit berechnete 3D-Position des Schnittpunktes liegt im Bezug zum Kamerakoordinatensystem vor. Um auf die 3D-Weltkoordinate des Punktes zu schließen, muss er noch mit den extrinsischen Parametern, von einer der beiden Kameras, rotiert und verschoben werden. Der Nachteil dieses Verfahrens besteht in einem erhöhten Kalibrier- und Rechenaufwand bei

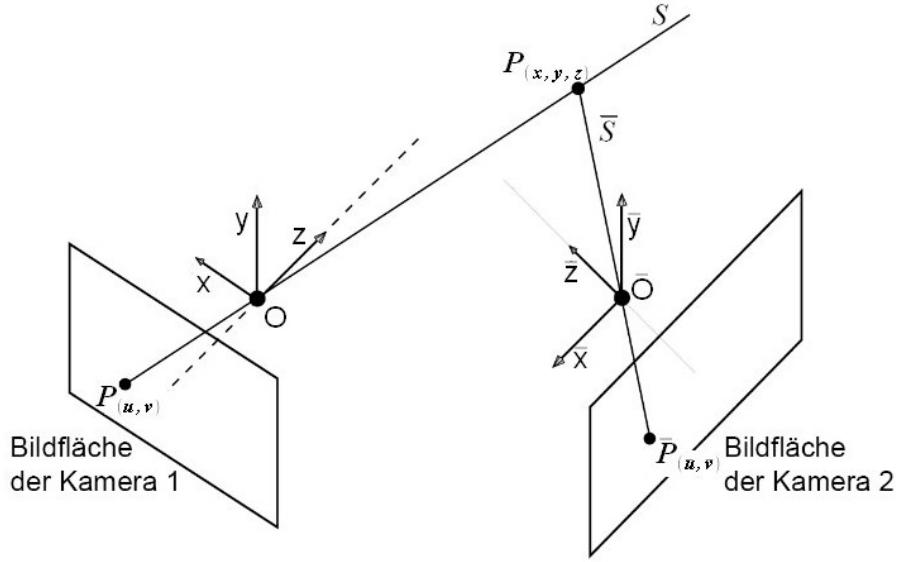


Abbildung 19: Ausgehend von den korrespondierenden Bildpunkten wird ein Strahl durch das optische Zentrum der Kamera projiziert. Der Schnittpunkt der beiden Strahlen stellt die zugehörige 3D-Position dar.

der Korrespondenzsuche. Außerdem verdoppelt sich die Anzahl der benötigten Kameras. Vor allem der hohe Rechenaufwand bei der Korrespondenzsuche und bei der Auswertung der verdoppelten Anzahl an Kamerabildern führten zu der Auswahl eines anderen Verfahrens.

Das angewandte Verfahren basiert auf der Annahme, dass sich das Suchmuster immer in der selben Ebene bewegt. Ähnlich des Geradenschnitts bei der Stereoanalyse erhält man die gesuchte 3D-Position im Kamerakoordinatensystem, indem man den Schnittpunkt  $P'$  der Geraden  $S$  mit der Ebene  $E$  berechnet. In Abbildung 20 ist die Idee dargestellt.

Für die Berechnung der 3D-Position muss also zunächst die Gerade  $S$  bestimmt werden. Die Geradengleichung für  $S$  erhält man, aus der Transformationsformel für die Projektion. Für die Projektion gilt folgende Transformation:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \frac{f}{z} \cdot \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} c_u \\ c_v \end{bmatrix} \quad (6)$$

Dabei sind  $[u, v]'$  die Bildkoordinaten,  $[x, y]'$  und  $z$  sind die 3D-Koordinaten aus dem Kamerakoordinatensystem,  $f$  bezeichnet die Brennweite und  $[c_u, c_v]'$  sind die Bildkoordinaten des Hauptpunktes. Stellt man die Gleichung nach  $[x, y]'$  um,

$$\begin{bmatrix} x \\ y \end{bmatrix} = \left( \begin{bmatrix} u \\ v \end{bmatrix} - \begin{bmatrix} c_u \\ c_v \end{bmatrix} \right) \cdot \frac{z}{f} \quad (7)$$

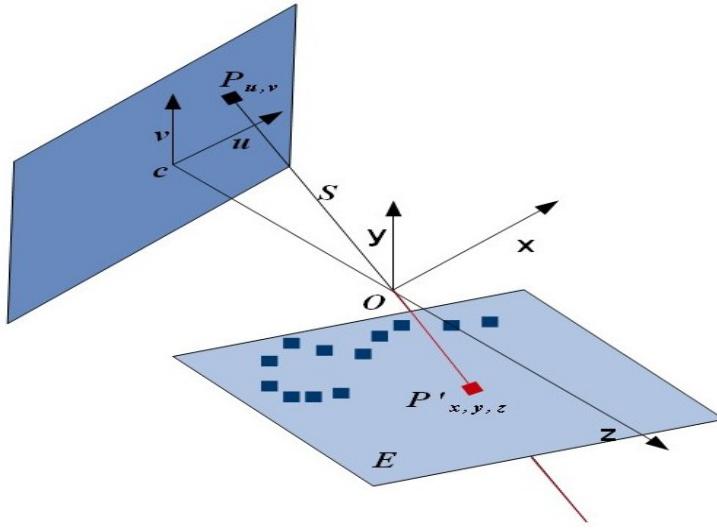


Abbildung 20:  $P_{u,v}$  soll aus dem Bildkoordinatensystem ins Kamerakoordinatensystem zurück projiziert werden. Die Position von  $P'_{x,y,z}$  ist der Schnittpunkt der Geraden  $S$  mit der Suchmusterebene  $E$ . Die Rechtecke stellen weitere Positionen des Staplers dar.

erhält man die Geradengleichungen  $x = (u - c_u) \cdot \frac{z}{f}$  und  $y = (v - c_v) \cdot \frac{z}{f}$ . Diese Gleichungen werden in die Ebenengleichung<sup>5</sup>  $E : a \cdot x + b \cdot y + c \cdot z + d = 0$  für  $x$  und  $y$  eingesetzt, um  $z$  zu berechnen.

$$\begin{aligned}
 a \cdot \left( (u - c_u) \cdot \frac{z}{f} \right) &+ b \cdot \left( (v - c_v) \cdot \frac{z}{f} \right) + c \cdot z + d = 0 && \text{durch } z \text{ teilen} \\
 a \cdot \left( (u - c_u) \cdot \frac{1}{f} \right) &+ b \cdot \left( (v - c_v) \cdot \frac{1}{f} \right) + c + \frac{d}{z} = 0 && \text{nach } z \text{ umstellen} \\
 -\frac{a}{d} \cdot (u - c_u) \cdot \frac{1}{f} &- \frac{b}{d} \cdot (v - c_v) \cdot \frac{1}{f} - \frac{c}{d} = \frac{1}{z} && \text{Kehrwert bilden} \\
 \Rightarrow -\frac{a}{d} \cdot (u - c_u) \cdot \frac{1}{f} - \frac{b}{d} \cdot (v - c_v) \cdot \frac{1}{f} - \frac{c}{d} &= z && (8)
 \end{aligned}$$

Wenn man  $z$  in Gleichung 7 einsetzt erhält man die dazugehörige  $x$ - und  $y$ -Koordinate.

$$x = (u - c_u) \cdot \frac{z}{f}, \quad y = (v - c_v) \cdot \frac{z}{f} \quad (9)$$

Somit lässt sich aus den 2D-Bildkoordinaten mit Hilfe der Ebenengleichung der Suchmusterebene die 3D-Position im Kamerakoordinatensystem rekonstruieren. Um aus den 3D-Kamerakoordinaten die 3D-Weltkoordinaten zu erhalten, müssen sie mit der zu Formel

---

<sup>5</sup>Die verschiedenen Ebenengleichungen werden im nächsten Abschnitt genauer beschrieben.

5 inversen Transformationsmatrix transformiert werden.

$$\begin{bmatrix} x^w \\ y^w \\ z^w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} x^c \\ y^c \\ z^c \\ 1 \end{bmatrix} \quad (10)$$

Die nächsten Abschnitte behandeln die Berechnung der Suchmusterebene.

## 4.2 Grundlagen zur Ebene in einem Raum

In diesem Abschnitt werden grundlegende Eigenschaften von Ebenen beschrieben. Schwerpunkt sind dabei die verschiedenen Darstellungsformen und ihre Vorteile.

Eine Ebene lässt sich durch drei nicht kollineare Punkte eindeutig festlegen. Im Allgemeinen existieren drei verschiedene Formen, um eine Ebene zu repräsentieren. Die *Koordinatenform*, die *Parameterform* und die *Normalendarstellung*. Jede dieser Darstellungsformen kann in eine der anderen überführt werden. Im Folgenden werden sie kurz beschrieben. Außerdem wird die Umformung exemplarisch anhand der Überführung der Koordinatenform in die beiden anderen gezeigt.

### Koordinatenform

Eine Ebene besteht aus unendlich vielen Punkten, wobei jeder dieser Punkte  $X = [x, y, z]'$  im Raum durch seine drei Koordinaten  $x, y$  und  $z$  eindeutig beschrieben ist. Die Koordinatendarstellung stellt die Ebene als eine Funktion dieser Punkte dar. Sie hat die Form:  $E : a \cdot x + b \cdot y + c \cdot z = d$  Dabei sind  $a, b, c$  und  $d$  reelle Koeffizienten, und  $x, y, z$  die Koordinaten der Punkte. Erfüllt ein Punkt diese Gleichung, so befindet er sich genau in der Ebene. Ein Vorteil der Koordinatenform ist, dass das Ergebnis beim Einsetzen eines Punktes den Abstand des Punktes zur Ebene darstellt. Liegt der Punkt in der Ebene, dann ist der Abstand Null, liegt er außerhalb der Ebene, ist der Wert ungleich Null. Normiert man die Koeffizienten der Ebenengleichung erhält man den Abstand im Längenmaß des verwendeten Koordinatensystems (z. B. in cm). Das sorgt für eine bessere Interpretierbarkeit des Abstands.

### Parameterform

Um eine Ebene in Parameterform darzustellen, benötigt man einen Stützpunkt  $s$ , der Punkt in der Ebene ist und zwei linear unabhängige Richtungsvektoren  $r_1, r_2$ . Mit Hilfe von Linearkombinationen der beiden Richtungsvektoren lassen sich alle Punkte der Ebene

erreichen. Möchte man bestimmen, ob ein Punkt  $X = (x, y, z)'$  in der Ebene liegt, muss man überprüfen, ob sich der Stützpunkt nach  $X$  verschieben lässt. Dazu untersucht man, ob es Parameter  $a, b$  gibt, die das Gleichungssystem erfüllen.

$$E : \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} s_x \\ s_y \\ s_z \end{bmatrix} + a \cdot \begin{bmatrix} r_{1x} \\ r_{1y} \\ r_{1z} \end{bmatrix} + b \cdot \begin{bmatrix} r_{2x} \\ r_{2y} \\ r_{2z} \end{bmatrix} \quad (11)$$

Die Parameterform lässt sich sehr einfach erzeugen. Man benötigt drei nicht kollineare Punkte der Ebene, bestimmt einen als Stützvektor und erhält die beiden Richtungsvektoren aus den Differenzen der Punkte mit dem Stützvektor.

## Normalendarstellung

Die Normalendarstellung ist eine sehr kompakte Darstellungsform. Man benötigt einen Punkt  $P$  der Ebene und einen Normalenvektor  $\mathbf{n}$ , der senkrecht auf der Ebene steht. Um zu überprüfen, ob ein beliebiger Punkt  $X = (x, y, z)$  in der Ebene liegt, berechnet man zunächst die Differenz zum Punkt  $P$ . Wenn  $X$  in der gleichen Ebene liegt, dann liegt der Differenzvektor  $(X - P)$  ebenfalls in der Ebene. Anschließend wird das Skalarprodukt zwischen Differenzvektor und Normalenvektor gebildet. Ist der Normalenvektor orthogonal zum Differenzvektor, so ist das Skalarprodukt der beiden Vektoren Null und der Punkt liegt in der Ebene. Die Normalendarstellung besitzt demnach folgende Form:

$$\langle X - P, \mathbf{n} \rangle = 0 \quad (12)$$

Verwendet man statt des Normalenvektors den auf die Länge Eins normierten Normalenvektor, so erhält man die sogenannte Hessesche Normalenform.

$$\frac{\langle X - P, \mathbf{n} \rangle}{|\mathbf{n}|} = 0 \quad \text{mit} \quad |\mathbf{n}| = \sqrt{n_x^2 + n_y^2 + n_z^2} \quad (13)$$

Der Vorteil bei dieser Darstellung ist der, dass man aus ihr direkt den Abstand des Punktes zur Ebene entnehmen kann. Dabei bedeutet Null, dass sich  $X$  in der Ebene befindet. Ein Wert ungleich Null steht für den Abstand zur Ebene im Längenmaß des verwandten Koordinatensystems.

