HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Thesisarbeit

Zur Erlangung des Grades

Bachelor of Science

im Studiengang Medieninformatik
an der Fakultät Digitale Medien

Implementierung einer Anwendung zur Kamerakalibrierung

Referent	:	Prof. Dr. Thomas Schneider
Koreferent	:	Prof. Dr. Ruxandra Lasowski
Vorgelegt am	:	28.02.2022
Vorgelegt von	:	Sabine Schleise Matrikelnummer: 259013 sabine.schleise@hs-furtwangen.de Bismarckstr. 60, 78120 Furtwangen

Hochschule Furtwangen
Wintersemester 2021/22

Abstract

Die vorliegende Arbeit beschäftigt sich mit der Entwicklung einer modularisierten Anwendung zur Berechnung der Kamerakalibrierung mit dem Computeralgebrasystem *Mathematica* sowie Prüfung der Möglichkeit die Berechnungen in *Python* zu portieren. Hierfür wird einleitend die Zentralperspektive beleuchtet und die Grundlagen der projektiven Geometrie sowie die der Kamerakalibrierung erklärt. Im Anschluss wird die Vorgehensweise zur Berechnung der Kamerakalibrierung über dem Bild des absoluten Kegelschnitts und die Rekonstruktion der Objektpunkte beschrieben.
Da Fotografien durch verschiedene Störfaktoren fehlerhaft sein können, wird unter anderem auch die Robustheit der Algorithmen geprüft und mit einem Bündelblockausgleich an einer Minimierung der Fehler, die durch diese Störungen entstehen können, gearbeitet. Die Anwendung soll modular aufgebaut werden, sodass sie leicht verständlich, nachvollziehbar und erweiterbar ist.

Inhaltsverzeichnis

1 Einleitung **1**

2 Definition Kamerakalibrierung **3**

3 Projektgrundlagen **7**

3.1 Grundlagen aus der projektiven Geometrie 7

3.2 Grundlagen der Kamerakalibrierung 9

3.2.1 Einführung in die Zentralperspektive 9

3.2.2 Lochkameramodell 11

3.2.3 Exkurs Verzeichnung 15

4 Berechnungen einer Kamerakalibrierung **17**

4.1 Initiale Berechnung der Kameramatrix 17

4.1.1 Der absolute Kegelschnitt 17

4.1.2 Homografien 20

4.1.3 Das Bild des absoluten Kegelschnitts 23

4.1.4 Bestimmung der Kameramatrix 25

4.2 Initiale Rekonstruktion der Objektpunkte 25

4.2.1 Initiale Berechnung der Objektorientierung 25

4.2.2 Rekonstruktion der Objektpunkte 30

4.3 Initiale Berechnung der extrinsischen Parameter 33

4.4 Bündelblockausgleich 37

5 Algorithmen der Anwendung 41

5.1 Einführung in *Mathematica* 41

5.2 Die Anwendung in *Mathematica* 43

5.3 Projektion des Schachbretts auf die Bildebene 44

5.4 Initiale Berechnung der Kameramatrix 48

5.4.1 Berechnung der Homografien 48

5.4.2 Bestimmung des absoluten Kegelschnitts 49

5.4.3 Beobachtungen der Algorithmen unter verschiedenen Bedingungen 51

5.5 Initiale Rekonstruktion der Objektpunkte 53

5.5.1 Berechnung des Normalenvektor der Ebene 53

5.5.2 Rekonstruktion der Objektpunkte 55

5.6 Initiale Berechnung der extrinsischen Kameraparameter . . 58

5.7 Untersuchung der Algorithmen mit gestörten Sensorpunkten 60

5.8 Bündelblockausgleich 63

5.9 Portierung der Berechnungen in *Python* 70

5.9.1 Einführung in *Python* 70

5.9.2 Berechnungen in *Python* 71

5.9.3 Fazit zur Portierung der Berechnungen 74

6 Fazit und Ausblick 77

7 Anhang 79

7.1 Singulärwertszerlegung 79

7.2 Installationsanweisungen 81

7.2.1 Installation von *Mathematica* 81

7.2.2 Installation von *Python* 82

7.3 Beispiel einer Ausgabe der *Mathematica*-Datei bei ungestörten Punkten 82
7.4 Beispiel einer Ausgabe der *Mathematica*-Datei bei gestörten Punkten 87

Eidesstattliche Erklärung **93**

Abkürzungsverzeichnis **97**

Literaturverzeichnis **99**

1 Einleitung

Ob in der Fotogrammetrie, bei der aus verschiedenen Fotografien einer Szene 3D-Modelle (wie beispielsweise 3D-Karten) entstehen oder in fotogrammetrischen Luftbildbearbeitungen - die Kamerakalibrierung findet in vielen Bereichen der Technik ihre Anwendung [1]. Um fotografierte Szenen rekonstruieren zu können, muss eine Kamerakalibrierung zur Bestimmung der Abbildungsparameter einer Kamera durchgeführt werden.. Durch die stetige Weiterentwicklung der Kameras, von Spiegelreflexkameras bis hin zu spiegellosen Systemkameras, kann die reale Welt nahezu identisch abgebildet werden. Trotz dessen entstehen Fehler, die beispielsweise durch Bildrauschen oder Verzeichnungen verursacht wurden und im Zuge der Kamerakalibrierung möglichst herausgerechnet werden sollen.
Die Aufgabenstellung besteht darin, eine Anwendung zu schreiben, bei der die Algorithmen der Kamerakalibrierung implementiert und dabei in ihrer Robustheit und Funktion geprüft werden soll, um festzustellen, wie problematisch die Berechnungen der Algorithmen bei Fehlern ist. Die Anwendung wird im kostenpflichtigen Computeralgebrasystem (CAS) *Mathematica* geschrieben und anschließend überprüft, ob sich diese Anwendung in einer modernen, effizienten und kostenfreien Programmiersprache portieren lässt. Die Arbeit soll Basis für weitere Projekte sein, bis die Anwendung schließlich als Desktopanwendung oder Application Projekt (APP) fungieren kann.

Wichtige Grundlage der Arbeit ist die projektive Geometrie, welche zuerst genauer betrachtet wird. Anschließend wird die Zentralperspektive beleuchtet und die Kamerakalibrierung anhand des Lochkameramodells erklärt und zur Bestimmung der Kameramatrix, in der die intrinsischen Kameraparameter enthalten sind, hingeführt. Das nächste Kapitel befasst sich mit der Berechnung dieser Kameramatrix über das Bild des absoluten Kegelschnitts. Nach der Berechnung der intrinsischen Kameraparameter werden die Objektpunkte rekonstruiert und die extrinsischen Kameraparameter berechnet, um anschließend einen Bündelblockausgleich durchzuführen, der den Reprojektionsfehler minimiert. Kapitel 5 befasst sich mit der Implementierung dieser Algorithmen in *Mathematica* und *Python*. Das letzte Kapitel beinhaltet eine Zusammenfassung der Arbeit und einen Ausblick auf weitere mögliche Projekte.

2 Definition Kamerakalibrierung

Bei einer Projektion, die zentral- oder parallelperspektivisch erfolgt, werden 3D-Objekte auf ein 2D-Bild abgebildet. Die Thesisarbeit befasst sich mit einer zentralperspektivischen Projektion, bei der das Modell einer Lochkamera, welches in Kapitel 3.2.2 beschrieben wird, herangezogen wird. Um das Objekt aufgrund der entstandenen Bilder rekonstruieren zu können, wird eine Kamerakalibrierung durchgeführt, bei der intrinsische und extrinsische Parameter einer Kamera berechnet werden. Dieses Kapitel behandelt die Definition der Kamerakalibrierung sowie deren Funktion. Ferner beschreibt es Methoden, mit welchen die Kalibrierung erfolgen kann. Abbildung 2.1 zeigt die zu berechnenden Parameter. Diese sind die intrinsischen Kameraparameter mit der Brennweite f, die den Abstand zwischen der Bildebene und dem optischen Zentrum C definiert und dem Hauptpunkt H, der den Lotfußpunkt des optischen Zentrums auf dem Sensor markiert. [2]. Die extrinsischen Parameter geben die Orientierung der Kamera im Bezug auf das Weltkoordinatensystem des Objekts an. Für jede Perspektive des Objekts werden dabei die Rotationsparameter R und die Translationsparameter t bestimmt. Während dem Prozess der Kamerakalibrierung werden verschiedene Koordinatensysteme verwendet, die ebenfalls in Abbildung 2.1 eingezeichnet sind. Es existiert ein Weltkoordinatensystem, ein Kamerakoordinatensystem und ein Sensorkoordinatensystem. Die intrinsischen Kameraparemter definieren dabei die Abbildung zwischen dem Kamerakoordinatensystem und dem Sensorkoordinatensystem [3].

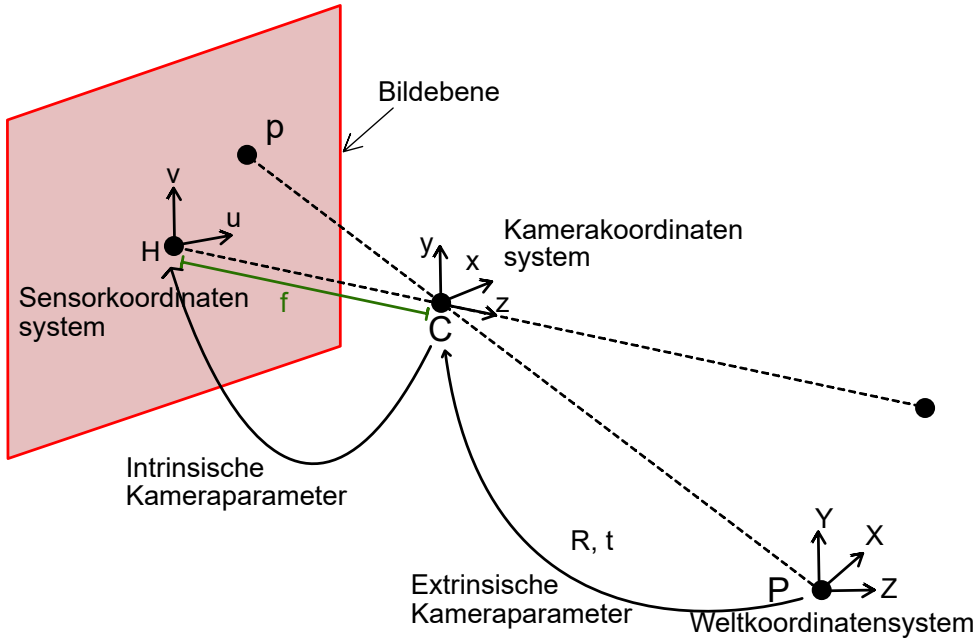


Abbildung 2.1: Intrinsische und extrinsische Kameraparameter (vgl. [4])

Abbildung 2.2 veranschaulicht die extrinsischen Kameraparameter, gegeben durch eine Rotation um die Euler-Winkel α, β, γ und einer Verschiebung des Objekts vom Ursprung des Weltkoordinatensystems mit dem Translationsvektor $t = (\ t_1,\ t_2,\ t_3\ )$, welche die Abbildung zwischen dem Weltkoordinatensystem des Objekts und dem Kamerakoordinatensystem beschreiben [3].

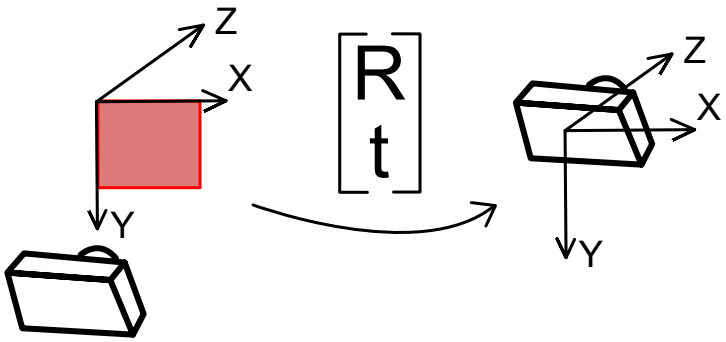


Abbildung 2.2: Extrinsische Kameraparameter (vgl. [2])

Um eine Kamera zu kalibrieren werden verschiedene Methoden angewandt. Eine Möglichkeit besteht darin, die Kamera mit einem 3D-Kalibrierkörper zu kalibrieren, welches jedoch teuer und aufwändig ist [5]. Die gängigste Methode zur Ermittlung der Kameraparameter ist das Verfahren, das von Tsai veröffentlicht wurde. Hierbei werden 3D-Objektkoordinaten und 2D-Pixelkoordinaten benötigt, um möglichst viele Parameter initial zu schätzen und mit einem nichtlinearen Optimierungsverfahren zu verfeinern [6]. Die Thesisarbeit befasst sich mit der von Hartley und Zisserman entwickelte Methode, bei der die Kameramatrix über das Bild des absoluten Kegelschnitts ermittelt wird [7].