## Umformung der Koordinatenform in eine andere Ebenendarstellung

Jede Ebene kann durch jede der vorgestellten Darstellungsformen repräsentiert werden und jede Darstellungsform kann in eine andere überführt werden. Da im Verlauf der Arbeit die Ebene hauptsächlich in Koordinatenform verwendet wird, soll ausgehend von der Koordinatenform die Umrechnung in die Parameterform und in die Normalendarstellung

gezeigt werden.

**Koordinatenform → Normalendarstellung:** Möchte man aus der Koordinatenform die Normalendarstellung erzeugen, sollte man folgendermaßen vorgehen: Den Normalenvektor  $\mathbf{n}$  erhält man direkt aus den Koeffizienten der Koordinatenform, denn es gilt:

$$\langle X - P, \mathbf{n} \rangle = 0 \quad (14)$$

$$\begin{aligned} \Rightarrow x \cdot n_x - p_x \cdot n_x + y \cdot n_y - p_y \cdot n_y + z \cdot n_z - p_z \cdot n_z &= 0 \\ \Rightarrow n_x \cdot x + n_y \cdot y + n_z \cdot z &= n_x \cdot p_x + n_y \cdot p_y + n_z \cdot p_z \end{aligned} \quad (15)$$

Ersetzt man nun  $n_x \cdot p_x + n_y \cdot p_y + n_z \cdot p_z$  durch  $d$ , dann erhält man:

$$n_x \cdot x + n_y \cdot y + n_z \cdot z = d. \quad (16)$$

Demnach kann man aus dem Normalenvektor direkt die Koeffizienten der Koordinatenform ablesen. Umgekehrt sind die Koeffizienten  $a, b, c$  der Koordinatenform die Elemente des Normalenvektors. Der benötigte Punkt  $P$  aus der Ebene wird frei gewählt, indem man beispielsweise in die Ebenengleichung ( $a \cdot x + b \cdot y + c \cdot z = d$ ) für  $y = 1$  und  $z = 0$  einsetzt und damit  $x = \frac{d-b}{a}$  erhält. Die Normalendarstellung besitzt folgende Form:

$$E : \left\langle \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} \frac{d-b}{a} \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} a \\ b \\ c \end{bmatrix} \right\rangle = 0 \quad (17)$$

**Koordinatenform → Parameterform:** Aus einer gegebenen Ebenenbeschreibung in Koordinatenform erhält man die Ebenengleichung in Parameterform, indem man die Koordinatenform nach einer Variablen umstellt und die anderen beiden Variablen durch zwei Parameter ersetzt.

Zunächst die Gleichung der Koordinatenform nach  $x$  umstellen:

$$E : a \cdot x + b \cdot y + c \cdot z = d \Rightarrow x = \frac{d}{a} - \frac{b}{a} \cdot y - \frac{c}{a} \cdot z \quad (18)$$

Dann wird  $y$  durch  $s$  und  $z$  durch  $t$  ersetzt:

$$E : x = \frac{d}{a} - \frac{b}{a} s - \frac{c}{a} t, \quad y = s, \quad z = t \quad (19)$$

Schließlich werden die einzelnen Komponenten zusammengefasst:

$$E : \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \frac{d}{a} & -\frac{b}{a} \cdot s & -\frac{c}{a} \cdot t \\ 0 & s & 0 \\ 0 & 0 & t \end{bmatrix} = \begin{bmatrix} \frac{d}{a} \\ 0 \\ 0 \end{bmatrix} + s \cdot \begin{bmatrix} -\frac{b}{a} \\ 1 \\ 0 \end{bmatrix} + t \cdot \begin{bmatrix} -\frac{c}{a} \\ 0 \\ 1 \end{bmatrix} \quad (20)$$

Nachdem gezeigt wurde, in welcher Form man Ebenen darstellen kann, soll im nächsten Abschnitt ein Verfahren zum Lösen von Ausgleichsproblemen vorgestellt werden.

## 4.3 Die Singulärwertzerlegung

Wie im vorigen Abschnitt beschrieben, benötigt man für die Berechnung einer Ebene drei nicht kollineare Punkte. Sind jedoch mehr als drei Punkte gegeben, so hat man zwei Möglichkeiten:

1. Man sucht sich willkürlich drei nicht kollineare Punkte aus und erhält eine Ebene in der diese drei Punkte liegen.
2. Man zieht alle Punkte zu einer Ebenenberechnung heran und bestimmt diejenige Ebene, welche den geringsten Abstand zu allen verwendeten Punkten besitzt.

Die erste Möglichkeit ist zu stark von der Wahl der drei Punkte abhängig. Gelingt es, zufällig drei Punkte, die ideal die Ebene repräsentieren, auszuwählen, dann ist das Resultat gut. Allerdings kann man ebenso drei Punkte auswählen, die überhaupt nicht zu den anderen Punkten passen und damit ein falsches Ergebnis liefern. Abbildung 21 illustriert, wie verschieden zwei Ebenen sein können, wenn man drei aus vier Punkten auswählen kann, also zwei Punkte gleich sind. In der linken und mittleren Abbildung wurden die schwarzen Punkte jeweils für die Ebenenberechnung genutzt. Das trotz zweier gleicher Punkte sehr unterschiedliche Ebenen entstehen können, zeigt die rechte Abbildung.

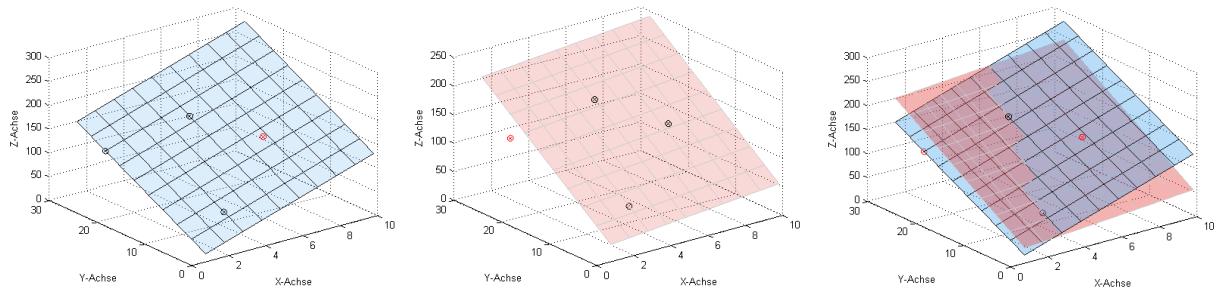


Abbildung 21: Die linke und mittlere Ebene unterscheiden sich in einem Stützpunkt. Rechts sieht man die Abweichungen.

Mit der zweiten Möglichkeit erreicht man hingegen mit hoher Wahrscheinlichkeit eine wesentlich bessere Repräsentation der gegebenen Punkte, da man nicht auf das Glück beim Punkte auswählen angewiesen ist. Allerdings ist die Berechnung der am Besten passenden Ebene sehr aufwendig. Es gilt für das überbestimmte Gleichungssystem mit N Positionsdaten, die alle die Ebenengleichung E:  $a \cdot x + b \cdot y + c \cdot z + d = 0$  (vier Unbekannte)

#### 4 Rekonstruktion der Tiefeninformation

erfüllen sollen, eine Lösung zu finden. Also:

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ X_N \end{bmatrix} = \begin{bmatrix} a \cdot x_1 + b \cdot y_1 + c \cdot z_1 + d = 0 \\ a \cdot x_2 + b \cdot y_2 + c \cdot z_2 + d = 0 \\ \vdots \\ a \cdot x_N + b \cdot y_N + c \cdot z_N + d = 0 \end{bmatrix} \quad (21)$$

Zu diesem Zweck wird die Singulärwertzerlegung genutzt. Sie ist eine elegante Möglichkeit um Ausgleichprobleme, wie z. B. überbestimmte Gleichungssysteme, zu lösen. Solche Probleme besitzen keine eindeutige Lösung. Mit Hilfe der Singulärwertzerlegung erhält man eine optimale Lösung im Sinne der kleinsten Fehlerquadrate.

Die Singulärwertzerlegung einer Matrix ist ihre Zerlegung in drei spezielle Einzelmatrizen, deren Produkt diese Matrix darstellt. So lässt sich Matrix  $\mathbf{A}$  in die drei Matrizen  $\mathbf{U}$ ,  $\mathbf{S}$ ,  $\mathbf{V}$  zerlegen:

$$\mathbf{A}_{n \times p} = \mathbf{U}_{n \times n} \cdot \mathbf{S}_{n \times p} \cdot \mathbf{V}_{p \times p}^T \quad (22)$$

Man kann sie bei jeder Matrix durchführen und ist nicht, wie bei den Eigenwerten, auf quadratische Matrizen beschränkt. Aus diesen Einzelmatrizen können die Singulärwerte der Matrix abgelesen werden. Die Linkssingulärwertmatrix  $\mathbf{U}$  und die Rechtssingulärwertmatrix  $\mathbf{V}$  sind orthogonale Matrizen,  $\mathbf{S}$  ist eine Diagonalmatrix, die die Singulärwerte von  $\mathbf{A}$  enthält. Nach Forbes [4] enthält der Eigenvektor, der zum kleinsten Singulärwert von  $\mathbf{B} = \mathbf{A}^T \mathbf{A}$  gehört, die Koeffizienten der besten Lösung des überbestimmten Gleichungssystems. Wobei die beste Lösung diejenige ist, die für alle Beobachtungen den kleinsten Fehler liefert. Den entsprechenden Eigenvektor, mit den zum kleinsten Singulärwert gehörenden Koeffizienten, findet man laut Forbes [4] in der  $p$ -ten Zeile der Rechtssingulärwertmatrix  $\mathbf{V}$ .

Ein einfaches Beispiel soll die Anwendung der Singulärwertzerlegung verdeutlichen. Gegeben sind zehn Beobachtungen in homogenen Koordinaten. Diese werden in einer Designmatrix  $\mathbf{A}$  angeordnet:

$$\mathbf{A}_{10 \times 4} = \begin{bmatrix} 100 & 617 & 1452 & 1 \\ 113 & 615 & 1453 & 1 \\ 113 & 615 & 1453 & 1 \\ 104 & 648 & 1524 & 1 \\ -79 & 637 & 1474 & 1 \\ -11 & 591 & 1453 & 1 \\ 19 & 564 & 1447 & 1 \\ 48 & 525 & 1406 & 1 \\ 70 & 485 & 1367 & 1 \\ 84 & 436 & 1301 & 1 \end{bmatrix}$$

#### 4 Rekonstruktion der Tiefeninformation

$$\mathbf{B}_{4 \times 4} = \mathbf{A}^T \mathbf{A} = \begin{bmatrix} 67337 & 317748 & 799600 & 561 \\ 317748 & 3331135 & 8252776 & 5733 \\ 799600 & 8252776 & 20569118 & 14330 \\ 561 & 5733 & 14330 & 10 \end{bmatrix}$$

Die Singulärwertzerlegung von B liefert folgende Einzelmatrizen:

$$\mathbf{U}_{4 \times 4} = \begin{bmatrix} -0,0361 & 0,9871 & 0,1558 & -0,0000 \\ -0,3724 & -0,1580 & 0,9145 & 0,0008 \\ -0,9274 & 0,0250 & -0,3733 & -0,0010 \\ -0,0006 & 0,0002 & -0,0011 & 1,0000 \end{bmatrix},$$

$$\mathbf{S}_{4 \times 4} = 10^7 \cdot \begin{bmatrix} 2,3914 & 0 & 0 & 0 \\ 0 & 0,0037 & 0 & 0 \\ 0 & 0 & 0,0017 & 0 \\ 0 & 0 & 0 & 0,000029 \end{bmatrix},$$

$$\mathbf{V}_{4 \times 4}^T = 10^{-2} \cdot \begin{bmatrix} -3,6058 & 98,7128 & 15,5812 & -0,0039 \\ -37,2389 & -15,7956 & 91,4536 & 0,0822 \\ -92,7376 & 2,5047 & -37,3291 & -0,1025 \\ -0,0646 & 0,0194 & -0,1128 & 99,9999 \end{bmatrix},$$

Im letzten Spaltenvektor von  $\mathbf{V}_{4 \times 4}^T$  befinden sich die Koeffizienten  $a, b, c, d$  der Approximation mit dem geringsten Fehler für die Lösung des überbestimmten Gleichungssystems. Diese Koeffizienten sollten noch normiert werden, indem der Spaltenvektor durch die Länge  $l$  seiner ersten drei Komponenten geteilt wird<sup>6</sup>.

$$l = \sqrt{(-0,000039)^2 + 0,000822^2 + (-0,001025)^2} = 0,001314$$

$$\text{Normierung der Koeffizienten: } \Rightarrow \begin{bmatrix} a \\ b \\ c \\ d \end{bmatrix} = \frac{1}{l} \cdot \begin{bmatrix} -0,000039 \\ 0,000822 \\ -0,001025 \\ 0,999999 \end{bmatrix} = \begin{bmatrix} -0,029737 \\ 0,625344 \\ -0,779940 \\ 761,0344 \end{bmatrix}$$

Mit Hilfe der Singulärwertzerlegung wurde folgende Ebenengleichung aufgestellt:

$$E : -0,029737 \cdot x + 0,625344 \cdot y - 0,779940 \cdot z + 761,0344 = 0$$

In Abbildung 22 sind die zehn Punkte und die mit der Singulärwertzerlegung berechnete Ebene eingezeichnet. Man kann gut erkennen, dass die berechnete Ebene eine gute Repräsentation der Punkte ist. Im nächsten Abschnitt werden einige Ausreißertest vorgestellt und überprüft, ob sie zur Verbesserung der Qualität der Ebene beitragen können.