3 Projektgrundlagen

Dieses Kapitel behandelt die Grundlagen der Kamerakalibrierung. Um ein Basiswissen zu schaffen, beginnt das Kapitel mit einer Einführung in die projektive Geometrie. Anschließend wird die Zentralperspektive beleuchtet, bei der das Modell einer Lochkamera herangezogen wird, welche die ideale Beziehung der Koordinaten eines Punktes im 3D-Raum und seiner Projektion auf die Bildebene darstellt. Ausgehend von der Grundlage, wird im Folgenden an die Kameramatrix herangeführt, aus welcher die intrinsischen Parameter einer Kamera ausgelesen werden können.

3.1 Grundlagen aus der projektiven Geometrie

Ziel dieses Kapitels ist es, ein Grundverständnis für die projektive Geometrie zu schaffen, welche ein Teilgebiet der Mathematik ist und eine systematische Untersuchung und Darstellung der Gesetzmäßigkeiten zentralperspektivischer Abbildungen ermöglicht.

Ein wichtiges algebraisches Hilfsmittel in der projektiven Geometrie ist die Erweiterung des gewöhnlichen affinen Raums um eine Dimension zum sogenannten projektiven Raum. Bei diesem wird ein Punkt $P = (x, y, z)$ um eine Dimension erweitert, wobei die erweiterte Komponente, die der homogenen Erweiterung ist [8]. Mit der Erweiterung ist ein Punkt im projektiven Raum nicht wie im affinen Raum eindeutig definiert, sondern auch

alle Vielfachen dieses Punktes $P = (\lambda x,\, \lambda y,\, \lambda z,\, \lambda 1\,)$ [8].

$$P \;:\; \mathbb{R}^3 \to \mathbb{R}^4, \begin{pmatrix} x \\ y \\ z \end{pmatrix} \mapsto \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \in \mathbb{R} \mid x,\, y,\, z \in \mathbb{R}$$

$$P \;=\; \lambda \cdot (x,\, y,\, z,\, 1\,) \mid \lambda \in \mathbb{R}$$

Bei einer projektiven Ebene, welche aus der affinen Ebene konstruiert wird, werden unendlich ferne Punkte hinzugenommen. Für diese projektive Ebene sollten drei Axiome erfüllt werden [9]:

1. Durch zwei verschiedene Punkte geht genau eine Gerade
2. Zwei verschiedene Geraden schneiden sich in genau einem Punkt
3. Es gibt 4 Punkte, so dass keine 3 Punkte auf einer Geraden liegen

Der Fall, dass Geraden parallel verlaufen und sich in keinem Punkt schneiden, existiert in der projektiven Geometrie nicht mehr, da sich das 2. Axiom nicht erfüllen würde. Ein Punkt in der projektiven Ebene legt im $\mathbb{R}^3$-Raum eine Gerade fest, die durch den Ursprung $\mathbb{O} = (0,\, 0,\, 0)$ und dem Punkt $P = [x,\, y,\, w]$ verläuft. Ist $w = 0,$ handelt es sich um einen Fernpunkt. Projektive Fernpunkte, im $\mathbb{R}^3$-Raum Ursprungsgeraden, sind Punkte, die die Einbettungsebene $z = 1$ nicht schneiden, also in der x-y-Ebene liegen. Mit $P = [\frac{x}{w}, \frac{y}{w}, 1]$ kann eine Standarddarstellung des Punktes erzeugt werden. Eine Gerade, die durch zwei verschiedene Punkte $P_1 = (x_1,\, y_2,\, 1)$ und $P_2 = (x_2,\, y_2,\, 1)$ verläuft, entspricht dabei einer $\mathbb{R}^3$-Ebene, die den Ursprung $\mathbb{O}$ enthält [9]. Die projektive Gerade kann auch in der linearen Form $ax + by + cz = 0$ geschrieben werden [10].

3.2 Grundlagen der Kamerakalibrierung

Dieses Kapitel behandelt die Hinführung zur Kameramatrix, aus welcher sich die intrinsischen Kameraparameter auslesen lassen. Hierbei soll ein Grundverständnis der Zentralprojektion entstehen und die Abhängigkeiten dieser erklärt werden. Ziel ist es den Zusammenhang der geometrischen Konzepte und der Kamerakalibrierung zu erklären.

3.2.1 Einführung in die Zentralperspektive

Ein 3D-Objekt lässt sich durch eine Zentralprojektion auf ein 2D-Bild abbilden. Für die Einführung in die Zentralperspektive wird als Modell für die Kamera das sogenannte Lochkameramodell verwendet. Die Lochkamera, die in Kapitel 3.2.2 genauer beschrieben wird, erzeugt eine zentralperspektivische Darstellung, wie sie in Abbildung 3.1 zu sehen ist.

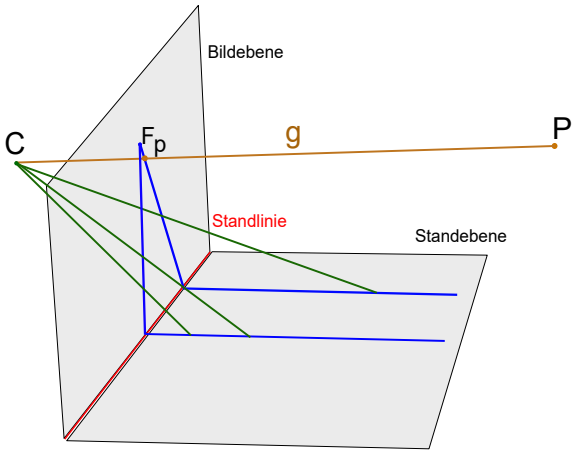


Abbildung 3.1: Zweidimensionale Konstruktion der Perspektive

C stellt hierbei das optische Zentrum, auch Projektionszentrum genannt, dar, welcher den Ausgangspunkt dieser Zentralprojektion markiert. Jeder Raumpunkt P wird durch einen Schnitt der Verbindungsgeraden $g = \overrightarrow{CP}$ mit der Bildebene auf die Bildebene abgebildet. Dabei ist jede Objekt-

gerade g, die parallel zur Bildebene verläuft, eine Gerade im zentralperspektvischen Bild, die parallel zu g ist. Objektgeraden, die zueinander parallel sind, aber nicht parallel zur rot abgebildeten Standlinie, schneiden sich in den Projektionsbildern in einem Punkt, der sich Fluchtpunkt nennt. Dieser bezeichnet einen realen Punkt in der Bildebene, der die Menge aller zueinander parallelen Geraden schneidet. Durch eine solche zentral-perspektivische Abbildung werden Größenverhältnisse nicht mehr korrekt abgebildet. So werden Objekte, die gleich groß sind, je nach Entfernung zum Projektionszentrum immer kleiner dargestellt. Abbildung 3.2 zeigt das Projektionsbild eines Schachbrettmusters, bei der sich neben dem räumlichen Eindruck auch die zwei Fluchtpunkte V_{p1}, V_{p2} ergeben [8].

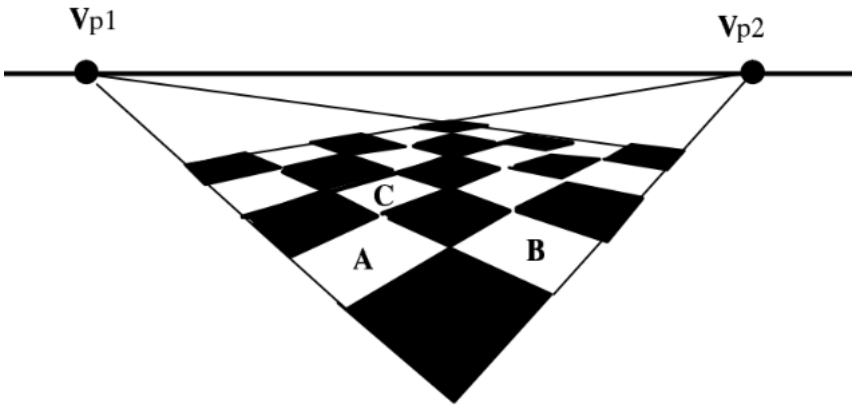


Abbildung 3.2: Projektionsbild eines Schachbrettmusters (vgl. [11])

3.2.2 Lochkameramodell

Um eine Projektion der 3D-Welt auf ein 2D-Bild zu veranschaulichen, wird das einfache Lochkameramodell verwendet, bei der sich die Zentralprojektion ideal modellieren lässt. Die ideale Lochkamera ist so aufgestellt, dass Lichtstrahlen nur durch eine sehr kleine Öffnung auf den Bildsensor einströmen kann [7].

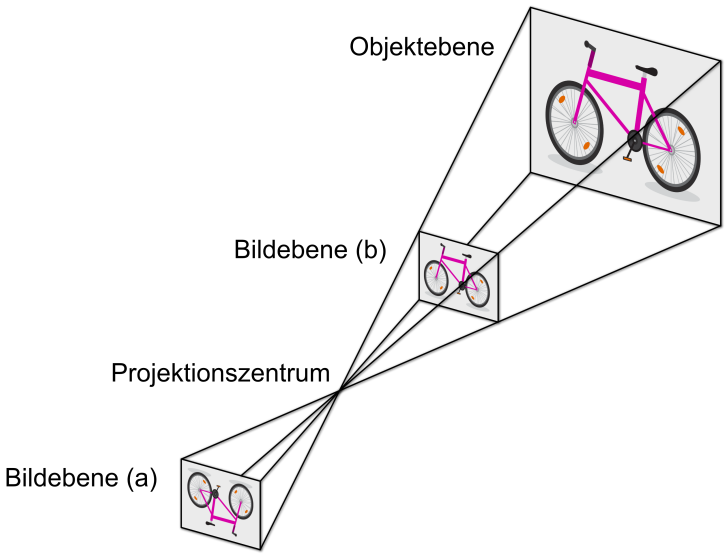
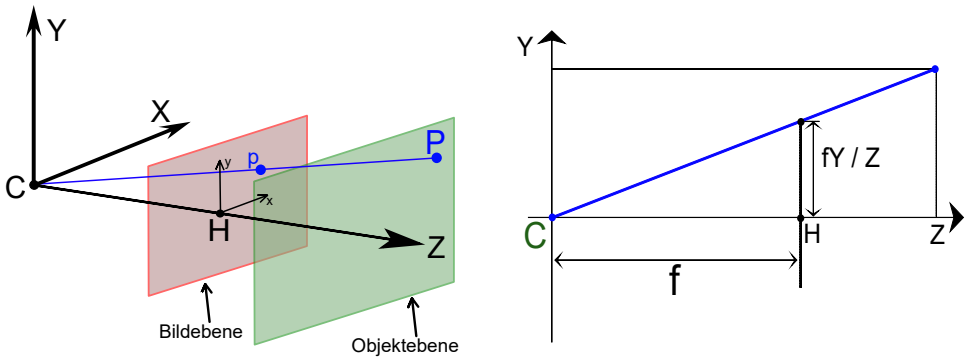


Abbildung 3.3: Das Lochkameramodell

Im Normalfall wird durch die Zentralprojektion ein Negativbild, in Abbildung 3.3 mit der Bildebene (a) markiert, erzeugt. Durch eine einfache Transformation kann ein Negativbild in ein Positivbild (Bildebene (b)) umgewandelt werden. Da sich die Brennweite und der Hauptpunkt durch die Transformation nicht ändern, wird zur Vereinfachung das Lochkameramodell in einer Positivlage verwendet. In dem Lochkameramodell, das in Abbildung 3.4 dargestellt wird, wird ein Punkt im Raum mit den Koordinaten $P = (X, Y, Z)$ auf einen Punkt $p = (x, y)$ auf die Bildebene abgebildet, davon ausgehend, dass $Z = f$ ist und das Projektionszentrum hierbei im Ursprung eines euklidischen Koordinatensystems liegt [7].



(a) (b)

Abbildung 3.4: Einfaches Lochkameramodell in Positivlage (vgl. [7])

Das Kamerazentrum C bildet das Projektionszentrum der Zentralprojektion. Die Hauptachse Z, auch optische Achse genannt, bezeichnet die Gerade, die im rechten Winkel zur Bildebene durch das Projektionszentrum verläuft. Dabei ist der Hauptpunkt H der Punkt, an der sich die optische Achse mit der Bildebene schneidet [7]. Diese Zentralprojektion kann mit dem zweiten Strahlensatz zur Projektion eines Szenenpunkts $P = (X, Y, Z)$ auf eine Bildebene $p = (x, y)$ beschrieben werden. Für die y-Koordinate und x-Koordinate des Punktes gilt hierbei:

$$\frac{Y}{Z} = \frac{y}{f} \quad \longrightarrow \quad y = f \cdot \frac{Y}{Z} \tag{3.1}$$

$$\frac{X}{Z} = \frac{x}{f} \quad \longrightarrow \quad x = f \cdot \frac{X}{Z} \tag{3.2}$$

Im Gesamten kann die zentralperspektivische Abbildung von Szenen- zu Bildpunkt damit wie folgt beschrieben werden:

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} f\frac{X}{Z} \\ f\frac{Y}{Z} \end{pmatrix} \tag{3.3}$$

Wird bei der Zentralprojektion die homogene Erweiterung hinzugefügt, kann die Zentralprojektion als lineare Abbildung zwischen dem projektiven Raum und einer projektiven Ebene beschrieben werden [7].

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \longmapsto \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} fX \\ fY \\ Z \end{bmatrix} = \begin{bmatrix} \frac{fX}{Z} \\ \frac{fY}{Z} \\ 1 \end{bmatrix} \tag{3.4}$$

Für den nächsten Schritt wird neben dem räumlichen Kamerakoordinatensystem, dessen Ursprung im Projektionszentrum liegt, das Bildkoordinatensystem eingeführt, welches in Abbildung 3.5 dargestellt wird. Dabei beschreiben die Koordinaten h_1 und h_2 die Lage des Hauptpunktes bezüglich des Bildkoordinatensystems $H_{BK} = (h_1, h_2)$.

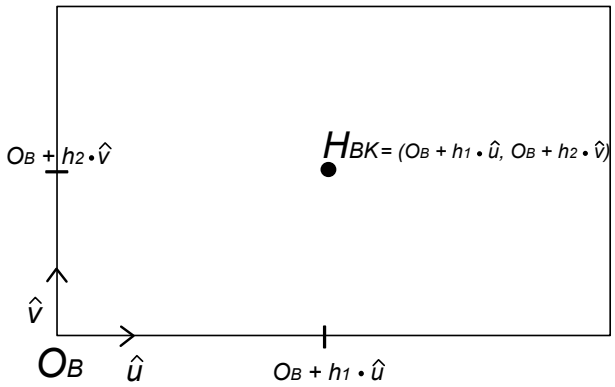


Abbildung 3.5: Koordinaten von H bezüglich des Bildkoordinatensystems

Da der Hauptpunkt nicht im Koordinatenursprung in der Bildebene liegen muss, sollte der Ausdruck 3.3 erweitert werden [7].

$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} f\frac{X}{Z} + h_1 \\ f\frac{Y}{Z} + h_2 \end{pmatrix} \tag{3.5}$$

In homogenen Koordinaten wird der Ausdruck 3.5 folgendermaßen ergänzt.

$$\begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} \longmapsto \begin{bmatrix} f & 0 & h_1 & 0 \\ 0 & f & h_2 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} fX + Zh_1 \\ fY + Zh_2 \\ Z \end{bmatrix} = \begin{bmatrix} \frac{fX+Zh_1}{Z} \\ \frac{fY+Zh_2}{Z} \\ 1 \end{bmatrix}$$

Aus dem zuletzt ergänzten Ausdruck kann die Kameramatrix K entnommen werden.

$$K = \begin{bmatrix} f & 0 & h_1 \\ 0 & f & h_2 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

Diese Kameramatrix entstammt dem Modell der idealen Lochkamera. Für digitale Sensorkameras sollte noch hinzukommen, dass Bildkoordinaten in Pixelkoordinaten gemessen werden. Während das Lochkameramodell davon ausgeht, dass die Bildkoordinaten in x und y-Richtung denselben Maßstab haben, muss für digitale Kameras noch ein Skalierungsfaktor m eingeführt werden. Die Parameter fm_x und fm_y beschreiben die Brennweite in Pixelkoordinaten, die den Abstand zwischen dem optischen Zentrum und dem Hauptpunkt markiert. Auch der Hauptpunkt H wird in Pixelkoordinaten gemessen, weswegen die Komponenten h_1m_x und h_2m_y erweitert werden müssen. Es kann vorkommen, dass die x und y-Achse nicht exakt senkrecht aufeinander stehen. In diesem Fall kommt ein Scherungsfaktor s, der *skew*-Faktor genannt wird, hinzu [7]. Damit wird die

Kameramatrix K für digitale Kameras wie folgt definiert:

$$K = \begin{bmatrix} fm_x & s & h_1 m_x \\ 0 & fm_y & h_2 m_y \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7}$$

Mit der Berechnung dieser Kameramatrix können die intrinsischen Parameter der Kamera ermittelt werden.

3.2.3 Exkurs Verzeichnung

In der Fotografie kommt es häufig zu Abbildungsfehler, da ein Objektiv nicht dem idealen Modell einer Lochkamera entspricht. Der häufigste Abbildungsfehler ist dabei die Verzeichnung, die in Folge eines Öffnungsfehlers auftritt [12]. Abbildung 3.6 zeigt eine Verzeichnung mit unterschiedlicher Blendenlage.

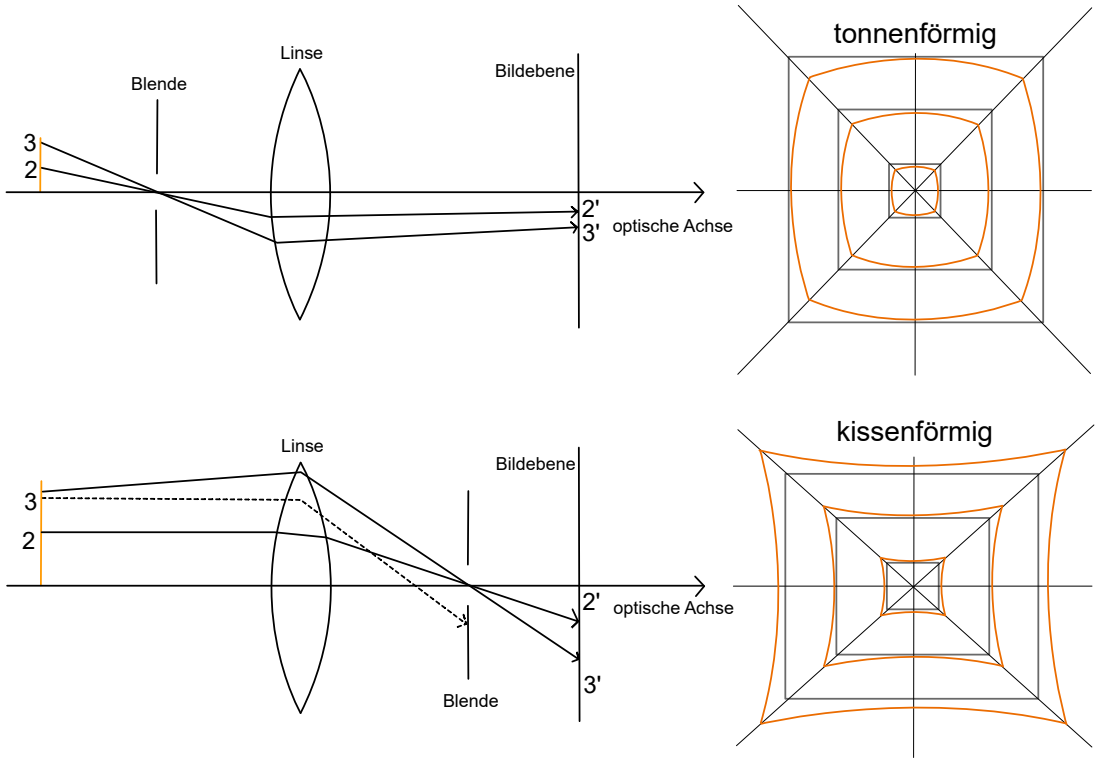


Abbildung 3.6: Verzeichnung abhängig von der Blendenlage

Bei der tonnenförmigen Verzeichnung liegt die Blende vor der Linse. Die äußeren Strahlen werden dabei stärker gebrochen als die inneren, was dazu führt, dass die äußeren Bildpunkte dichter aneinander liegen. Die kissenförmige Verzeichnung zeichnet sich dadurch aus, dass die äußeren Bildpunkte größere Abstände haben, als die inneren. Hier befindet sich die Blende hinter der Linse [12]. Die Berechnungen der Kameramatrix gehen von einer linearen Modellierung aus. Dies bedeutet, es liegt das Bild eines Schachbretts vor, bei der Geraden auch als Geraden abgebildet werden. Bei Verzeichnungen sind diese Geraden jedoch gekrümmte Linien, die vor der Berechnung korrigiert werden müssen. In der Thesisarbeit wird davon ausgegangen, dass die lokalisierten Schachbretteckpunkte bereits verzeichnungsfrei sind.

4 Berechnungen einer Kamerakalibrierung

Mit der Kamerakalibrierung werden intrinsische und extrinsische Kameraparameter initial berechnet. Für diese Parameter wird ein Bündelblockausgleich, bei der initial berechnete Werte optimiert werden, durchgeführt. Dieses Kapitel befasst sich mit dieser Kamerakalibrierung, bei der am Ende die optimierten Abbildungsparameter berechnet werden.

4.1 Initiale Berechnung der Kameramatrix

Im Folgenden wird die Bestimmung der intrinsischen Kameraparameter behandelt. Hierfür wird die Kameramatrix über dem Bild des absoluten Kegelschnitts berechnet. Dazu ist es notwendig, Homografien zu berechnen, mit der das Bild des absoluten Kegelschnitts bestimmt werden kann. Aus dem Bild des absoluten Kegelschnitts lässt sich die Kameramatrix durch die Zerlegung der *omega*-Matrix bestimmen.

4.1.1 Der absolute Kegelschnitt

Bei einem Schnitt eines Kegels mit einer Ebene entstehen Kegelschnitte in Form eines Kreises, einer Ellipse, einer Parabel oder einer Hyperbel. Ein Teil der projektiven Geometrie ist der absolute Kegelschnitt, im englischen *absolute Conic* genannt, der einen Kreis im projektiven Raum beschreibt, bei der sich die Ebene im Unendlichen befindet und im Schnittpunkt aller Kegeln im Raum liegt [7, 8].