---

<sup>6</sup>Durch die Normierung erhält man die Abstandsinformationen, beim Einsetzen der Punkte in die Ebenengleichung, in dem Längenmaß des verwendeten Koordinatensystems.

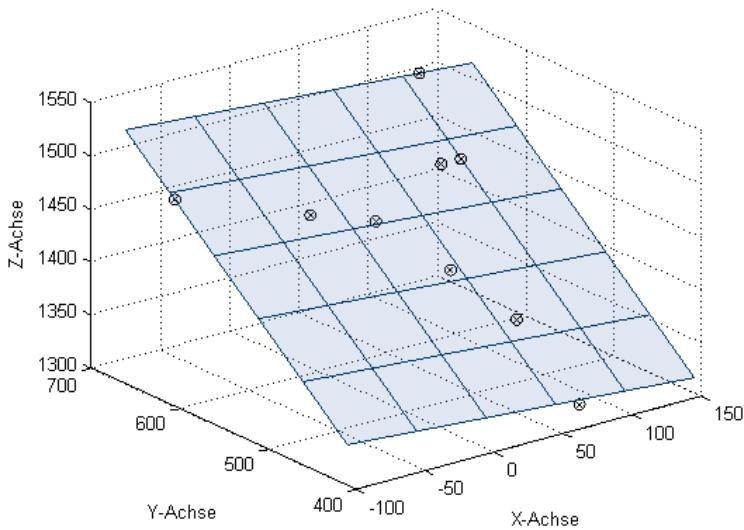


Abbildung 22: Die passende Ebene zu den zehn gegebenen Punkten.

## 4.4 Ausreißerbehandlung

In diesem Abschnitt werden statistische Testverfahren vorgestellt, die genutzt werden, um zu überprüfen, ob ein Wert zu der Stichprobe passt. Es soll eine Möglichkeit gefunden werden, Positionsdaten, die sehr stark von den anderen abweichen, zu erkennen und aus der Ebenenberechnung zu entfernen.

In der Statistik bezeichnet ein Ausreißer einen Messwert, der nicht in die erwartete Messreihe passt. Grundlage der Ausreißertests sind die Abstände der einzelnen Positionsdaten zur berechneten Ebene. Diese Abstände können in beide Richtungen auftreten und sollten betragsmäßig möglichst gering sein. Hohe Abstände sollen als Ausreißer erkannt werden. Ausreißer sind, in diesem Sinne, Positionen, die aufgrund von Messungenauigkeiten weiter von ihrer tatsächlichen Position entfernt gesichtet wurden oder auch Fehldetektionen von Mustern, die eigentlich gar nicht vorhanden sind.

Nach den Voraussetzungen aus Kapitel 3 kann man erwarten, dass alle korrekt erkannten Positionsdaten innerhalb einer Ebene liegen sollten. Dementsprechend sollte der Abstand der Punkte zu der Ebene, die mittels Singulärwertzerlegung berechnet wurde, entsprechend gering sein. Da bei der Berechnung der Suchmusterebene zunächst alle korrekt erkannten Punkte genutzt werden, werden auch Ausreißer, die eventuell in der Stichprobe vorhandene sind mit in die Ausgleichsrechnung einbezogen.

Das Vorhandensein von Ausreißern führt zum sogenannten Hebeleffekt, der in Abbildung 23 veranschaulicht wird. Dort sind einige Punkte abgebildet, die durch eine Gerade approximiert werden sollen. Durch den einen Messwert der rechts unten liegt (Ausreißer), wird das Ergebnis sehr stark verfälscht. Normalerweise sollte die Regressionsgerade einen

#### 4 Rekonstruktion der Tiefeninformation

Anstieg von etwa 1 haben. Tatsächlich besitzt die berechnete Gerade den Anstieg  $\frac{1}{2}$  und ist damit nicht repräsentativ für die Messreihe. Dieser Effekt tritt im dreidimensionalen

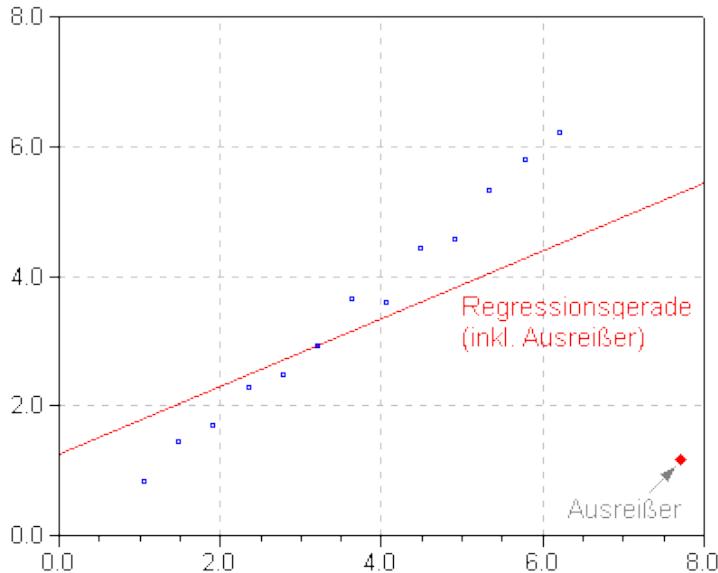


Abbildung 23: Der unerwartete Anstieg der Regressionsgeraden zeigt den Hebeleffekt von Ausreißern.

Fall ebenfalls auf und kann die Lage der Ebene verfälschen.

Um eine möglichst gute Repräsentation der Suchmusterebene zu erhalten, müssen die eventuell vorhandenen Ausreißer gefunden und aus der Berechnung der Ebene entfernt werden. Im Folgenden wird eine Übersicht über Ausreißertests in der Statistik gegeben. Mit Hilfe dieser Tests kann eine Aussage gemacht werden, mit welcher Wahrscheinlichkeit es sich um einen Ausreißer handelt.

### Ausreißertest nach Grubbs

Dieser Test setzt voraus, dass die Grundgesamtheit der Stichprobe annähernd normalverteilt ist. Der Test überprüft, ob der kleinste ( $x_1$ ) oder der größte Wert ( $x_n$ ) der Stichprobe (vom Umfang  $N$ ) ein Ausreißer ist. Dabei wird die Distanz zwischen dem kleinsten oder größten Wert und dem Mittelwert ( $\bar{x}$ ) der Stichprobe ins Verhältnis zur Standardabweichung ( $\sigma$ ) gesetzt. Ist der Abstand im Verhältnis zur Streuung um den Mittelwert zu groß, dann wird der untersuchte Wert als Ausreißer klassifiziert.

Man berechnet zunächst eine Prüfgröße  $P_i$  mit  $i \in \{1, n\}$  wobei  $P_i$  entweder den kleinsten ( $P_1$ ) oder den größten Wert ( $P_n$ ) der Stichprobe prüft. Eine Sortierung der Stichprobe

ist dafür nicht erforderlich.

$$P_i = \left| \frac{\bar{x} - x_i}{\sigma} \right|, \quad \text{mit } \bar{x} = \frac{1}{N} \sum_{i=1}^N x_i \quad \text{und} \quad \sigma = \sqrt{\frac{1}{N-1} \sum_{i=1}^N (x_i - \bar{x})^2} \quad (23)$$

Die ermittelte(n) Prüfgröße(n)  $P_1$  bzw.  $P_n$  werden schließlich mit einem Wert  $P_{N,\alpha}$  aus einer Prüfgrößen-Tabelle verglichen. In dieser Prüfgrößen-Tabelle stehen obere Grenzen, die mit einer bestimmten Wahrscheinlichkeit gelten. Je nach Stichprobenumfang  $N$  und benötigter Sicherheit  $(1 - \alpha)$  existieren verschiedene kritische Werte. Werden diese von  $P_1$  bzw.  $P_n$  überschritten, handelt es sich bei dem geprüften Wert  $x_i$  mit einer Wahrscheinlichkeit von  $1 - \alpha$  um einen Ausreißer.

Hat man einen Ausreißer klassifiziert, eliminiert man ihn aus der Stichprobe und wiederholt den Vorgang bis sich keine weiteren Ausreißer finden lassen. Dabei werden das arithmetische Mittel und die Standardabweichung jedes Mal neu berechnet.

Eine Erweiterung des Testverfahrens von *Grubbs* ist der Ausreißertest nach *Nalimov*. Beim *Nalimov* Test wird die Prüfgröße durch einen, vom Stichprobenumfang  $N$  abhängigen Korrekturfaktor ( $\sqrt{\frac{N}{N-1}}$ ), korrigiert.

$$P_i = \sqrt{\frac{N}{N-1}} \cdot \left| \frac{\bar{x} - x_i}{\sigma} \right| \quad (24)$$

Dadurch ist der Test bei kleinen Stichprobenumfängen ( $N < 30$ ) etwas genauer als der Ausreißertest nach *Grubbs*. Bei großen Stichprobenumfängen hat der Korrekturfaktor jedoch keinen großen Einfluss mehr.

## Ausreißertest nach Dixon

Mit dem Ausreißertest nach *Dixon* kann man überprüfen, ob der kleinste ( $x_1$ ) oder größte Wert ( $x_n$ ) einer Stichprobe ein Ausreißer ist. Voraussetzung für den Test ist eine annähernd normalverteilte Grundgesamtheit. Für den Test werden die Abstände der Punkte zur Ebene der Größe nach sortiert (aufsteigend, um den kleinsten Wert zu testen oder absteigend um den größten Wert zu testen).

Von diesem Test gibt es mehrere Variationen, die es erlauben, auf die unterschiedlichen Ausprägungen der Stichprobe zu reagieren. Wenn beispielsweise mehrere Ausreißer vermutet werden, kann man diese von der Berechnung ausschließen. Damit ist es selbst dann möglich, Ausreißer als solche zu erkennen, wenn mehrere in etwa gleichgroße Ausreißer vorliegen, die die Streuung um den Mittelwert an sich heranziehen. Gibt es lediglich einen verdächtigen Wert nutzt man die folgende Formel:

$$P_1 = \frac{x_2 - x_1}{x_n - x_1}, \quad P_n = \frac{x_n - x_{(n-1)}}{x_n - x_1} \quad (25)$$

Mit  $P_1$  wird die Prüfgröße bezeichnet, die zum Testen des kleinsten Wertes genutzt wird,  $P_n$  bezeichnet entsprechend die Prüfgröße zum Testen des größten Wertes. Bei mehreren

verdächtigen Werten in der Stichprobe sollten die anderen verdächtigen Werte beim Test berücksichtigt werden, indem sie übersprungen werden:

$$P_1 = \frac{x_{(1+k)} - x_1}{x_{(n-g)} - x_1}, \quad P_n = \frac{x_n - x_{(n-g)}}{x_n - x_{(1+k)}}. \quad (26)$$

$x_1$  und  $x_n$  sind wie zuvor Minimum und Maximum der Stichprobe,  $k$  und  $g$  bezeichnen die Anzahl der kleinsten bzw. größten Werte, die als Ausreißer verdächtigt werden.

Die ermittelten Prüfgrößen werden, wie beim *Grubbs Test*, mit einem Wert aus einer Prüfgrößen-Tabelle verglichen. Die Prüfgrößen-Tabelle ist ähnlich aufgebaut, wie die Tabelle zum *Grubbs Test*. Man sucht den zum Stichprobenumfang  $N$  und der gewünschten Sicherheit  $(1 - \alpha)$  gehörenden kritischen Wert heraus und vergleicht ihn mit den ermittelten Prüfgrößen. Ist die berechnete Prüfgröße größer als der kritische Wert, dann wird der betrachtete Wert als Ausreißer klassifiziert und sollte aus der Stichprobe entfernt werden.