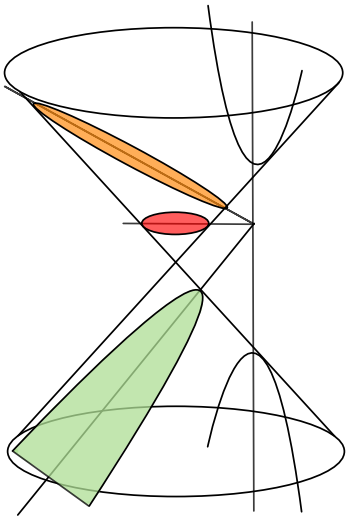


Abbildung 4.1: Formen eines Kegelschnitts

Er beschreibt den Schnittpunkt einer projektiven Quadrik 4.1, bei der die vierte Komponente als die der homogenen Erweiterung hinzugefügt wurde, mit einer Fernebene 4.2.

$$X_1^2 + X_2^2 + X_3^2 + X_4^2 = 0 \tag{4.1}$$

$$X_4 = 0 \tag{4.2}$$

Auf diesem absoluten Kegelschnitt können dann zum Beispiel die komplexen Fernpunkte $(1, -i, 0, 0)$ und $(1, i, 0, 0)$ liegen. Zu jeder Ebene im Raum existieren also zwei Punkte auf dem absoluten Kegelschnitt. Ist die Ebenengleichung bekannt, könnten die Punkte auf dem absoluten Kegelschnitt berechnet werden.
Gegeben sei beispielsweise die Ebene in allgemeiner Koordinatenform

$$x_1 + x_2 + x_3 - 1 = 0$$

Wird die Komponente der homogenen Erweiterung hinzugefügt, erweitert sich die Ebenengleichung zu $x_1 + x_2 + x_3 - x_4 = 0$. Um den absoluten Kegelschnitt zu berechnen, muss dann folgendes Gleichungssystem gelöst

werden.

$$X_1 + X_2 + X_3 - X_4 = 0 \tag{4.3}$$

$$X_1^2 + X_2^2 + X_3^2 + X_4^2 = 0 \tag{4.4}$$

$$X_4 = 0 \tag{4.5}$$

Um das Gleichungssystem zu lösen, wird die Gleichung 4.5 in 4.3 eingesetzt.

$$X_1 + X_2 + X_3 - 0 = 0 \tag{4.6}$$

$$X_1 + X_2 + X_3 = 0 \tag{4.7}$$

$$X_3 = -X_1 - X_2 \tag{4.8}$$

Gleichung 4.8 in 4.4 eingesetzt, ergibt:

$$\begin{aligned} X_1^2 + X_2^2 + (-X_1 - X_2)^2 &= 0 \\ X_1^2 + X_2^2 + (X_1 + X_2)^2 &= 0 \\ 2X_1^2 + 2X_2^2 + 2X_1X_2 &= 0 \\ X_1^2 + X_2^2 + X_1X_2 &= 0 \end{aligned}$$

Wird für $X_2 = 1$ gewählt, ergibt sich die Gleichung

$$X_1^2 + X_1 + 1 = 0. \tag{4.9}$$

Gleichung 4.9 lässt sich mit der *pq*- oder *abc*-Formel lösen:

$$x_{1,2} = -\frac{1}{2} \pm \sqrt{\frac{1}{4} - 1} \tag{4.10}$$

$$= -\frac{1}{2} \pm \sqrt{-\frac{3}{4}} = -\frac{1}{2} \pm i\sqrt{\frac{3}{2}} \tag{4.11}$$

Die Lösungsmenge des Gleichungssystems und die Punkte auf dem absoluten Kegelschnitt können damit geschrieben werden als

$$\mathbb{L} = \left\{ \begin{bmatrix} -\frac{1}{2} \pm i\sqrt{\frac{3}{2}} \\ 1 \\ -(\frac{1}{2} \pm i\sqrt{\frac{3}{2}}) \\ 0 \end{bmatrix} \right\}.$$

Der absolute Kegelschnitt ist unabhängig von der Orientierung der Kamera und steht im Zusammenhang mit der Kameramatrix K [8]. Bei einer Zentralprojektion wird auch der absolute Kegelschnitt (auf englisch *absolute conic (AC)* auf die Bildebene abgebildet, welches dabei das Bild des absoluten Kegelschnitts (*image of absolute conic (IAC)*) genannt wird [7]. Mit der bekannten Projektionsmatrix lassen sich die Punkte auf die Bildebene projizieren und daraus die ω-Matrix, auf die im weiteren Verlauf eingegangen wird, ermitteln. Da die Projektionsmatrix jedoch nicht bekannt ist, muss das Bild des absoluten Kegelschnitts über den Weg der Homografien bestimmt werden.

4.1.2 Homografien

Eine Homografie ist eine projektive Transformation, die eine Gruppe von Punkten auf eine andere Gruppe von Punkten und Linien auf andere Linien projiziert. Zu je zwei Urvierecken $P_{1..4}$ und $P'_{1..4}$ in allgemeiner Lage existiert genau eine projektive Abbildung, die die Punkte auf die Bilder abbildet [13]. Diese wird durch eine 3x3 Matrix beschrieben, welche z. B. die optimale, unverzerrte Anordnung der Schachbrettpunkte auf die Sensorpunkte, die in einer Ebene liegen, abbildet [8]. In allgemeiner Lage bedeutet das, dass drei dieser Punkte nicht kollinear sein dürfen, was wiederum

bedeutet, dass ein Punkt nicht auf einer Punktgerade zweier verbundener Punkte liegen darf.

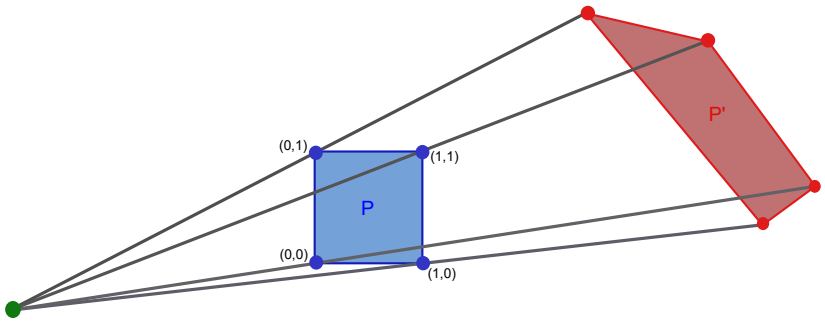


Abbildung 4.2: Homografie

Ausgeartete Vierecke erfüllen das 3. Axiom nicht und könnten Probleme bei der Berechnung der Homografie bereiten. Für die Homografiematrix gilt:

$$\begin{aligned} H \cdot P_1 &= \lambda \cdot P_1' \\ H \cdot P_2 &= \lambda \cdot P_2' \\ H \cdot P_3 &= \lambda \cdot P_3' \\ H \cdot P_4 &= \lambda \cdot P_4' \end{aligned} \qquad H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Pro Indize- und Sensorpaar entstehen damit insgesamt drei Gleichungen.

$$\begin{aligned} h_{11} \cdot P_x + h_{12} \cdot P_y + h_{13} \cdot P_z &= \lambda \cdot P_x' \\ h_{21} \cdot P_x + h_{22} \cdot P_y + h_{23} \cdot P_z &= \lambda \cdot P_y' \\ h_{31} \cdot P_x + h_{32} \cdot P_y + h_{33} \cdot P_z &= \lambda \cdot P_z' \end{aligned}$$

Da sich sowohl der Indize-, als auch der Sensorpunkt im homogenen Raum befinden, gilt für $P_z = 1$ und $P_z' = 1$ folgendes:

$$h_{31} \cdot P_x + h_{32} \cdot P_y + h_{33} = \lambda$$

Die letzte Gleichung kann nun in die oberen Gleichungen eingesetzt werden.

$$\begin{aligned} h_{11} \cdot P_x + h_{12} \cdot P_y + h_{13} &= (h_{31} \cdot P_x + h_{32} \cdot P_y + h_{33}) \cdot P'_x \\ h_{21} \cdot P_x + h_{22} \cdot P_y + h_{23} &= (h_{31} \cdot P_x + h_{32} \cdot P_y + h_{33}) \cdot P'_y \end{aligned}$$

Um ein homogenes Gleichungssystem zu erzeugen, wird die rechte Seite von der Gleichung der linken Seite subtrahiert.

$$\begin{aligned} h_{11} \cdot P_x + h_{12} \cdot P_y + h_{13} - (h_{31} \cdot P_x + h_{32} \cdot P_y + h_{33}) \cdot P'_x &= 0 \\ h_{21} \cdot P_x + h_{22} \cdot P_y + h_{23} - (h_{31} \cdot P_x + h_{32} \cdot P_y + h_{33}) \cdot P'_y &= 0 \end{aligned}$$

Ein homogenes Gleichungssystem mit der erweiterten Koeffizientematrix $(A \,|\, b)$ nennt sich homogen, wenn für $b = 0$ gilt. Für ein homogenes Gleichungssystem gibt es als Lösung immer die triviale Lösung $(0, 0, 0, ..., 0)$ [14]. A ist hierbei die Koeffizientenmatrix, die mit allen Sensorpunkten ein überbestimmtes Gleichungssystem bildet, bei der das Gleichungssystem mehr Gleichungen als Unbekannte hat. Bei einem Schachbrett gibt es mehr als ein Schachbrettquadrat, wodurch das Gleichungssystem immer überbestimmt ist und in der Regel keine exakte Lösung ermittelt werden kann, weshalb eine Singulärwertszerlegung (SVD), auf die in Kapitel 7.1 näher eingegangen wird, durchgeführt werden sollte. Diese SVD zerlegt dieses überbestimmte homogene Gleichungssystem und die Lösung kann der letzten Spalte der V-Matrix, welche mit der SVD erzeugt wird, entnommen werden. Pro Sensorpunktebene ergibt sich mit der SVD eine Homografiematrix, welche für die Bestimmung des absoluten Kegelschnitts, der im nächsten Abschnitt folgt, benötigt wird. Insgesamt werden für diese Berechnung mindestens drei Homografien benötigt.

4.1.3 Das Bild des absoluten Kegelschnitts

Das Bild des absoluten Kegelschnitts kann durch eine symmetrische 3x3 Matrix beschrieben werden.

$$\omega = \begin{bmatrix} \omega_{11} & \omega_{12} & \omega_{13} \\ \omega_{12} & \omega_{22} & \omega_{23} \\ \omega_{13} & \omega_{23} & \omega_{33} \end{bmatrix}$$

Um eine einfache 3x3 ω-Matrix zu beschreiben, wird diese mit der einfachsten Form eines Kegelschnitts dargestellt. Die Gleichung $r^2 = x^2 + y^2$ stellt einen Kreiskegelschnitt dar, bei dem sich der Mittelpunkt im Ursprung und der Radius r auf einer zweidimensionalen Ebene [15] befindet. Der Kreis kann durch die Erweiterung der homogenen Komponente auch im 3D-Raum dargstellt werden. Daraus ergibt sich die Gleichung $r^2 \cdot z^2 = x^2 + y^2$. Anhand dieses Kegelschnitts wird nun eine ω-Matrix berechnet, für die gelten muss:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \cdot \omega \cdot \begin{bmatrix} x & y & z \end{bmatrix} = 0 \tag{4.12}$$

Die ω-Matrix für den Kreis würde wie folgt aussehen:

$$\omega = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -r^2 \end{bmatrix} \tag{4.13}$$

Um die ω-Matrix zu prüfen, wird diese in die Gleichung 4.12 eingesetzt.

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & -r^2 \end{bmatrix} \cdot \begin{bmatrix} x & y & z \end{bmatrix} = 0 \tag{4.14}$$

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} \cdot x + z - r^2 \cdot z^2 = 0 \tag{4.15}$$

$$x^2 + y^2 - r^2 \cdot z^2 = 0 \tag{4.16}$$

$$x^2 + y^2 = r^2 \cdot z^2 \tag{4.17}$$

Auf diese Weise lassen sich die Kegelschnitte in Form einer 3x3 Matrix darstellen [15]. Um die ω-Matrix des Bildes des absoluten Kegelschnitts zu bestimmen, werden die berechneten Homografiematrizen herangezogen. Dabei bildet jede Homografie die idealen Kreispunkte $H(1, \pm i, 0)^T$ auf Punkte des Bildes des absoluten Kegelschnitts ab [7].
Pro Homografie $h = \begin{bmatrix} h1 & h2 & h3 \end{bmatrix}$ gilt für den Real- und Imaginärteil

$$h_1^T \omega h_2 = 0 \tag{4.18}$$

$$h_1^T \omega h_1 - h_2^T \omega h_2 = 0. \tag{4.19}$$

Bei drei Homografiematrizen entsteht ein Gleichungssystem mit sechs Gleichungen und sechs Unbekannten. Bei ungestörten Punkten kann dieses homogene Gleichungssystem mit dem Befehl Nullspace[ ] gelöst werden. Dieser gibt eine Liste von Vektoren an, die eine Basis für den Nullraum der Koeffizientenmatrix bildet. In dieser Liste stehen die Werte für die Homografiematrix H. Bei gestörten Punkten ändert sich der Rang der Matrix, weshalb die ω-Matrix durch eine Ausgleichslösung berechnet werden muss.