## Box and Whisker Plot (Box-Plot)

Beim Box and Whisker Plot handelt es sich nicht direkt um einen Ausreißertest, sondern um eine Visualisierung der Stichprobe. Ein Box and Whisker Plot, wie er in Abbildung 24 zu sehen ist, erhält man, indem man die sogenannten Quartile der Stichprobe bestimmt und das erste und dritte Quartil einzeichnet. In der Literatur definierten sich die Quartile nach Tukey wie folgt:

$$\begin{aligned} Q_1 &= \frac{N+3}{4}, \quad Q_3 = \frac{3 \cdot N + 1}{4}, & \text{wenn } N \text{ ungerade ist} \\ Q_1 &= \frac{N+2}{4}, \quad Q_3 = \frac{3 \cdot N + 2}{4}, & \text{wenn } N \text{ gerade ist} \end{aligned} \quad (27)$$

Bei der Berechnung wird jeweils auf die nächste ganze Zahl aufgerundet. Jedes Quartil beinhaltet 25% des Stichprobenumfangs. Das erste und dritte Quartil wird durch einen waagerechten Strich gekennzeichnet und miteinander verbunden, so dass man eine Box erhält. Die Breite der Box spielt keine Rolle, die Höhe wird als Interquartilsabstand bezeichnet. Nach der Definition der Quartile beinhaltet die Box 50% der Stichprobe. Als weitere Visualisierung wird der Median ( $M$ ) als waagerechter Strich in die Box eingezeichnet.

Die Whisker, das sind die senkrecht zu den Quartilsgrenzen verlaufenden Geraden, die ebenfalls mit einer waagerechten Linie abgeschlossen werden, stellen eine Art Abgrenzung zu eventuell vorhandenen Ausreißern dar. Ihre maximale Länge berechnet sich aus dem Interquartilsabstand und beträgt  $(Q_3 - Q_1) \cdot 1,5$ . Die waagerechten Begrenzungslinien kennzeichnen allerdings den letzten Wert, der noch innerhalb dieser maximalen Whiskerlänge liegt. Aus diesem Grund kann es vorkommen, dass die Whisker eine unterschiedliche Länge besitzen. Alle Werte, die außerhalb der Whisker-Begrenzung liegen, werden als Ausreißer betrachtet.

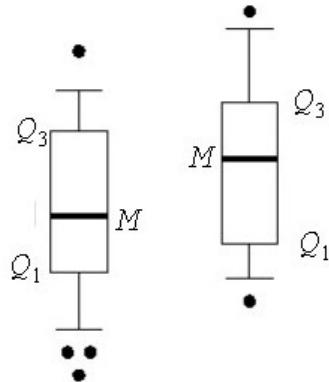


Abbildung 24: Box and Whisker Plot einer Stichprobe.  $Q_1$  ist das erste Quartil,  $Q_3$  das dritte Quartil und  $M$  der Median der Stichprobe. Punkte die ober- oder unterhalb der waagerechten Striche (Whisker) liegen, stellen potentielle Ausreißer dar.

## Der RANSAC Algorithmus

Der *RANSAC* (Random Sample Consensus) Algorithmus ist kein typischer Ausreißertest, da er nicht die Ausreißer innerhalb einer Stichprobe klassifiziert. Vielmehr nominiert er Beobachtungen, die ein bestimmtes Modell erfüllen. Werte, die nicht das Gütekriterium erfüllen und entsprechend nicht zum Modell passen, werden nicht ausgewählt. So werden auch Werte ausgesondert, die herkömmliche Ausreißertest als „noch zur Stichprobe gehörig“ befunden hätten.

Angewandt wird er bei größeren Stichproben, die mit hoher Wahrscheinlichkeit Ausreißer und Messfehler enthalten. Der Algorithmus wählt zunächst zufällig Punkte und benutzt diese, um die Parameter für ein vorgegebenes Modell zu berechnen (z.B. Koeffizienten, die eine Ebene beschreiben). Anschließend wird mit allen Punkten in einer Fehlerfunktion überprüft, wie gut diese zu dem Modell passen. Dabei werden alle Punkte, die ein bestimmtes Gütekriterium erfüllen, gespeichert. Je mehr Punkte das Gütekriterium erfüllen, desto besser repräsentieren die berechneten Modellparameter die anderen Punkte. Anschließend wird der Vorgang wiederholt und es wird überprüft, ob die neu gewählten Modellparameter besser zu allen Punkten passen, als die Vorherigen. Das ist dann der Fall, wenn mehr Punkte das Gütekriterium erfüllen. So wird eine Menge an *zuverlässigen Punkten* aufgebaut. Punkte, die nicht zu dieser Menge gehören, sind Abweichungen vom Modell und sollen im nächsten Schritt nicht berücksichtigt werden. Hat man die festgelegte Anzahl an Wiederholungen durchlaufen oder wurde eine festgelegte Anzahl an *zuverlässigen Punkten* überschritten, werden mit Hilfe einer Ausgleichsrechnung die Modellparameter berechnet, die am Besten zu den *zuverlässigen Punkten* passen.

Die wiederholte zufällige Punktauswahl und Berechnung der Modellparameter sind nötig, um sicher zu gehen, dass unter den zufällig gewählten Punkten mit einer hohen Wahrscheinlichkeit mindestens einmal *gute* Punkte ausgewählt werden. Dabei sind *gute* Punkte solche, die in, oder sehr dicht an, der tatsächlichen Suchmusterebene liegen. *Schlechte* Punkte bezeichnen Positionen, die aufgrund von Messungenauigkeiten weit von der eigentlichen Suchmusterebene entfernt liegen.

### Die Anzahl der notwendigen Wiederholungen ( $k$ )

Die Anzahl der notwendigen Wiederholungen ( $k$ ) um mit hoher Wahrscheinlichkeit mindestens einmal keinen schlechten Punkt, bei der zufälligen Auswahl, zu wählen, kann man, mit Hilfe der Wahrscheinlichkeitsrechnung, folgendermaßen abschätzen:

1. Zunächst überlegt man, wie der Anteil an *guten* Punkten ( $w$ ) im Datensatz ist. Wenn über die Verteilung nichts bekannt ist, ist  $w = 0,5$  ein üblicher Wert.
2. Nun benötigt man die Anzahl ( $n$ ) der Punkte, die zufällig ausgewählt werden, um die Modellparameter zu berechnen. In diesem Fall wäre  $n = 3$ . Dementsprechend beschreibt  $w^n$  die Wahrscheinlichkeit, dass alle  $n$  gewählten Punkte *gut* sind und  $1 - w^n$  ist die Wahrscheinlichkeit, dass mindestens einmal ein *schlechter* Punkt ausgewählt wurde.
3. Die Wahrscheinlichkeit, dass man bei  $k$  Wiederholungen immer mindestens einen *schlechten* Punkt mit auswählt beträgt  $(1 - w^n)^k$ . Möchte man mit einer bestimmten Wahrscheinlichkeit  $(1 - p)$  erreichen, dass in keinem Durchgang  $n$  *gute* Punkte ausgewählt werden, gilt:  $1 - p = (1 - w^n)^k$ .
4. In der Regel möchte man diese Wahrscheinlichkeit sehr gering halten. Also wählt man  $p$  (die Sicherheit) so groß, wie man es benötigt (z. B.  $p = 0,99$ ) und bestimmt die nötige Anzahl an Wiederholungen ( $k$ ).
5. Nach Umstellen der Formel aus 3. ergibt sich:

$$k = \frac{\log(1 - p)}{\log((1 - w^n)^k)} \quad (28)$$

$k$  ist also eine obere Schranke an Wiederholungen um mit einer bestimmten Wahrscheinlichkeit  $(1 - p)$  in keinem Durchgang  $n$  *gute* Punkte auszuwählen.

An einem Beispiel soll die Abschätzung der nötigen Wiederholungen verdeutlicht werden. Es werden drei Punkte zufällig ausgewählt und eine Sicherheit von  $p = 0,99$  angestrebt. Dann sind bei einem Datensatz, mit einer unbekannten Menge an Ausreißern ( $0,5$  wird angenommen),  $k = \log(0,01)/\log(1 - (0,5)^3) = 34,4875$ , also 35 Wiederholungen des Auswahlverfahrens, nötig.

Vom *RANSAC* Algorithmus selber existieren verschiedene Variationen. In der Arbeit wurden zwei Varianten getestet. Sie unterscheiden sich in der Anzahl der Punkte, die zufällig aus der Stichprobe ausgewählt wurden. Die erste Variante wählt genau drei Punkte, die andere wählt 20% der gesamten Beobachtungen aus. Anschließend wird aus den Punkten jeweils eine Ebene berechnet, wobei die erste Variante eine eindeutige Lösung hat. Bei der zweiten Variante wird eine Näherungslösung über die Singulärwertzerlegung bestimmt.

## Auswahl eines geeigneten Tests

Von den vorgestellten Ausreißertests eignete sich der *RANSAC* Algorithmus am Besten für die Bereinigung des Datensatzes von Ausreißern. Der Vorteil dieses Algorithmus besteht darin, dass er nur Punkte auswählt, die zu dem gewählten Modell passen. Durch die eigene Definition einer Fehlerfunktion kann man beeinflussen, wie stark die Abweichungen der Punkte sein dürfen. Dabei muss darauf geachtet werden, dass man nicht zu strenge Regeln formuliert, die nur von wenigen Punkten eingehalten werden können. Denn dann ist das Modell nicht repräsentativ für den Datensatz. Es ist ratsam, die Datensätze vor der Formulierung der Fehlerfunktion zu analysieren, um herauszufinden, welche Werte erwartet werden und welche Werte auftreten können, aber nicht erwünscht sind.

Ein weiterer Vorteil ist der, dass die Verteilung der Daten keine Rolle spielt. Von Nachteil ist, dass die Punkte, die zur Bestimmung der Modellparameter genutzt werden, zufällig ausgewählt werden. Dabei besteht die Möglichkeit, dass *schlechte* Daten, die eigentlich besser aussortiert werden sollten, gewählt werden. Dieser Nachteil lässt sich durch mehrfache Wiederholungen der Modellschätzung mit hoher Wahrscheinlichkeit ausschließen.

In Abbildung 25 werden drei verschiedene Ebenen gegenüber gestellt. Während in der oberen Abbildung alle Punkte des Datensatzes für die Berechnung der Ebene genutzt wurden, wurden bei den anderen Abbildungen die Punkte für die Ebenenberechnung mittels *RANSAC* ausgewählt (Variante 1 mittlere Abbildung und Variante 2 untere Abbildung). Man kann anhand der Seitenansichten der Ebene (rechte Spalte) deutlich erkennen, dass bei der Ebenenberechnung aus allen Punkten der Hebeleffekt auftritt und damit die Mehrzahl der Punkte eine größere Abweichung zur Ebene besitzen. Die Ebenen, die aus den von *RANSAC* ermittelten *zuverlässigen Punkten* berechnet wurden, unterscheiden sich nur geringfügig. Da die Punkte zufällig ausgewählt werden, ist mal die eine Variante und mal die andere Variante besser<sup>7</sup>. Bei einem Vergleich der beiden Varianten anhand von zwölf Datensätzen zeigte sich kein signifikanter Unterschied. Aufgrund des erhöhten Rechenaufwands bei Variante 2 und der Möglichkeit bei Variante 1 die nötige Anzahl

---

<sup>7</sup>Aussagen zur Qualität lassen sich über die mittleren Abstände der Punkte zur Ebene machen.

#### 4 Rekonstruktion der Tiefeninformation

an Wiederholungen in *RANSAC* abzuschätzen, wurde Variante 1 als Ausreißertest ausgewählt.

Der Box and Whisker Test eignet sich bei den vorliegenden Datensätzen nicht, da die Varianz innerhalb der Datensätze so groß ist, dass durch die Whiskerbegrenzung nicht die gewünschte Menge an Ausreißern klassifiziert werden konnte.

Der Ausreißertest nach *Grubbs* lässt sich in diesem Fall nicht sinnvoll anwenden, weil der Rechenaufwand vergleichsweise hoch ist. Denn je Durchlauf kann nur der kleinste und der größte Wert getestet werden. Um weitere Werte zu testen, muss man den Test erneut durchführen. Zuvor muss die Ebene erneut berechnet werden, um zu überprüfen, ob sich die Lage der Ebene bereits verändert hat. Eine weitere Einschränkung des Test ist die Voraussetzung einer annähernd normalverteilten Stichprobe.

Der Ausreißertest nach *Dixon* lässt sich ebenfalls nicht nutzen, um die Güte der Positionsdaten zu überprüfen. Denn um den Test erfolgreich einzusetzen, ist es nötig, die Anzahl der potentiellen Ausreißer zu kennen. Ansonsten kann es vorkommen, dass ein Ausreißer nicht erkannt wird, weil er mit einem anderen Ausreißer verglichen wird. Dieser andere Ausreißer wird jedoch übersprungen, wenn er ebenfalls als Ausreißer verdächtigt wird. Die Auswahl von Ausreißerkandidaten müsste vor der Anwendung des *Dixon Tests* geschehen. Ein weiterer Nachteil ist, das der Test ebenfalls eine normalverteilte Grundgesamtheit erfordert, von der nicht immer ausgegangen werden kann.

#### 4 Rekonstruktion der Tiefeninformation

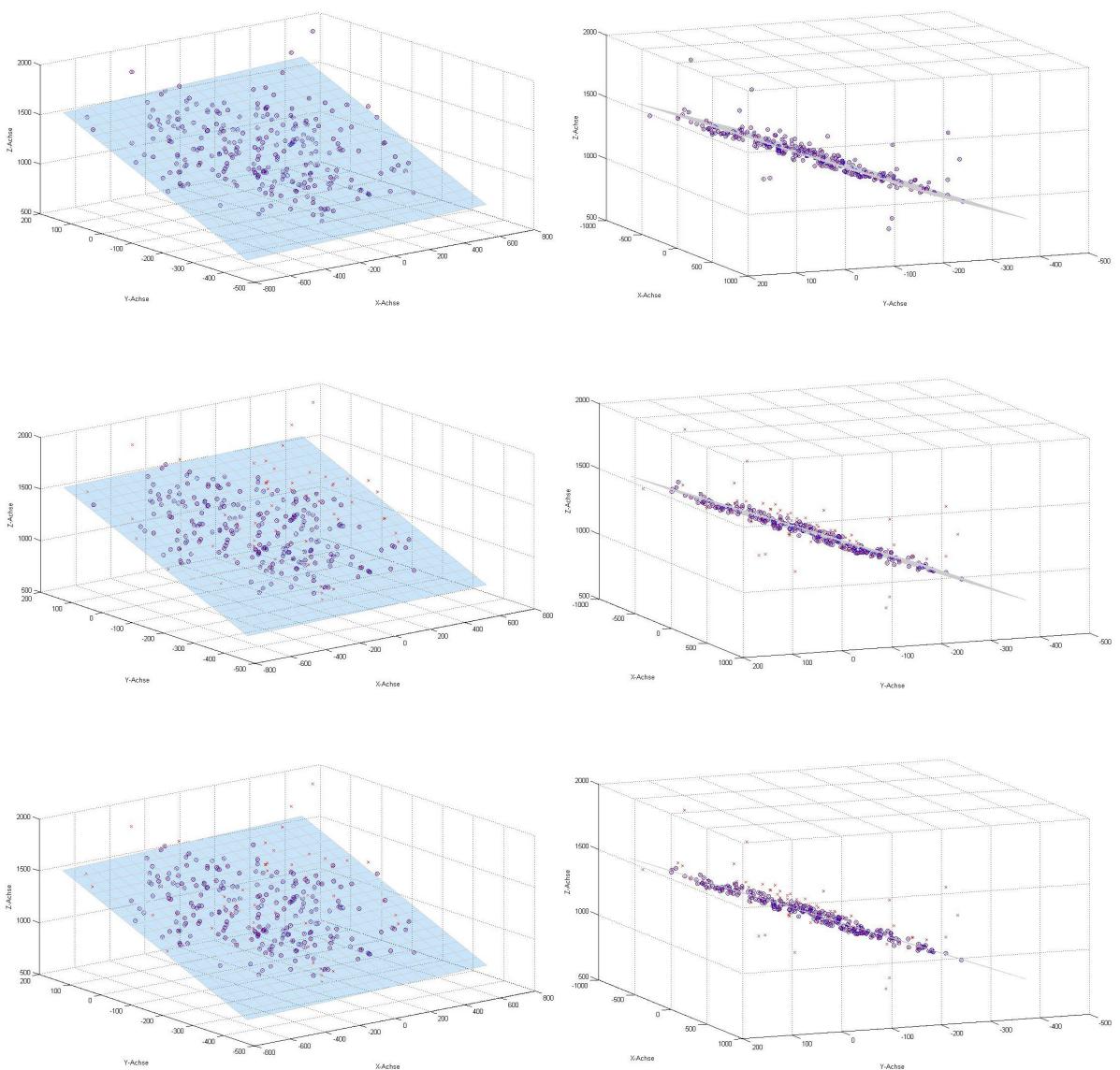


Abbildung 25: Die dargestellte Ebene wurde aus den blauen Punkten berechnet. Die Auswahl der Punkte erfolgte mittels *RANSAC* Variante 1 (mitte) und *RANSAC* Variante 2 (unten). Die Seitenansicht (rechts) verdeutlicht jeweils die Abstände zwischen den Punkten und der Ebene.

# 5 Implementierung

In diesem Kapitel wird die praktische Umsetzung der Korrektur von fehlerhaften Positionsdaten beschrieben. Dafür sind folgende Funktionalitäten nötig:

- Die Berechnung der Bewegungsebene, als Grundlage für die Korrektur.
- Die Rückprojektion von 2D-Bildkoordinaten in 3D-Weltkoordinaten.
- Die Projektion von 3D-Koordinaten in 2D-Bildkoordinaten.
- Die Korrektur der fehlerhaften Positionsdaten.
- Die Verschiebung der Ebene.

Implementiert und getestet wurde alles in Matlab Version 7.1. Grundlage der Berechnungen sind Positionsdaten, die sich entweder als Datei im Arbeitsverzeichnis oder bereits als Variablen im Workspace von Matlab befinden. Im Folgenden werden die einzelnen Funktionen genauer beschrieben.

## 5.1 Berechnung der Bewegungsebene der Objekte

Um die Bewegungsebene zu berechnen werden folgende Daten benötigt:

- Die 3D-Kamerakoordinaten der gesichteten Ankerpunkte,  $\mathbf{X}$ .
- Und der binäre Zustandsvektor  $\omega_i$ , der zu jedem Punkt die Gültigkeitsinformation enthält.

$$\text{Wobei: } \mathbf{X} = \begin{bmatrix} x_1 & y_1 & z_1 \\ x_2 & y_2 & z_2 \\ \vdots & & \\ x_N & y_N & z_N \end{bmatrix} \quad \text{und} \quad \omega_i = \begin{bmatrix} \omega_1 \\ \omega_2 \\ \vdots \\ \omega_N \end{bmatrix} \quad \omega_i \in \{\text{gültig, ungültig}\} \quad (29)$$

Allgemein wird folgender Algorithmus angewandt:

1. Aus den Eingabewerten wird eine erweiterte Designmatrix  $\mathbf{A}$  aufgebaut.
2. Nach dem *RANSAC* Algorithmus, Variante 1, werden aus  $\mathbf{A}$  die *zuverlässigen* Punkten ausgewählt.

- a) Zuerst zufällig drei Punkte aus  $\mathbf{A}$  auswählen.
  - b) Diese Punkte auf Kollinearität überprüfen, falls positiv, drei neue Punkte wählen.
  - c) Die Ebene berechnen, die von den drei Punkten aufgespannt wird.
  - d) Die Fehlerfunktion anwenden.
  - e) Die Anzahl der *zuverlässigen* Punkte mit der bisherigen, besten Lösung vergleichen. Wenn es mehr sind als bisher, dann ersetzt die aktuelle Lösung die bisher beste.
  - f) Schritte 2.a) bis 2.e)  $k$  mal wiederholen.
3. Die *zuverlässigen* Punkte werden für die Berechnung der Bewegungsebene benutzt.
  4. Die Koeffizienten der Bewegungsebene werden ausgegeben.

Nun werden einige Schritte genauer erläutert.

**Zu 1.:** Die Designmatrix  $\mathbf{A}$  ist folgendermaßen aufgebaut:

$$\mathbf{A} = \begin{bmatrix} x_1 & y_1 & z_1 & 1 & \omega_1 \\ x_2 & y_2 & z_2 & 1 & \omega_2 \\ \vdots \\ x_N & y_N & z_N & 1 & \omega_N \end{bmatrix}. \quad (30)$$

In den ersten drei Spalten sind die Koordinaten von  $\mathbf{X}$  enthalten. Die vierte Spalte erweitert die Koordinaten auf homogene Koordinaten. Das ist nötig, damit später die Punkte ( $\tilde{\mathbf{A}}_{N \times 4}$ ) in die Ebenengleichung in Koordinatenform eingesetzt werden können. In Spalte fünf sind die Zustandsinformationen zu den jeweiligen Punkten enthalten.

**Zu 2.b:** Drei Punkte  $P_1, P_2, P_3$  sind nicht kollinear, wenn das daraus resultierende Gleichungssystem  $G$  eine eindeutige, nicht triviale Lösung besitzt. Das ist genau dann der Fall, wenn die dazugehörige Determinante Null ist.

$$G = \left[ a \cdot \begin{pmatrix} p_{11} \\ p_{12} \\ p_{13} \end{pmatrix} + b \cdot \begin{pmatrix} p_{21} \\ p_{22} \\ p_{23} \end{pmatrix} + c \cdot \begin{pmatrix} p_{31} \\ p_{32} \\ p_{33} \end{pmatrix} \right] = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \Rightarrow \det G = 0 \quad (31)$$

**Zu 2.c:** Am Einfachsten ist es, zunächst die Ebenengleichung in Parameterform aufzustellen und sie dann in die Koordinatenform zu überführen. Dazu wählt man einen Punkt  $P_1$  als Stützpunkt und bestimmt die beiden Richtungsvektoren aus  $P_2 - P_1$  bzw.  $P_3 - P_1$ . Das Kreuzprodukt beider Richtungsvektoren stellt den Normalenvektor dar. Dieser wird anschließend normiert. Somit erhält man die Koeffizienten  $a, b$  und  $c$  der Ebenengleichung in Koordinatenform. Den Koeffizienten  $d$  erhält man über das Skalarprodukt von  $P_1$  und

## 5 Implementierung

dem Normalenvektor. Nun sind alle Koeffizienten der Ebenengleichung in Koordinatenform bestimmt:  $E : a \cdot x + b \cdot y + c \cdot z = d$ . Da die Darstellungsform  $E : a \cdot x + b \cdot y + c \cdot z - d = 0$  für die weiteren Rechenoperationen besser geeignet ist, muss das Vorzeichen von  $d$  umgekehrt werden.

**Zu 2.d:** Für die Fehlerfunktion werden die Abstände  $\delta_i$  der Punkte zu der berechneten Ebene benötigt. Diese erhält man, indem man die homogenen Koordinaten ( $\mathbf{A}_{N \times 4}$ ) mit den Koeffizienten  $e = (a, b, c, d)^T$  der Koordinatenform multipliziert:  $\mathbf{A}_{N \times 4} \cdot e_{4 \times 1} = \delta_{N \times 1}$ . Ziel ist es nun, die in 2.c bestimmte Ebene zu bewerten. Je besser diese Ebene alle gültigen Punkte repräsentiert, desto mehr Punkte sollten von der Fehlerfunktion als *zuverlässig* klassifiziert werden. Dafür werden, in Anlehnung an den *Box and Whisker Plot*, zunächst alle Abstände aufsteigend sortiert und die Quartile bestimmt. Im Unterschied zu der Quartilsdefinition von Tukey bezeichnet  $Q_2$  den Median und  $Q_4$  den größten Wert. Es gilt:

$$\begin{aligned}\Delta^{Q_1} &= \{\delta_i \mid 1 \leq i \leq Q_1\} & \Delta^{Q_2} &= \{\delta_i \mid Q_1 < i \leq Q_2\} \\ \Delta^{Q_3} &= \{\delta_i \mid Q_2 < i \leq Q_3\} & \Delta^{Q_4} &= \{\delta_i \mid Q_3 < i \leq Q_4\}\end{aligned}\quad (32)$$

$\Delta^{Q_1}$  enthält die größten Abstände in negativer Richtung.  $\Delta^{Q_4}$  beinhaltet die größten Abstände in positiver Richtung. Die Abstände in  $\Delta^{Q_2}$  und  $\Delta^{Q_3}$  sind betragsmäßig kleiner. Abbildung 26 soll das Prinzip veranschaulichen.