4.1.4 Bestimmung der Kameramatrix

Die Kameramatrix K kann nach Bestimmung der ω-Matrix mithilfe der Cholesky Zerlegung bestimmt werden [7].

$$\omega = (KK^T)^{-1} \tag{4.20}$$

Die ist nur möglich, da die ω-Matrix positiv definit ist. Positive Definitheit einer Matrix kann durch das Berechnen der Eigenwerte der Matrix nachgewiesen werden. Sind diese Eigenwerte positiv, ist auch die Matrix positiv definit [16]. Durch das Invertieren des Ergebnisses der Cholesky Zerlegung wird die gesuchte Kameramatrix K ermittelt [15].

4.2 Initiale Rekonstruktion der Objektpunkte

Dieses Kapitel befasst sich mit der Rekonstruktion der Objektpunkte. Mit den lokalisierten 2D-Schachbretteckpunkten auf dem Sensor und der Kameramatrix können die Punkte im Raum berechnet werden. Die Berechnung erfolgt dabei über den Normalenvektor $\vec{n_0}$ der Ebene, der die Lage des Schachbretts beschreibt und dem Lösen eines nichtlinearen Gleichungssystem. Für die initiale Rekonstruktion sind die initial berechnete Kameramatrix, der Abstand der Schachbretteckpunkte zueinander, der initial berechnete Normalenvektor und die Sensorpunke notwendig.

4.2.1 Initiale Berechnung der Objektorientierung

Ziel der Berechnung ist es, den Normalenvektor n_0, der die Lage des Schachbretts im Raum beschreibt, zu bestimmen. In Abbildung 4.3 ist dieser Normalenvektor beispielhaft eingezeichnet. Hierfür werden zuerst die

Fluchtpunkte berechnet. Anhand dieser lassen sich zwei Richtungsvektoren $\overrightarrow{OF_v}$ und $\overrightarrow{OF_u}$ bestimmen, die senkrecht zu dem Normalenvektor $\vec{n}$ stehen.

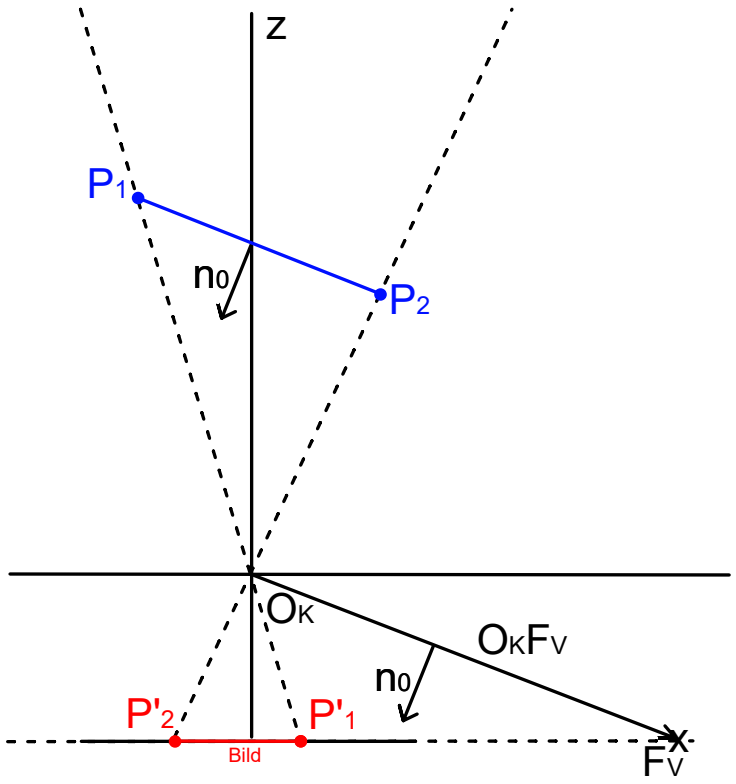


Abbildung 4.3: Lage der Objektpunkte

Für die Berechnung der Fluchtpunkte, die den Schnittpunkte der zueinander parallelen Schachbrettgeraden beschreibt, muss für jede Punktreihe eine Gerade bestimmt werden. Befinden sich die Punkte einer Punktreihe exakt auf einer Gerade, kann mit dem Kreuzprodukt zweier nebeneinanderliegender Punkte $P_1 = [\, p_1^1,\ p_1^2,\ 1\,]$ und $P_2 = [\, p_2^1,\ p_2^2,\ 1\,]$ der Normalenvektor $\vec{n} = (\, a\ b\ c\,)^T$ der $\mathbb{R}^3$-Ebene, welche von den Vektoren aufgespannt wird, bestimmt werden [9].

$$\begin{bmatrix} p_1^1 \\ p_1^2 \\ 1 \end{bmatrix} \times \begin{bmatrix} p_2^1 \\ p_2^2 \\ 1 \end{bmatrix} = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

Mit dem Normalenvektor $\vec{n_0}$ kann die $\mathbb{R}^3$-Ebene in der Form

$$a \cdot x + b \cdot y + c \cdot z = 0$$

dargestellt werden. Jeder projektive Punkt $P = [\, p_1^1,\ p_1^2,\ 1\,]$ liegt dann auf der Geraden, wenn die Gleichung $ap_1^1 + bp_1^2 + cz = 0$ erfüllt ist [9]. Um den Normalenvektor $\vec{n_0}$ zu berechnen, der zu allen homogenen Punkten $[\, P_1,\ P_2, ..,\ P_n\,]$ einer Punktreihe möglichst senkrecht steht, wird ein homogenes Gleichungssystem der Form

$$\begin{bmatrix} p_1^1 & p_1^2 & 1 \\ p_2^1 & p_2^2 & 1 \\ \vdots & \vdots & \vdots \\ p_n^1 & p_n^2 & 1 \end{bmatrix} \cdot \begin{bmatrix} a \\ b \\ c \end{bmatrix} = 0 \tag{4.21}$$

gebildet. Der gesuchte Normalenvektor $\vec{n_0}$ der projektiven Gerade wird mit einer SVD ermittelt und besteht aus den Werten der letzten Spalte der Matrix $V^T = (v_1\ v_2\ v_3\ ...\ v_n)$.

Der nächste Schritt befasst sich mit der Berechnung des Schnittpunkts der Geraden einer gleichen geometrischen Lage. Der Schnittpunkt zweier Geraden lässt sich ebenfalls mit dem Kreuzprodukt zweier Normalenvektoren bestimmen [9]. In diesem Fall gibt es jedoch ebenfalls mehr als zwei Geraden, weshalb sich der Schnittpunkt dieser Geraden auch mit dem Lösen durch die SVD ergibt.

$$\begin{bmatrix} a_1 & b_1 & c_1 \\ a_2 & b_2 & c_2 \\ \vdots & \vdots & \vdots \\ a_n & b_n & c_n \end{bmatrix} \cdot \begin{bmatrix} f_1 \\ f_2 \\ f_3 \end{bmatrix} = 0$$

Der Fluchtpunkt, der berechnet wurde, muss anschließend in die Standarddarstellung mit $F = (\frac{f_1}{f_3}, \frac{f_2}{f_3}, 1)$ gebracht werden. Abbildung 4.4 zeigt die Geraden und die zwei Fluchtpunkte, die sich aus der Berechnung ergeben.

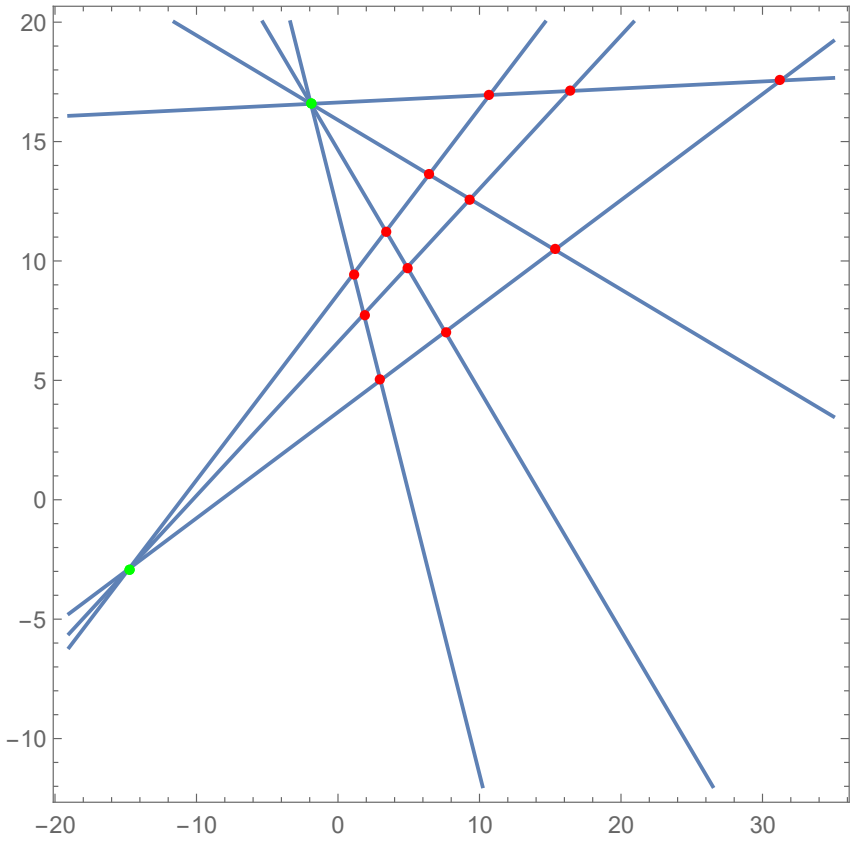


Abbildung 4.4: Geraden und Fluchtpunkte eines Schachbretts

Nach Bestimmung der Fluchtpunkte werden die Richtungsvektoren $\overrightarrow{O_k F_v}$ und $\overrightarrow{O_k F_u}$ vom optischen Zentrum zu den Fluchtpunkten berechnet. Die Koordinaten $F = (x, y)$ des Fluchtpunkts beschreiben den Fluchtpunkt im Sensorkoordinatensystem $K_s = (O_s; u, v)$. Damit ergeben sich für einen Fluchtpunkt, der auf der Sensorebene liegt, die Koordinaten

$$F = \begin{pmatrix} f_{11} \\ f_{22} \\ 0 \end{pmatrix}.$$

Der Punkt O_k, in dem sich das Projektionszentrum befindet, ergibt sich

aus dem Hauptpunkt $H = (h_1, h_2)$ und der Entfernung des Projektionszentrum zum Hauptpunkt f.

$$O_k = \begin{pmatrix} h_1 \\ h_2 \\ -f \end{pmatrix}$$

In Abbildung 4.3 ist die Ermittlung der Richtungsvektoren eingezeichnet.