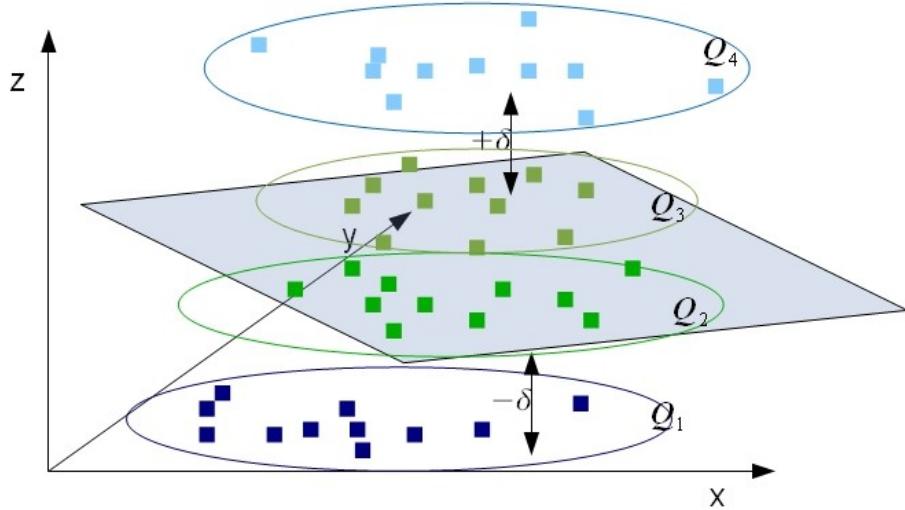


Abbildung 26: Je nach Abstand der Punkte zur Ebene werden die Punkte in die unterschiedlichen Quartile eingegordnet.

Nun werden die Abstände aus  $\Delta^{Q_1}$  und  $\Delta^{Q_4}$  sowie die Abstände aus  $\Delta^{Q_2}$  und  $\Delta^{Q_3}$  zusammengefasst:

$$\Delta^+ = \{|\delta| \mid \delta \in \Delta^{Q_1} \cup \Delta^{Q_4}\} \quad \Delta^- = \{|\delta| \mid \delta \in \Delta^{Q_2} \cup \Delta^{Q_3}\}. \quad (33)$$

## 5 Implementierung

In  $\Delta^+$  sind die absoluten Abstände aus  $\Delta^{Q_1}$  und  $\Delta^{Q_4}$  enthalten,  $\Delta^-$  beinhaltet die absoluten Abstände aus  $\Delta^{Q_2}$  und  $\Delta^{Q_3}$ . Außerdem gilt:

$$\begin{aligned}\overline{\delta^+} &= \frac{1}{N} \cdot \sum_{i=1}^N \delta_i^+, \quad \delta_i^+ \in \Delta^+, \\ \overline{\delta^-} &= \frac{1}{M} \cdot \sum_{i=1}^M \delta_i^-, \quad \delta_i^- \in \Delta^-. \end{aligned}\tag{34}$$

Mit  $\overline{\delta^+}$  bzw.  $\overline{\delta^-}$  werden die mittleren Abstände aus  $\Delta^+$  und  $\Delta^-$  bezeichnet. Dabei sollte  $\overline{\delta^-} < \overline{\delta^+}$  sein. Neben den beiden mittleren Abständen werden noch zwei weitere Werte  $\epsilon_1$  und  $\epsilon_2$  für die Fehlerfunktion benötigt. Beides sind selbst festzulegende Grenzwerte<sup>8</sup>. Mit  $\epsilon_1$  wird entschieden, ob der Abstand eines Punktes zur Ebene noch zulässig ist oder nicht. Mit  $\epsilon_2$  kann gesteuert werden, welches Verhältnis von  $\overline{\delta^+}$  zu  $\overline{\delta^-}$  für *gut* befunden wird. Für die *zuverlässigen* Punkte  $\mathbf{Z}$ , die mit Hilfe der Fehlerfunktion ausgewählt werden, ergibt sich folgendes:

$$\mathbf{Z} = \begin{cases} \{X_i | \delta_i \leq \overline{\delta^-}\}, & \text{wenn } \overline{\delta^-} > \epsilon_1 \\ \{X_i | \delta_i \leq \overline{\delta^-} + \overline{\delta^+}\}, & \text{wenn } \overline{\delta^-} \leq \frac{\epsilon_1}{2} \text{ und } \frac{\overline{\delta^+}}{\overline{\delta^-}} \leq \epsilon_2 \\ \{X_i | \delta_i \leq \overline{\delta^+}\}, & \text{wenn } \overline{\delta^-} \leq \frac{\epsilon_1}{2} \text{ und } \frac{\overline{\delta^+}}{\overline{\delta^-}} > \epsilon_2 \\ \{X_i | \delta_i \leq \overline{\delta^+}\}, & \text{wenn } \frac{\epsilon_1}{2} < \overline{\delta^-} \leq \epsilon_1 \text{ und } \frac{\overline{\delta^+}}{\overline{\delta^-}} \leq \epsilon_2 \\ \{X_i | \delta_i \leq \overline{\delta^-}\}, & \text{wenn } \frac{\epsilon_1}{2} < \overline{\delta^-} \leq \epsilon_1 \text{ und } \frac{\overline{\delta^+}}{\overline{\delta^-}} > \epsilon_2 \end{cases} \tag{35}$$

wobei:  $\delta_i \in \Delta^+ \cup \Delta^-$ ,  $X_i \in \mathbf{X}$ ,  $i = 1 \dots N$ ,  $N = |\Delta^+ \cup \Delta^-|$

$\mathbf{Z}$  enthält schließlich diejenigen Punkte  $X_i$ , deren Abstände  $\delta_i$  die jeweilige Bedingung erfüllen.

**Zu 3.:** Die homogenen Koordinaten  $\tilde{\mathbf{Z}}_{\mathbf{N} \times 4}$  der in 2. ausgewählten *zuverlässigen* Punkte  $\mathbf{Z}$  werden nun zu einer Ausgleichsrechnung herangezogen. Zunächst wird aus ihnen die Matrix  $\mathbf{B}$  erzeugt:

$\mathbf{B} = \tilde{\mathbf{Z}}^T \cdot \tilde{\mathbf{Z}}$ . Die Matrix  $\mathbf{B}$  wird nun einer Singulärwertzerlegung unterzogen. Aus der vierten Zeile der Rechtssingulärwertmatrix werden die Koeffizienten der Ebenengleichung entnommen. Diese müssen nun noch normiert werden.

In Abbildung 27 ist das Ergebnis der Ebenenberechnung zu sehen. Die roten Punkte kennzeichnen alle *gültigen* Punkte des Datensatzes, die blau markierten Punkte wurden als *zuverlässig* klassifiziert und für die Ausgleichsrechnung genutzt.

---

<sup>8</sup>Bei den vorliegenden Datensätzen haben sich  $\epsilon_1 = 10$  und  $\epsilon_2 = 2$  als gute Grenzwerte erwiesen.

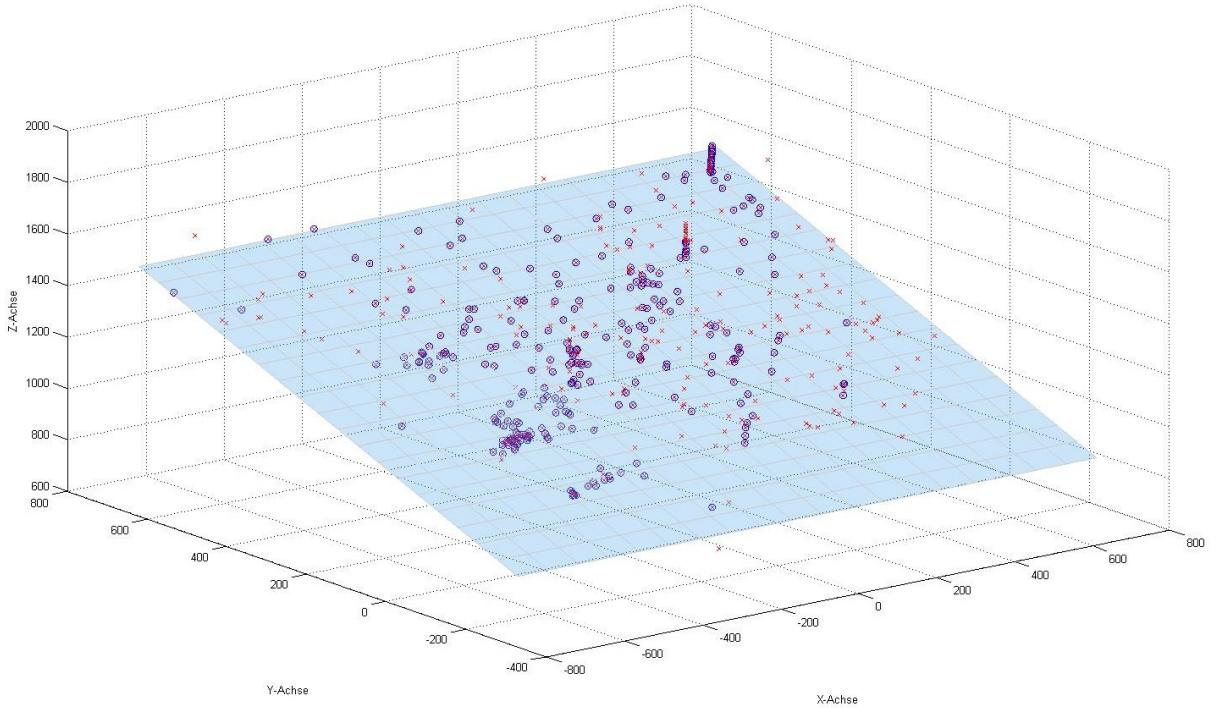


Abbildung 27: Aus den blau markierten Punkten wurde die Bewegungsebene der Objekte berechnet.

## 5.2 Rückprojektion in 3D-Weltkoordinaten

Diese Funktionalität ist erforderlich, damit fehlerhafte Punkte korrigiert werden können. Aus den 2D-Bildkoordinaten der Ankerpunkte werden, mit Hilfe der berechneten Bewegungsebene, die korrekten 3D-Weltkoordinaten berechnet. Dafür werden folgende Daten benötigt:

- Die 2D-Bildkoordinaten der Ankerpunkte.
- Die Koeffizienten der Bewegungsebene.
- Die intrinsischen und extrinsischen Kameraparameter.

Da die Kameraparameter für jede Kamera verschieden sind, ist es nötig, sie der Funktion zu übergeben oder für jede Kamera eine individuelle Funktion, mit ihren korrekten Kameraparametern, bereitzustellen. Das Prinzip für die Rückprojektion wurde in Abschnitt 4.1 erläutert.

1. Die Projektionsgleichung nach  $x_i$  und  $y_i$  umstellen.
2. Diese dann in die Ebenegleichung einsetzen und dadurch  $z_i$  bestimmen.
3. In die umgestellte Projektionsgleichung  $z_i$  einsetzen und  $x_i$  sowie  $y_i$  berechnen.

## 5 Implementierung

4. Die Punkte aus dem 3D-Kamerakoordinatensystem in das Weltkoordinatensystem transformieren.

**Zu 1.:** die Projektionsgleichung (Formel 4) kann in diesem Fall vereinfacht werden, da für die Rückprojektion die Verzerrung der Kameraoptik nicht nochmal korrigiert werden muss. Es ergibt sich:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \frac{f}{z_i} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} + \begin{bmatrix} c_u \\ c_v \end{bmatrix} \Rightarrow \begin{bmatrix} x_i \\ y_i \end{bmatrix} = \begin{bmatrix} u_i - c_u \\ v_i - c_v \end{bmatrix} \cdot \frac{z_i}{f}. \quad (36)$$

**Zu 2.:** für die Ebenengleichung gilt:  $E : a \cdot x + b \cdot y + c \cdot z + d = 0$ . Die Koeffizienten  $a$  bis  $d$  sind gegeben, für  $x$  und  $y$  werden die  $x_i$  und  $y_i$  aus 1. eingesetzt.

$$\begin{aligned} a \cdot (u_i - c_u) \cdot \frac{z_i}{f} + b \cdot (v_i - c_v) \cdot \frac{z_i}{f} + c \cdot z_i + d &= 0 \quad \text{nach } z_i \text{ umstellen} \\ \Rightarrow z_i &= \frac{1}{-\frac{a}{d} \cdot (u_i - c_u) \cdot \frac{1}{f} - \frac{b}{d} \cdot (v_i - c_v) \cdot \frac{1}{f} - \frac{c}{d}} \end{aligned} \quad (37)$$

**Zu 3.:** mit dem aus 2. gewonnenen  $z_i$  können nun die dazugehörigen Koordinaten  $x_i$  und  $y_i$  bestimmt werden.

$$\text{setze } z_i \text{ in Formel 36 aus 1. ein: } x_i = (u_i - c_u) \cdot \frac{z_i}{f} \text{ und } y_i = (v_i - c_v) \cdot \frac{z_i}{f} \quad (38)$$

Die Koordinaten  $x_i$ ,  $y_i$  und  $z_i$  beschreiben die Position des Ankerpunktes  $X_i^c$  bezüglich des Kamerakoordinatensystems.

**Zu 4.:** Um aus  $X_i^c$  die korrespondierende Position  $X_i^w$ , im 3D-Weltkoordinatensystem, zu erhalten, müssen sie mit der inversen Transformationsmatrix  $\mathbf{T}^{-1}$  transformiert werden.

$$\text{Es gilt: } \begin{bmatrix} x_i^w \\ y_i^w \\ z_i^w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \cdot \begin{bmatrix} x_i^c \\ y_i^c \\ z_i^c \\ 1 \end{bmatrix} \quad (39)$$

Die Weltkoordinaten  $[x_i^w, y_i^w, z_i^w]'$  kennzeichnen die korrigierten Positionen der bisher ungültigen Sichtungen.

## 5.3 Projektion der 3D-Kamera in 2D-Bildkoordinaten

Die Möglichkeit der Projektion von 3D-Koordinaten in 2D-Bildkoordinaten dient vor allem der Verifizierung der Rückprojektion. Man benötigt:

- die 3D-Kamerakoordinaten  $\mathbf{X}$  der Punkte,
- und die intrinsischen Kameraparameter der verwendeten Kamera.

## 5 Implementierung

Sollten die 3D-Koordinaten noch nicht ins Kamerakoordinatensystem transformiert worden sein, so muss dies zuerst geschehen. Die homogenen Weltkoordinaten  $\tilde{\mathbf{X}}^w$  müssen in homogene 3D-Kamerakoordinaten  $\tilde{\mathbf{X}}^c$  transformiert werden:

$$\begin{bmatrix} x_i^w \\ y_i^w \\ z_i^w \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_i^c \\ y_i^c \\ z_i^c \\ 1 \end{bmatrix} \quad (40)$$

Für die Projektion der Punkte werden nur die euklidischen Raumkoordinaten  $\mathbf{X}^c$  genutzt.

Die intrinsischen Kameraparameter unterscheiden sich in der Regel für jede Kamera. Werden also mehrere Kameras verwendet, müssen die benötigten Parameter der Funktion übergeben werden, oder für jede Kamera muss eine eigene Funktion bereit gestellt werden. Der Algorithmus für die Projektion, wurde bereits in Abschnitt 2.2 vorgestellt (Formel 1 bis 4). Umgesetzt wird er folgendermaßen:

1. Die Projektion der 3D-Kamerakoordinaten nach dem Lochkameramodell durchführen.
2. Die zugehörigen 2D-Bildkoordinaten bestimmen.

**Zu 1.:** es werden zunächst die Kamerakoordinaten projiziert, man erhält  $[\hat{u}_i, \hat{v}_i]'$ :

$$\begin{bmatrix} \hat{u}_i \\ \hat{v}_i \end{bmatrix} = \frac{f}{z_i} \cdot \begin{bmatrix} x_i \\ y_i \end{bmatrix} \quad (41)$$

Dabei bezeichnen  $x_i$ ,  $y_i$  und  $z_i$  die Koordinaten eines Punktes  $X_i^c$  aus dem 3D-Kamerakoordinatensystem,  $f$  ist die Brennweite der Kamera.

**Zu 2.:** um die zugehörigen 2D-Bildkoordinaten zu erhalten, muss nun die radiale und tangentiale Verzerrung korrigiert werden. Zusätzlich ist eine Verschiebung um die Koordinaten des Bildhauptpunktes  $[c_u, c_v]'$  nötig. Unter der Annahme, dass die Pixel quadratisch sind, gilt folgende Formel:

$$\begin{bmatrix} u_i \\ v_i \end{bmatrix} = \begin{bmatrix} (\hat{u}_i + \theta u_i^{(r)} + \theta u_i^{(t)}) \\ (\hat{v}_i + \theta v_i^{(r)} + \theta v_i^{(t)}) \end{bmatrix} + \begin{bmatrix} c_u \\ c_v \end{bmatrix} \quad (42)$$

Die Koordinaten  $[u_i, v_i]'$  stellen die zum Punkt  $X_i$  korrespondierenden Bildkoordinaten dar.

## 5.4 Korrektur der fehlerhaften Lösungen

Die Korrektur der fehlerhaften Lösung vereint die Berechnung der Suchmusterebene und die Rückprojektion miteinander. Für die Funktion werden folgende Daten benötigt:

- Die 3D-Kamerakoordinaten der gesichteten Ankerpunkte,  $\mathbf{X}^c$ .
- Der binäre Zustandsvektor  $\boldsymbol{\omega}_i$ , der zu jedem Punkt  $X_i^c$  die Gültigkeitsinformation enthält.
- Die 2D-Bildkoordinaten der Ankerpunkte.
- Die intrinsischen und extrengischen Kameraparameter.

Diese werden durch den nachstehenden Algorithmus verarbeitet.

1. Die Suchmusterebene berechnen.
2. Die fehlerhaften Positionsdaten auswählen.
3. Die zugehörigen 2D-Bildpunkte mittels Rückprojektions korrigieren.

**Zu 1.:** für die Berechnung der Suchmusterebene wird die Funktion aus Abschnitt 5.1 genutzt.

**Zu 2.:** die fehlerhaften Positionsdaten werden anhand des binären Zustandsvektors  $\boldsymbol{\omega}$  ausgewählt. Es werden diejenigen 2D-Bildkoordinaten  $[u_i, v_i]'$  ausgewählt, die zu den *ungültigen* Zuständen  $\omega_i$  gehören.

**Zu 3.:** diese Bildkoordinaten werden mit Hilfe des in Abschnitt 5.2 vorgestellten Algorithmus in die Suchmusterebene zurück projiziert und in Weltkoordinaten transformiert. Somit erhält man die korrekten Positionsdaten  $[x_i^w, y_i^w, z_i^w]'$  zu den als fehlerhaft klassifizierten Positionen  $[x_i, y_i, z_i]'$ . Optional kann man das Ergebnis nun noch verifizieren, indem man die korrigierten Positionsdaten  $[x_i^w, y_i^w, z_i^w]'$  erneut ins Bild projiziert. Dafür kann man den in Abschnitt 5.3 vorgestellten Algorithmus nutzen.

## 5.5 Verschiebung der Suchmusterebene

Mit Hilfe dieser Funktion soll die berechnete Suchmusterebene um einen konstanten Faktor  $\lambda$  verschoben werden. Auf diese Weise ist es möglich, andere Ebenen, die parallel zur Suchmusterebene liegen, auszugeben oder für weitere Berechnungen zur Verfügung zu stellen. Für die Funktion benötigt man:

- Eine Distanz  $\lambda$ , um die die Ebene verschoben wird.
- Die Koeffizienten der Suchmusterebene in Koordinatenform.
- Eine Richtungsinformation, in welche die Verschiebung stattfinden soll.

Um die Ebene zu verschieben, geht man folgendermaßen vor:

1. Eine Gerade konstruieren, die senkrecht zur Ebene steht.
  - a) Den normierten Normalenvektor  $\mathbf{n}$  der Ebene bestimmen, dieser dient als Richtungsvektor für die Verschiebung.

## 5 Implementierung

- b) Einen Punkt  $S$  aus der Ebene bestimmen, er wird Stützpunkt der Normalengeraden.
  - c) Die Geradengleichung für die Normalengerade aufstellen.
2. Einen Stützpunkt  $S'$  der neuen Ebene berechnen.
  3. Aus  $S'$  und  $\mathbf{n}$  werden die Koeffizienten der neuen Ebenengleichung in Koordinatenform bestimmt.

**Zu 1.:** mit Hilfe der Geraden soll eine Parallelverschiebung der Ebene durchgeführt werden. Dies gelingt, indem man eine Entfernung  $\lambda$  in die Geradengleichung einsetzt und dadurch einen Punkt erhält, der Stützpunkt der neuen Ebene wird. Da diese Ebene parallel zur Suchmusterebene liegt, besitzen sie den gleichen Normalenvektor  $\mathbf{n}$ . Aus dem Stützpunkt und  $\mathbf{n}$  lässt sich für die neue Ebene die Ebenengleichung in Koordinatenform ermitteln.

**Zu 1.a:** der Normalenvektor  $\mathbf{n}$  besteht aus den normierten, ersten drei Koeffizienten der Koordinatenform  $E : a \cdot x + b \cdot y + c \cdot z + d = 0$ .

$$\hat{\mathbf{n}} = \begin{bmatrix} a \\ b \\ c \end{bmatrix} \Rightarrow, \mathbf{n} = \frac{\hat{\mathbf{n}}}{|\hat{\mathbf{n}}|} \quad (43)$$

**Zu 1.b:** einen Stützpunkt  $S$  der Ebene erhält man, indem man in der Ebenengleichung  $E : a \cdot x + b \cdot y + c \cdot z + d = 0$  z. B.  $y$  und  $z$  Null setzt und nach  $x$  umstellt.

$$S = \begin{bmatrix} -\frac{d}{a} \\ 0 \\ 0 \end{bmatrix} \quad (44)$$

**Zu 1.c:** aus 1.a und 1.b ergibt sich für die Normalengerade  $G$ :

$$G : \mathbf{r} = S + \lambda \cdot \mathbf{n} \quad (45)$$

Dabei muss berücksichtigt werden, ob man die Ebene in Richtung der positiven oder der negativen Z-Achse verschieben möchte. Der Normalenvektor  $\mathbf{n}$  zeigt in Richtung der Positiven Z-Achse, wenn der Winkel  $\alpha$  zwischen  $\mathbf{n}$  und Z-Achse  $\mathbf{z}^A$  kleiner als  $90^\circ$  ist.

$$\cos \alpha = \frac{\langle \mathbf{n}, \mathbf{z}^A \rangle}{|\mathbf{n}| \cdot |\mathbf{z}^A|} \Rightarrow \alpha = \arccos \left( \frac{\langle \mathbf{n}, \mathbf{z}^A \rangle}{|\mathbf{n}| \cdot |\mathbf{z}^A|} \right), \quad \text{wobei } \mathbf{Z}^A = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \quad (46)$$

Ist  $\alpha$  größer als  $90^\circ$ , dann zeigt  $\mathbf{n}$  in Richtung der negativen Z-Achse. Je nach gewünschter Richtung muss unter Umständen der Normalenvektor negiert werden.

**Zu 2.:** einen Stützpunkt  $S'$  der neuen Ebene erhält man, indem man  $\lambda$  in die Gleichung einsetzt.

$$S' = S + \lambda \cdot \mathbf{n} \quad (47)$$

**Zu 3.:** für die neue Ebene ist es zweckmäßig die Ebenengleichung ebenfalls in Koordinatenform darzustellen. Die Koeffizienten  $a'$ ,  $b'$  und  $c'$  der Ebenengleichung  $E'$  :  $a' \cdot x + b' \cdot y + c' \cdot z + d' = 0$  sind identisch mit denen der Suchmusterebene. Den Koeffizient  $d$  bestimmt man über das Skalarprodukt von  $S'$  und  $\mathbf{n}$ . Dementsprechend gilt für die neue Ebene:

$$E' : a \cdot x + b \cdot y + c \cdot z - \langle S', \mathbf{n} \rangle = 0 \quad (48)$$

## 5.6 Datenerhebung und Auswertung

Dieser Abschnitt beinhaltet die Auswertung der erhobenen Datensätze. Zuvor wird beschrieben, wie die Daten gewonnen wurden.

### Datenerhebung

In einem Arbeitsraum der Humboldt-Universität zu Berlin wurde ein Lagermodell aufgebaut, um die entwickelten, intelligenten Kameras zu testen. Zunächst wurde der Raum genau vermessen und eine Karte angefertigt. Dann wurden sechs Kameras aufgestellt, deren Position ebenfalls im Weltkoordinatensystem bestimmt wurde. Auf dem Fußboden wurden Marker angebracht, um die Kameras auszurichten und ihren Sichtbereich in Weltkoordinaten zu bestimmen. In Abbildung 28 sind die Bilder aller sechs Kameras und ihre Sichtbereiche dargestellt. Die Sichtbereiche sind in der Karte in den Farben der einzelnen Kameras abgebildet. Die im Bild befindlichen Kreuze stellen die Marker dar. Die Sichtbereiche der Kameras überschneiden sich zum Teil, damit die schlechtere Erkennungsrate am Randbereich der Kamera kompensiert werden kann.