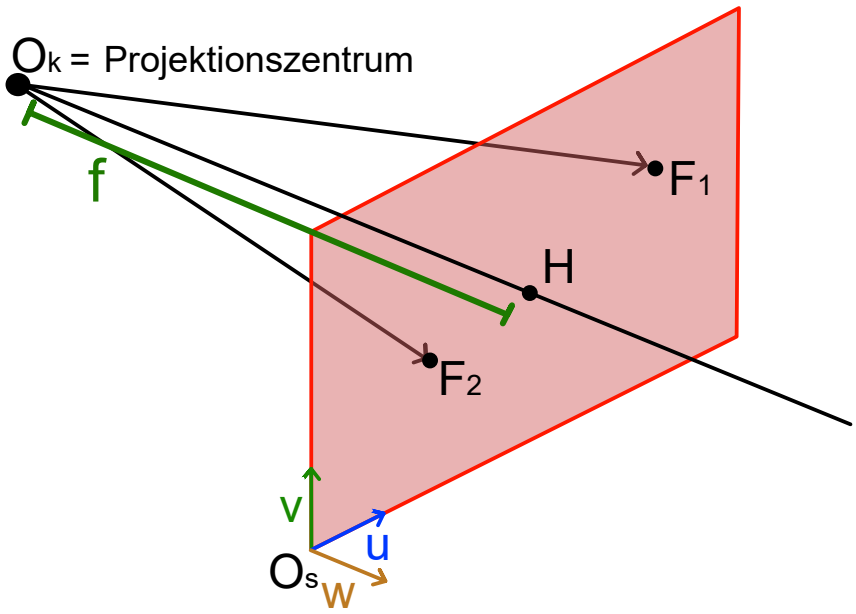


Abbildung 4.5: Ermittlung der Richtungsvektoren

Die Richtungsvektoren $\overrightarrow{O_kF_v}$ und $\overrightarrow{O_kF_u}$ können folgendermaßen berechnet werden:

$$\overrightarrow{O_kF_{v,u}} = \begin{pmatrix} f_1 \\ f_2 \\ 0 \end{pmatrix} - \begin{pmatrix} h_1 \\ h_2 \\ -f \end{pmatrix}.$$

Mit dem Kreuzprodukt der beiden Richtungsvektoren

$$\overrightarrow{O_kF_v} \times \overrightarrow{O_kF_v}$$

kann der Normalenvektor $\vec{n_0}$ berechnet werden. Auf der Abbildung 4.3 ist der erste Richtungsvektor $\overrightarrow{O_k F_v}$ eingezeichnet, der zweite Richtungsvektor zeigt nach oben und ist auf dem Bild nicht sichtbar. Zu erkennen ist der Normalenvektor $\vec{n_0}$, der senkrecht zu beiden Richtungsvektoren steht.

4.2.2 Rekonstruktion der Objektpunkte

Dieses Kapitel behandelt die Rekonstruktion der Objektpunkte durch das Lösen eines nichtlinearen Gleichungssystem. Für die Rekonstruktion werden folgende Parameter benötigt

- der Abstand zweier benachbarter Schachbretteckpunkte d
- der Normalenvektor $\vec{n_0}$
- die lokalisierten Sensorpunkte

Der Koordinatenursprung liegt im Projektionszentrum O_K.

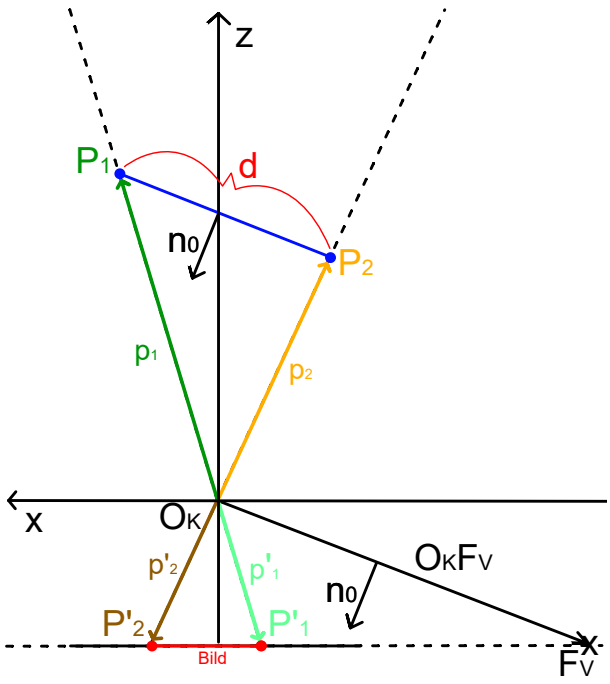


Abbildung 4.6: Lage der Objektpunkte (vgl. [15])

In Abbildung 4.6 kann beobachtet werden, dass p'_1 und p'_2 die Ortsvektoren der Sensorpunkte P'_1 und P'_2 und p_1 und p_2 die Ortsvektoren der Objektpunkte P_1 und P_2 sind. Zusätzlich kann erkannt werden, dass die Vektoren Vielfache voneinander sind.

$$s \cdot p'_1 = p_1 \tag{4.22}$$

$$t \cdot p'_2 = p_2 \tag{4.23}$$

Die Koordinaten der Punkte P_1 und P_2 können mit

$$O + p_1 = P_1 \tag{4.24}$$

$$O + p_2 = P_2 \tag{4.25}$$

berechnet werden. Der Abstand d der Punkte P_1 und P_2 ist die gegebene Gitterkonstante eines Schachbretteckpunktes zum nächsten.

$$d = \|P_1 - P_2\|$$

Werden die Gleichungen 4.25 in die obige Gleichung eingesetzt, ergibt sich

$$\begin{aligned} \|(O + p_1) - (O + p_2)\| &= d \\ \|p_1 - p_2\| &= d. \end{aligned}$$

Wird Gleichung 4.23 eingesetzt, ergibt dies die nichtlineare Gleichung

$$\|(s \cdot p'_1) - (t \cdot p'_2)\| = d$$

mit s und t als unbekannte Parameter. Zusätzlich sollte noch gelten, dass der Verbindungsvektor $\overrightarrow{P_2P_1}$ orthogonal zum Normalenvektor der Ebene

sein sollte.

$$\begin{aligned} n \cdot \overrightarrow{P_2 P_1} &= 0 \\ n \cdot (P_1 - P_2) &= 0 \\ n \cdot (s \cdot p'_1 - t \cdot p'_2) &= 0 \end{aligned}$$

Für die Rekonstruktion der Objektpunkte sollten also zwei Bedingungen

$$\|(s \cdot p'_1) - (t \cdot p'_2)\| = d \tag{4.26}$$

$$n \cdot (s \cdot p_1^1 - t \cdot p_2^1) = 0z \tag{4.27}$$

erfüllt werden. Dieses nichtlineare Gleichungssystem hat zwei Gleichungen mit zwei Unbekannten, bei der die Lösung die Werte s und t enthält. Für jedes Punktepaar können mit dem Ergebnis die zwei Punkte P_1 und P_2 berechnet werden.

$$P_1 = \begin{pmatrix} s \cdot p'_{1.1} \\ s \cdot p'_{1.2} \\ s \cdot p'_{1.3} \end{pmatrix} \qquad P_2 = \begin{pmatrix} t \cdot p'_{2.1} \\ t \cdot p'_{2.2} \\ t \cdot p'_{2.3} \end{pmatrix}$$

Um zu prüfen, wie exakt die ermittelten Kameraparameter sind und wie gut die Objektpunkte rekonstruiert wurden, kann der Reprojektionsfehler berechnet werden, der entsteht, wenn die rekonstruierten Objektpunkte mit der ermittelten Kameramatrix wieder auf den Sensor projiziert werden, nicht den ursprünglich lokalisierten Sensorpunkten entsprechen. In Kapitel 4.4 wird näher darauf eingegangen.

4.3 Initiale Berechnung der extrinsischen Parameter

Nach der initialen Berechnung der intrinschen Kameraparameter und der initialen Rekonstruktion der Objektpunkte können die extrinsischen Kameraparameter berechnet werden. Diese beschreiben die Transformation von Weltkoordinaten in Kamerakoordinaten in Form einer Drehmatrix D und eines Translationsvektors t. Es müssen dabei sechs extrinsische Parameter berechnet werden, welche zur Übersicht aufgeführt werden.

- D_x: Rotation um die x-Achse mit Winkel α
- D_y: Rotation um die y-Achse mit Winkel β
- D_z: Rotation um die z-Achse mit Winkel γ
- t_x: Translation in x-Richtung
- t_y: Translation in y-Richtung
- t_z: Translation in z-Richtung

Ein Schachbrettgitter $S_{m\,\mathrm{x}\,n}$ mit mxn Punkten, beispielsweise ein 2x2 Quadrat wie in 4.28 dargestellt, befindet sich im Ursprung des Weltkoordinatensystems und wird zuerst einer affinen Transformation mit einer Drehung

$$D = D_{\hat{x},\alpha} \circ D_{\hat{y},\beta} \circ D_{\hat{z},\gamma}$$

und einem Translationsvektor

$$t = \begin{pmatrix} t_x \\ t_y \\ t_z \\ 0 \end{pmatrix}$$

unterzogen. Die Gitterkonstante d gibt den Abstand zweier benachbarter Punkte an.

$$S_{m\,\mathrm{x}\,n} = \begin{pmatrix} \begin{pmatrix} 0 & 0 & 0 & 1 \end{pmatrix}^T & \begin{pmatrix} d & 0 & 0 & 1 \end{pmatrix}^T \\ \begin{pmatrix} 0 & d & 0 & 1 \end{pmatrix}^T & \begin{pmatrix} d & d & 0 & 1 \end{pmatrix}^T \end{pmatrix} \tag{4.28}$$

Diese Drehung und Translation kann durch die Gleichung

$$G_{m\,\mathrm{x}\,n} = (D(\alpha, \beta, \gamma) \cdot S_{m\,\mathrm{x}\,n}) + t \tag{4.29}$$

veranschaulicht werden. Wird der erste Punkt $s_{1\mathrm{x}1} = (0,\, 0,\, 0,\, 1)^T$ gedreht, verändert sich dieser Punkt nicht. Erst mit einer Translation wird dieser Punkt vom Ursprung des Weltkoordinatensystem verschoben. Damit gibt der erste Punkt $s_{1\mathrm{x}1}$ eines Schachbretts auch den gesuchte Translationsvektor t an. Abbildung 4.7 zeigt den Translationsvektor $t = \begin{pmatrix} 1 & 0 & 1 & 1 \end{pmatrix}$.

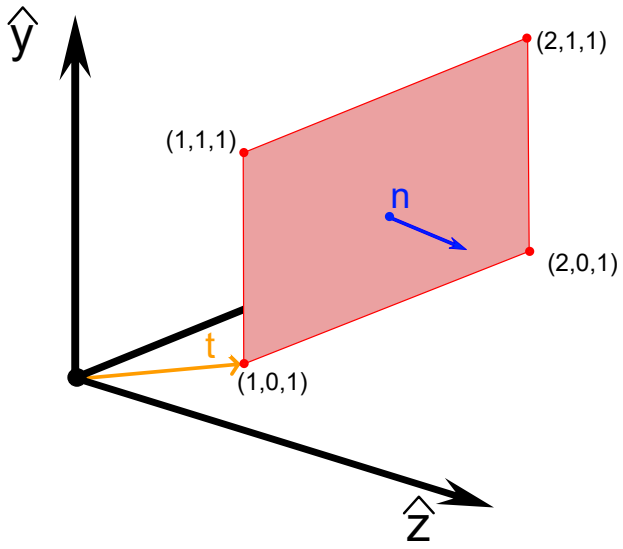


Abbildung 4.7: Translationsvektor t

Für die Berechnung der Euler-Winkel wird die Drehung D im Folgenden

genauer beschrieben.

$$D = D_{\hat{x},\alpha} \circ D_{\hat{y},\beta} \circ D_{\hat{z},\gamma}$$

Die Schachbrettebene wird zuerst um die $\hat{z}$-Achse mit dem Euler-Winkel γ gedreht, wobei sich der Normalenvektor $\vec{n_0}$ der Ebene nicht verändert. Als nächstes wird um die $\hat{y}$-Achse mit dem Winkel β gedreht. Dabei ändert sich der Normalenvektor der Ebene und liegt jetzt in der x-z-Ebene wie in Abbbildung 4.8 eingezeichnet. Wird um die x-Achse mit dem Winkel α gedreht, dreht sich der Normalenvektor ebenfalls um die x-Achse. Nachfolgend werden die Drehungen unter Beobachtungen des Vektors n beschrieben.

$$n \overset{D_{(\hat{z},\gamma)}}{\longmapsto} n \overset{D_{(\hat{y},\beta)}}{\longmapsto} n' \overset{D_{(\hat{x},\alpha)}}{\longmapsto} n''$$

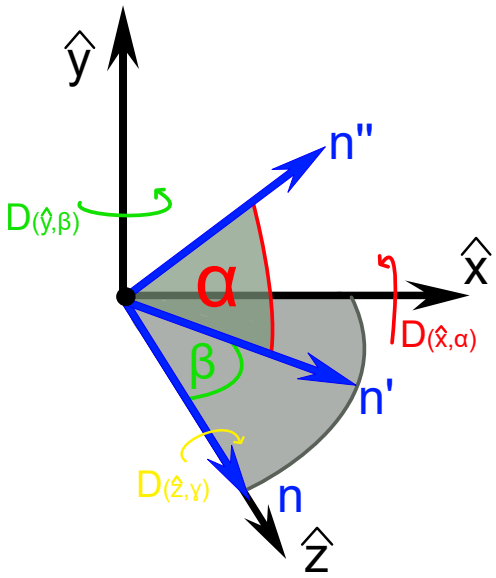


Abbildung 4.8: Drehung D mit dem Normalenvektor $\vec{n}$

Der Vektor n'' entspricht dem ermittelten Normalenvektor n_0 der Ebene. Mithilfe des Normalenvektors können die beiden Winkel α und β berechnet werden. Hierfür wird der Normalenvektor n_0 zuerst in die y-z-Ebene

projiziert. n_0' in Abbildung 4.9 bezeichnet die Projektion von n_0 auf die y-z-Ebene mit $(0\ n_2\ n_3)^T$.

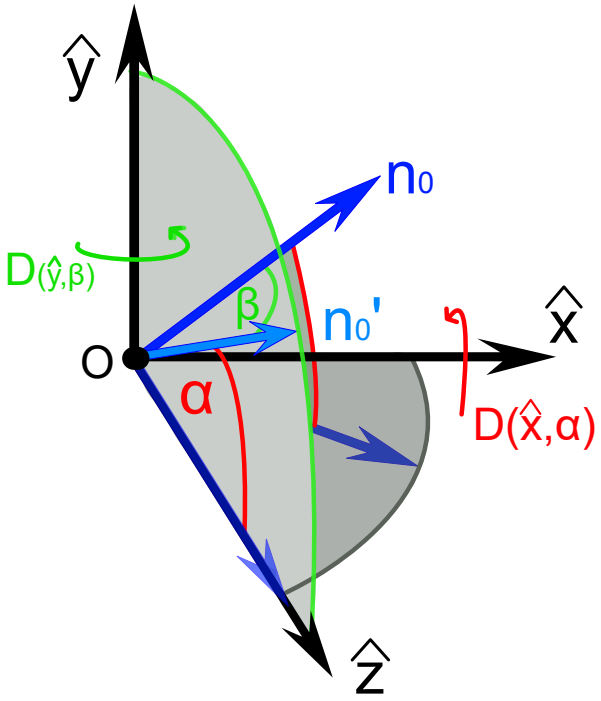


Abbildung 4.9: Projektion von n_0 in die y-z-Ebene

Der Winkel α kann durch die Berechnung zwischen n' und der $\hat{z}$-Achse bestimmt werden.

$$\alpha = \measuredangle(\hat{z}, n') = \arccos\left(\frac{n' \cdot \hat{z}}{\|n'\|\,\|\hat{z}\|}\right) \tag{4.30}$$

Der Winkel β ergibt sich aus der Berechnung $\beta = \measuredangle(n', n)$.

$$\beta = \measuredangle(n, n') = \arccos\left(\frac{n' \cdot n}{\|n'\|\,\|n\|}\right)$$

Wie in dem Ausdruck 4.30 bestätigt wird, ändert sich der Normalenvektor bei einer Drehung $D_{(\hat{z},\gamma)}$ nicht. Deswegen muss dieser Winkel über einen anderen Weg bestimmt werden. Für den Winkel γ wird die Gleichung

$$G_{m\,\mathrm{x}\,n} = (D(\alpha, \beta) \cdot S_{m\,\mathrm{x}\,n}) +$$

auf zwei benachbarte Schachbrettpunkte S_A und S_B angewandt. Die errechneten Punkte G_A, G_B und die rekonstruierten Punkte R_A, R_B unterscheiden sich in der γ-Drehung. Dahingehend ergibt sich der Winkel γ mit der Berechnung $\gamma = \measuredangle(\overrightarrow{G_A G_B}, \overrightarrow{R_A R_B})$.

$$\gamma = \measuredangle(\overrightarrow{G_A G_B}, \overrightarrow{R_A R_B}) = \arccos\left(\frac{\overrightarrow{R_A R_B} \cdot \overrightarrow{G_A G_B}}{\left\|\overrightarrow{R_A R_B}\right\| \left\|\overrightarrow{G_A G_B}\right\|}\right)$$

Die hier aufgezeigten Berechnungen beschäftigen sich nicht mit den unterschiedlichen Fallunterscheidungen, wenn beispielsweise die Winkel $\geq 90°$ betragen, weshalb für die Anwendung Winkel unter $90°$ gewählt werden sollten. Damit wurden auch die extrinsischen Kameraparameter initial bestimmt. Diese sind nützlich um im nächsten Schritt einen Bündelblockausgleich durchzuführen, der die initialen Werte final berechnet.

4.4 Bündelblockausgleich

Bisher wurden die Kameraparameter und die Objektpunkte initial berechnet. Dabei wird davon ausgegangen, dass verschiedene Faktoren die Berechnung beeinflussen. Es kann zum Beispiel vorkommen, dass die Schachbretteckpunkte aufgrund eines Sensorrauschens oder einer Über- oder Unterbelichtung nicht richtig abgebildet und lokalisiert wurden und folglich auch die Berechnung mit fehlerhaften detektierten Schachbretteckpunkten stattfindet. Daraus ergibt sich eine nicht korrekte Berechnung der Kameramatrix und die Rekonstruktion der Objektpunkte, die sowohl von dieser Kameramatrix, als auch von den lokalisierten Schachbretteckpunkten abhängig ist, erfolgt ungenau.

$$RP_{SP} = K \cdot RP_{OP}$$

Werden diese fehlerhaften Rekonstruktionspunkte RP_{OP} mit der fehlerhaften Kameramatrix reprojiziert, entstehen reprojizierte Sensorpunkte. Diese stimmen nicht mehr mit den ursprünglich lokalisierten Sensorpunkten überein und es entsteht ein Abstand zwischen diesen beiden Punkten. Abbildung 4.10 zeigt ein Bild, das Reprojektionsfehler enthält. Die roten Punkte sind die detektierten Schachbretteckpunkte L_{SP} einer projizierten Schachbrettebene, mithilfe derer die Kameramatrix berechnet wurde. Die grünen Punkte sind jene, die entstanden sind, als die rekonstruierten Objektpunkte reprojiziert wurden.

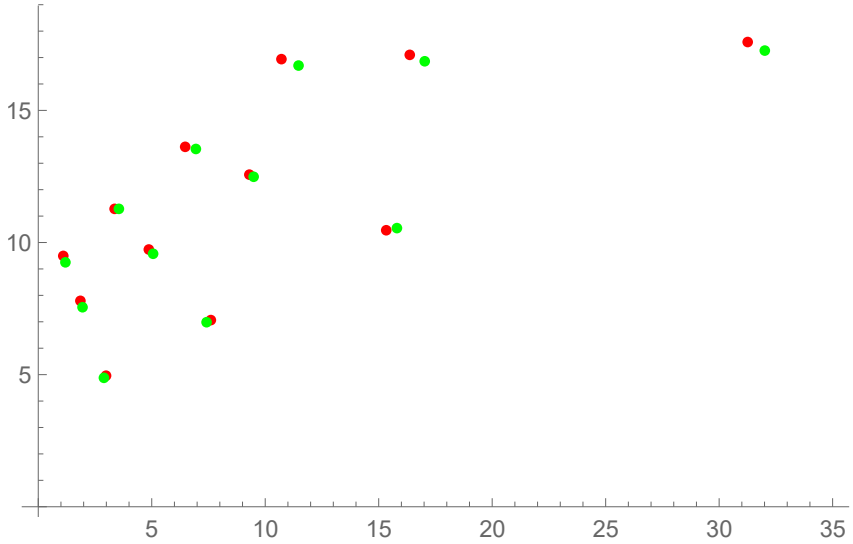


Abbildung 4.10: Bild mit Reprojektionsfehler

Der Reprojektionsfehler ist die Summe der Abstände d dieser Punkte zueinander im Quadrat [7].

$$R = \sum_{m \times n} \| L_{SP}^{m \times n} - RP_{SP}^{m \times n} \|^2$$

Ist R gering, war die Kalibrierung erfolgreich und von guter Qualität. Je größer dieser Fehler wird, desto schlechter war die Ermittlung der initialen Werte. Dieser Reprojektionsfehler kann durch die Anpassung der initialen Werte minimiert werden, sodass der Reprojektionsfehler möglichst gegen

Null geht. Für den Algorithmus gegebene Parameter sind:

- die Gitterkonstante d
- die Zeilen- m und Spaltenanzahl n des Schachbretts
- die Bildpunkte auf dem Sensor b_{jk}

Es wird davon ausgegangen, dass der $skew$-Faktor, der nur in seltenen Fällen abweicht, Null beträgt und f_x und f_y denselben Wert haben. Gesucht wird

- die Projektionsmatrix P, mit f, h_1 und h_2

$$P = \begin{bmatrix} f & 0 & h_1 & 0 \\ 0 & f & h_2 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

- die Drehwinkel α, β und γ
- der Translationsvektor $t = (t_1,\ t_2,\ t_3,\ 0)^T$.

Diese neun voneinander abhängigen unbekannten Parameter müssen optimiert werden.

$$min \sum_{jk} \|g_{jk} - b_{jk}\|^2$$
$$min \sum_{jk} \|P(f, h_1, h_2) \cdot ((D(\alpha, \beta, \gamma) \cdot s_{m\,\mathrm{x}\,n}) + t) - b_{jk}\|^2$$

Mit dem Levenberg-Marquardt-Algorithmus, einem Optimierungsalgorithmus zur Lösung nichtlinearer Ausgleichsprobleme, lassen sich für die gesuchten Parameter, die mit Startwerten initialisiert werden, Ausgleichslösungen bestimmen, sodass der Reprojektionsfehler minimal ist [17].

5 Algorithmen der Anwendung

Im Rahmen dieser Thesisarbeit wurde eine modularisierte Anwendung in *Mathematica* geschrieben, in der die Berechnungen der Kamerakalibrierung und der Bündelblockausgleich implementiert wurden. Ferner wurde geprüft, ob sich die Berechnungen auch in Python übertragen lassen. Dies ermöglicht es, eine Anwendung zu gestalten, die nutzerfreundlich als weiterentwickelt werden kann.
Für die Implementierung der Anwendung ist es wichtig, dass einige Funktionen wie die `Singulärwertszerlegung` oder die `Choleskyzerlegung` implementiert sind. Besonders wichtig hierbei ist auch der Umgang mit Matrizen und deren Berechnungen. Die Funktionalität, Ergebnisse graphisch darzustellen, sollte bestenfalls ebenfalls implementiert sein.

5.1 Einführung in *Mathematica*

Geht es um technische Berechnungen in Forschung und Entwicklung, ist Mathematica eines der meistgenutzten Systeme [18]. Das System basiert auf der *Wolfram Language*, welche eine symbolischen Programmiersprache ist. Dies bedeutet, Berechnungen erfolgen nach Möglichkeit exakt, wie z. B. die Berechnung von $\sqrt{2}$ symbolisch und nicht nummerisch mit $\sqrt{2} = 1.41421356237$ dargestellt wird. Bei einer nummerischen Darstellung kann es zu minimalen Rundungsfehlern kommen. *Mathematica* hat demzufolge nicht nur den Vorteil einer exakten Arithmetik, sondern hat auch

die notwendigen Algorithmen zur Kamerakalibrierung implementiert und kann auch Gleichungen lösen wie etwa über- oder unterbestimmte, nichtlineare und lineare Gleichungssysteme. Zudem besitzt *Mathematica* eine ausgezeichnete Dokumentation, in der Funktionen nachgeschlagen werden können. Die Dokumentation liefert Beispiele und erklärt, wie die Algorithmen implementiert wurden [18].