Während der Datenerhebung wurden drei ferngesteuerte Gabelstaplermodelle mit einem auf dem Dach befindlichen Suchmuster durch den Raum gesteuert. Dabei wurden typische Situationen, die in einem Lager auftreten können, nachgestellt. Es wurden „Lagergüter“ aufgenommen und abgestellt, sowie mit unterschiedlichen Geschwindigkeiten transportiert. Die Datenerhebung umfasste mehrere Durchläufe, die jeweils zwischen 19 und 32 Minuten dauerten. Dabei wurde darauf geachtet, dass die äußeren Bedingungen konstant blieben. Zur Auswertung lagen schließlich 18 Videodateien bereit, die jeweils zwei mal verarbeitet wurden. Beim ersten Mal war die Lagekorrektur der fehlerhaften Positionsdaten deaktiviert. Beim zweiten Mal war die Lagekorrektur aktiviert. Als Ergebnis der Verarbeitung entstand jeweils ein Datensatz, der neben den erkannten Positionen in 3D-Welt und

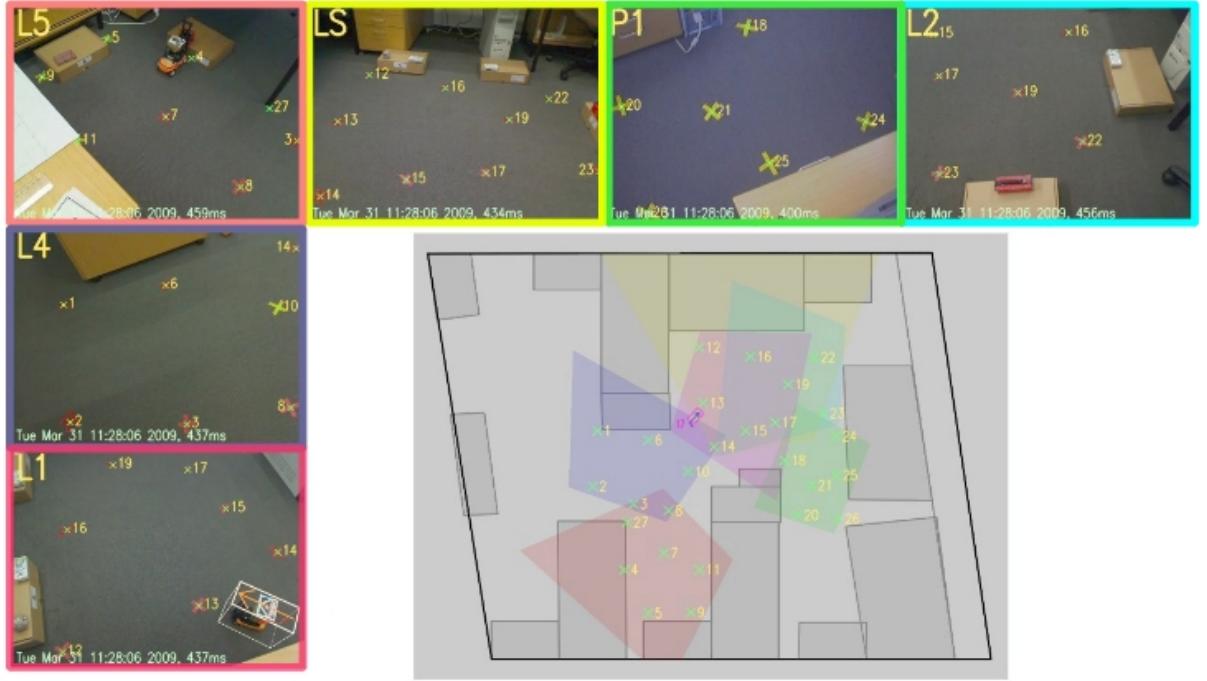


Abbildung 28: Die aktuellen Bilder aller Kameras und ihre Sichtbereiche.

2D-Bildkoordinaten auch die Gültigkeitsinformationen der einzelnen Sichtungen enthält. Somit wurden zu jeder Videodatei zwei Datensätze erstellt. Diese Datensätze wurden zur Auswertung herangezogen.

## Auswertung der Datensätze

Insgesamt wurde das Suchmuster in den Datensätzen ohne Korrektur der Lage 42285 mal von der intelligenten Kamera erkannt. War die Lagekorrektur aktiviert, so wurde das Suchmuster 42286 mal gefunden. Wie es zu diesem Unterschied kommt, konnte nicht endgültig geklärt werden. Vermutlich lässt er sich auf Fehldetektionen am Anfang der Datenerfassung zurückführen.

Die Auswertung der Datensätze führte zu einem eindeutigen Ergebnis: Ohne die Lagekorrektur sind mit 45,82 Prozent fast die Hälfte aller erkannten Suchmuster fehlerhaft erkannt und ihre Positionsdaten ungültig. Demzufolge wären 19375 Sichtungen des Suchmusters als fehlerhaft verworfen worden. Mit aktiver Lagekorrektur ließen sich die fehlerhaften Lösungen bei der Positionsbestimmung des Suchmusters auf 0,23 Prozent reduzieren. Durch die Lagekorrektur können in diesem Fall 19279 normalerweise verlorene Positionsdaten korrigiert werden. Die Daten sind in Tabelle 6 nochmal zusammengefasst.

Der Anteil der fehlerhaften Lösungen bei aktiver Lagekorrektur lässt sich weiter reduzieren, wenn man die Dauer der Beobachtung verlängern würde. Denn die meisten

## 5 Implementierung

fehlerhaften Lösungen traten am Anfang der Datensätze auf. Das liegt daran, dass sich die Ebene erst nach einigen Beobachtungen zuverlässig aufgebaut hat und daher noch nicht sofort zur Lagekorrektur genutzt werden kann. Unter realen Bedingungen würden die Kameras ununterbrochen laufen. Dementsprechend würden die wenigen fehlerhaften Positionsdaten, die zu Beginn auftreten können, nicht mehr ins Gewicht fallen. In Kapitel 6 ist eine Möglichkeit vorgeschlagen, wie die Anzahl dieser fehlerhaften Positionsdaten weiter reduziert werden kann.

	Datensätze ohne Korrektur der Lage	Datensätze mit Lagekorrektur
Summe der erkannten Muster	42285	42286
davon fehlerhaften Lösungen	19375	96
korrekt erkannten Muster	22910	42190
Anteil fehlerhafte Lösungen	45,82 %	0,23 %
Anteil korrekte Lösungen	54,18 %	99,77 %

Tabelle 6: Die Gegenüberstellung der Datensätze.

## Aussagen zum Rechenaufwand

Angesichts der zusätzlich erforderlichen Rechenoperationen durch die Berechnung der Suchmusterebene und die Lagekorrektur von fehlerhaften Positionsdaten steigt der Rechenaufwand. Ein Vergleich der Rechenzeit der beiden Versionen der intelligenten Kamera (deaktivierte Ebenenberechnung, aktivierte Ebenenberechnung) ergab folgendes: Bei der gleichzeitigen Verarbeitung von vier aufgezeichneten Videodateien (mit fünf Bildern pro Sekunde) benötigte der Kamerakonzentrator bei deaktivierter Ebenenberechnung 22 Minuten und 56 Sekunden. Bei aktiver Lagekorrektur wurden für die gleichen Videodateien 22 Minuten und 55 Sekunden benötigt. Es ergibt sich demnach keine Verlängerung der Rechenzeit durch die Lagekorrektur der fehlerhaften Positionsdaten. Allerdings betrug die Originallänge der Videosequenz 19 Minuten und 32 Sekunden. Eine Verarbeitung der vier Videodateien in Echtzeit ist demnach auf diesem System nicht möglich (Core 2 Duo P8700 CPU mit 2,53 GHz und 4 GB DDR2-800 RAM).

# 6 Zusammenfassung und Ausblick

## Zusammenfassung

Im Zuge der Arbeit wurde ein Verfahren zur Lagekorrektur von Kalibrierungsmustern in Bildsequenzen entwickelt, implementiert und getestet. Zunächst wurden Grundlagen zu Kalibriermustern, zentralprojektive Abbildungen, Kameraparameter, Kamerakalibrierung und Ebenendarstellungsformen vermittelt. Anschließend wurden einige Verfahren zur Wiederherstellung von Weltkoordinaten aus Bildkoordinaten und Ausreißertests, sowie die Singulärwertzerlegung vorgestellt. Zudem wurde ihre Anwendung im Verfahren zur Lagekorrektur erläutert. Zum Schluss wurde das Verfahren getestet. Die Auswertung der Test ergab eine signifikante Verbesserung der intelligenten Kamera, weil durch die Lagekorrektur fast alle fehlerhaften Positionsdaten korrigiert und somit in der Lagerlogistik verwendet werden können. Der Anteil von fehlerhaften Positionsdaten konnte von 45,82 Prozent auf 0,23 Prozent gesenkt werden, was bedeutet, dass in diesem Fall fast 20000 zusätzliche Positionsdaten gespeichert werden können.

## Ausblick

Es existieren einige Ideen, um die man die Berechnung der Ebene erweitern könnte. Diese sollen an dieser Stelle kurz vorgestellt werden.

Eine sinnvolle Erweiterung wäre die Einführung einer *Standardebene*. Damit soll erreicht werden, dass schon zu Beginn der Beobachtungen eine Ebene zur Lagekorrektur verfügbar ist. Auf diese Weise können die fehlerhaften Lösungen, die am Anfang vermehrt auftreten, reduziert werden. Dafür ist es notwendig, dass in einer Art Kalibriervorgang, vor Inbetriebnahme des Systems, eine Ebene bestimmt wird. Dieser Vorgang muss für jede Kamera individuell durchgeführt werden. Ist diese *Standardebene* für jede Kamera bestimmt worden, kann bei einem Reset des Systems oder sonstigen Vorkommnissen, die einen Verlust der Ebenendaten zur Folge haben, sofort eine gültige Ebene geladen werden. Ein weiterer Vorteil einer Initialisierung mit einer *Standardebene* ist der, dass durch diese der Aufbau einer *schlechten* Ebene verhindert wird. Eine solche Ebene könnte entstehen, wenn während des Aufbaus einer ersten Ebene viele schlechte oder ungültige (z. B. Fehldetections) Positionsdaten ermittelt werden. Wird aus diesen Positionsdaten

## *6 Zusammenfassung und Ausblick*

eine Ebene berechnet, ist sie wenig repräsentativ für die Suchmusterebene. Werden nun fehlerhafte Positionsdaten mit Hilfe dieser Ebene korrigiert, sind die daraus resultierenden Positionsdaten ungenau.

Eine weitere mögliche Erweiterung im Zusammenhang mit der Berechnung der Suchmusterebene ist der Umgang mit mehreren Bewegungsebenen. Im Moment ist das Einsatzgebiet auf Lager mit einem ebenen Untergrund beschränkt. Durch die Erweiterung wäre es möglich, Lagerhäuser mit einem anderen Höhenprofil zu unterstützen.

Sehr nützlich wäre die Erweiterung der Ebenenberechnung um einen Prüfmechanismus für die Kameraposition. Damit kann erreicht werden, dass Veränderungen an der Orientierung der Kamera mit Hilfe der Lagekorrektur erkannt werden. Treten beispielsweise plötzlich sehr viele fehlerhafte Positionsdaten auf, oder erhöhen sich die Abstände der ermittelten Positionsdaten zur Suchmusterebene signifikant, dann kann man daraus, eventuell, auf eine Veränderung der Orientierung der Kamera schließen. Eine solche Veränderung sollte schnellstmöglich bemerkt werden, da sämtliche erhobenen Positionsdaten in diesem Fall verfälscht werden.

# Literaturverzeichnis

- [1] AIF: *AIF-Homepage*. Internet, 11 2009
- [2] BOUGUET, Jean-Yves: *Camera Calibration Toolbox for Matlab*. Internet, Juni 2008.  
– "[http://www.vision.caltech.edu/bouguetj/calib\\_doc/index.html](http://www.vision.caltech.edu/bouguetj/calib_doc/index.html)"
- [3] FISCHLER, Robert C. Martin A. Bolles B. Martin A. Bolles: *Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography*. Version: June 1981.  
<http://www.tu-chemnitz.de/etit/proaut/paperdb/download/fischler81.pdf>, Abruf: 12.01.2010. Internet
- [4] FORBES, Alistair: *Sphere Fitting*
- [5] HEIKKILÄ, Janne ; SILVÉN, Olli: A Four-step Camera Calibration Procedure with Implicit Image Correction / University of Oulu. 1997. – Forschungsbericht
- [6] JÄHNE, Bernd: *Digitale Bildverarbeitung*. Springer, 1997
- [7] LEPPLA, Yifan Hu R. Stefan Zhou Z. Stefan Zhou: *Seminarband SS07 - Seminar intelligente Industrieroboter Kalibrierungsverfahren für die Erweiterte Realität*. Seminarband - PDF, 2007
- [8] LOTHAR SACHS, Jürgen H.: *Angewandte Statistik*. Springer, 2006
- [9] REULKE, Ralf: *Ausgleichungsrechnung*. PDF, 05 2009
- [10] WEISSTEIN, Eric W.: *Box-and-Whisker Plots*. Internet.  
<http://mathworld.wolfram.com/Box-and-WhiskerPlot.html>