> „Wolfram Notebooks sind strukturierte interaktive Dokumente, die Text, Graphiken, Ton, Berechnungen, Typeset-Ausdrücke und Elemente von Benutzeroberflächen enthalten können.“ [19]

Notebooks haben die Dateiendung `.nb` und es kann, wie das Zitat erwähnt, jede Zeile interaktiv ausgeführt werden und dabei Grafiken, Berechnung etc. erstellen. Um jedoch alle Berechnungen der Kamerakalibrierung zu implementieren, müssen in einem Notebook viele Funktionen geschrieben werden, weshalb ein einziges Notebook zur Unübersichtlichkeit und Unstrukturiertheit neigt. Deswegen wurde im Rahmen der Bachelorthesis von Katharina Ußling die Nutzung einer Entwicklungsumgebung (IDE) getestet, die Module (Packages) und Notebooks miteinander vereint und eine modularisierte Anwendung erstellt, die übersichtlicher gestaltet ist. Hierbei kommt ein Plugin von *Wolfram* für die IDE Eclipse zum Einsatz.
Der Aufbau einer Applikation, die mithilfe dieses Plugins erstellt wurde, wird in der Bachelorarbeit von Katharina Ußling genauer erläutert [20]. In ihrem Prinzip besteht eine *Mathematica*-Applikation aus einem Haupt-Notebook, in der die `main`-Methode, die den Startpunkt der Anwendung markiert, ausgeführt wird. Die `main`-Methode befindet sich wiederum in einem Haupt-Modul, dass in einer Quelldatei, die mit der Dateiendung `.m` markiert ist, geschrieben wurde. In diesem Hauptmodul können weitere Quelldateien geladen und damit die Funktionen, die in dieser Quelldatei

enthalten sind, verwendet werden. Das grundlegende Konzept von Mathematica baut auf `Listen`, die durch geschweifte Klammern erzeugt werden und mehrdimensional aufgebaut werden können, auf: $\{\{...\},\{...\}\}$. Wichtig für das Verständnis der Implementierung ist die Tatsache, dass die Indizierung in *Mathematica* bei 1 beginnt, anders als bei anderen technischen Programmiersprachen.

5.2 Die Anwendung in *Mathematica*

In *Eclipse* befindet sich das Notebook `Kamerakalibrierung.nb`, welche die Anwendung startCalibration [] mit einigen Eingabeparametern startet.

```
Winkelliste = {{9, 12, 18, {1000, 500, 4000}}, {23, 34,
    3, {0, 0, 2000}}, {12, 4, 13, {0, 0, 3000}}};
f = 9; H = {12, 18}; (* in millimeters*)
Gitterkonstante = 25; (* in millimeter*)
Zeilen = 3; Spalten = 4;
pixelPitch = 3.75; (*camera model: canon eos 6D in millimeter*)
(*"Ungestoert" gibt die Liste der Punkte an,die nicht gestoert \
wurde,"Gestoert" gibt die Liste der Punkte an,die gestoert wurden*)
Quadrat = "Gestoert";
startCalibration[Winkelliste, Gitterkonstante, Zeilen, Spalten,
  Quadrat, pixelPitch, f, H];
```

Die Eingaben sind dabei

- eine Liste mit Informationen zur affinen Transformation des Schachbretts
  - die Drehwinkel α, β und γ
  - der Translationsvektor $t = (t_1, t_2, t_3)^T$, der die Verschiebung in x, y und z-Richtung in Millimeter angibt
- Informationen der Projektionsmatrix

  - die Brennweite f in Millimeter
  - der Hauptpunkt $H(h_1, h_2)$ in Millimeter
  - der Pixelpitch in μm, der den Abstand zweier benachbarter Pixel bezeichnet

- die Gitterkonstante d: der Abstand zweier Schachbrettpunkte zueinander in Millimeter
- die Zeilen- und Spaltenanzahl mxn des Schachbretts

Für die Modellrechnung wird dabei das Modell einer Lochkamera in Positivlage herangezogen.

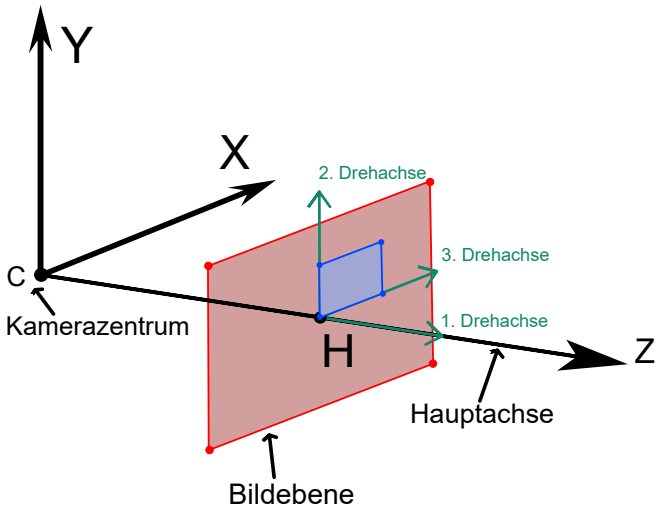


Abbildung 5.1: Lochkameramodell für die Modellierung

5.3 Projektion des Schachbretts auf die Bildebene

Begonnen wird mit Erstellung der Projektionsbilder, um aus den projizierten Sensorpunkten die Kamerakalibrierung durchzuführen. Um das Schachbrett in unterschiedlichen Lagen zu projizieren, wird ein Urschachbrett, das sich im Ursprung des Weltkoordinatensystems befindet, generiert und einer

affinen Rotation und Translation unterzogen und anschließend projiziert. Abbildung 5.2 zeigt die Vorgehensweise des Algorithmus.

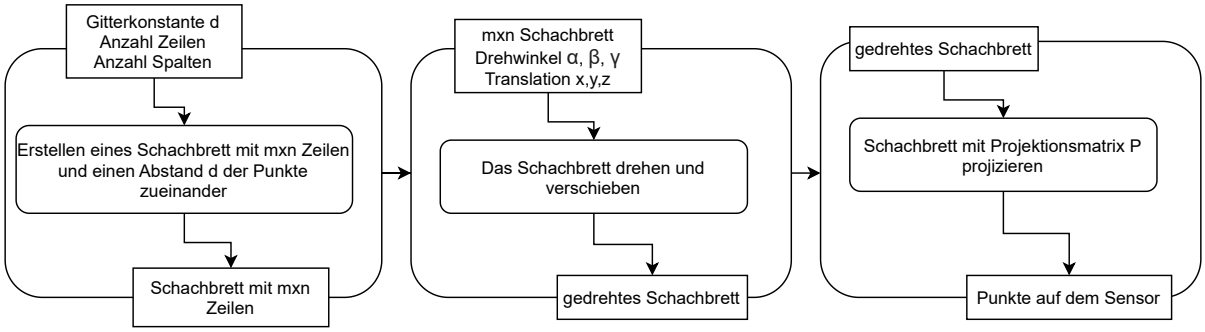


Abbildung 5.2: Ablauf bei der Projektion der Objektpunkte

Zu Beginn wird ein Schachbrett S mit $m \times n$ Zeilen generiert, bei dem die Gitterkonstante d den Abstand zweier benachbarter Punkte angibt. Um die Spalten und Zeilen einer Punktreihe eindeutig identifizieren zu können, wird eine mehrdimensionale Liste erstellt. Im Folgenden wurde beispielhaft ein 2 x 2 Schachbrett mit $d = 25\,mm$ generiert. Die in einer Liste S gespeicherten Sensorpunkte entsprechen einer Punktreihe einer Zeile. Durch das Transponieren kann die Spalte einer Punktreihe gespeichert werden.

```
1 S = {{{0,0,0,1},{25,0,0,1}},{{0,25,0,1},{25,25,0,1}}}
2 S_transponiert = {{{0,0,0,1},{0,25,0,1}},{{25,0,0,1},{25,25,0,1}}}
```

Anschließend wird dieses Schachbrett um drei Drehachsen, wie in Abbildung 5.1 eingezeichnet, mit einer Drehmatrix $D = D_{\hat{x},\alpha} \circ D_{\hat{y},\beta} \circ D_{\hat{z},\gamma}$ gedreht und mit einem Translationsvektor $t = (t_x \; t_y \; t_z)^T$ verschoben. Veranschaulicht wird dies durch die Gleichung

$$G_{m \times n} = (D(\alpha, \beta, \gamma) \cdot S_{m \times n}) + t.$$

Das Schachbrett $G_{m \times n}$, das sich nun in einer bestimmten Lage im Raum befindet, kann mit einer Projektionsmatrix P auf die Bildebene projiziert werden. P entsteht aus den Nutzereingaben für f und H. Die in mm

angegebenen Parameter müssen dafür vorerst in Pixelkoordinaten umgerechnet werden.

$$\begin{aligned} 1_{Pixel} &= \frac{1\ mm \cdot 1000}{Pixelpitch \frac{\mu m}{px}} \\ &= \frac{1000\ \mu m}{Pixelpitch\ \frac{\mu m}{px}} \\ &= \frac{1000}{Pixelpitch\, px} \end{aligned}$$

Die Umrechnung der Pixelkoordinaten in mm kann mit der Umstellung der Formel berechnet werden.

$$1_{mm} = \frac{1\, Pixel \cdot Pixelpitch\, \frac{\mu}{px}}{1000}$$

Die Projektionsmatrix P ist damit folgende:

$$P = \begin{bmatrix} f_{Pixel} & 0 & h_{1Pixel} & 0 \\ 0 & f_{Pixel} & h_{2Pixel} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{5.1}$$

Die Bildpunkte $B_{i\,\mathrm{x}\,j}$ in Pixelkoordinaten ergeben sich durch die Projektion der Objektpunkte auf die Bildebene mit der Projektionsmatrix P.

$$B_{i\,\mathrm{x}\,j} = P \cdot G_{m\,\mathrm{x}\,n} \tag{5.2}$$

Abbildung 5.3 veranschaulicht ein realistisches Projektionsbild eines Schachbrettes. Dabei werden die Werte einer `CANON 6D`-Kamera herangezogen, bei der der Pixelpitch $pp = 3.75\,\mu m$ beträgt und der Hauptpunkt des Vollformatssensors bei $H(12\,mm, 18\,mm)$ liegt. Für das Objektiv wur-

de eine Brennweite $f = 9\,mm$ gewählt und die Gitterkonstante beträgt $d = 25\,mm$.

$$B_{i\,\mathrm{x}\,j} = P(9,\,12,\,18) \cdot (D(6°,\,9°,\,13°) \cdot S_{3\mathrm{x}4} + (1000,\,500,\,4000)^T) \quad (5.3)$$

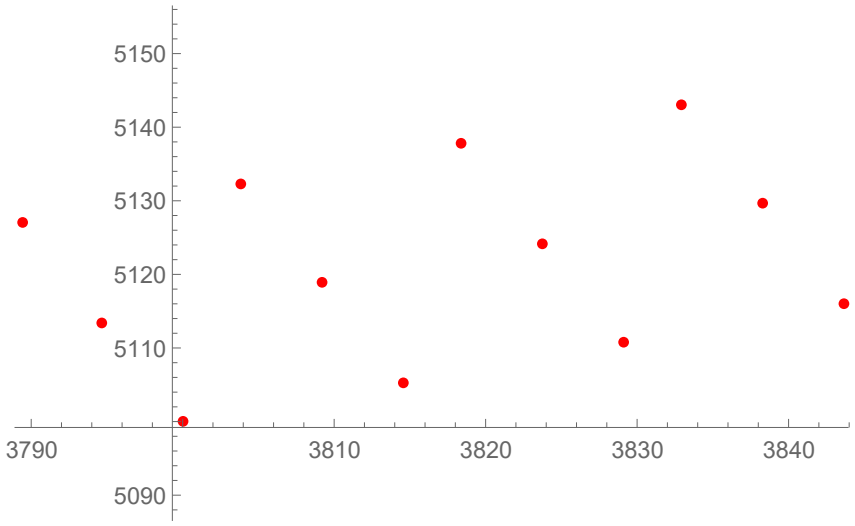


Abbildung 5.3: Projiziertes Schachbrett aus den Werten der Gleichung 5.3

Dies ergibt die homogenen Bildpunkte:

$$\text{Sensorpunkte}: \begin{pmatrix} \begin{pmatrix} 3800. \\ 5100. \\ 1. \end{pmatrix} & \begin{pmatrix} 3814.52 \\ 5105.32 \\ 1. \end{pmatrix} & \begin{pmatrix} 3829.06 \\ 5110.66 \\ 1. \end{pmatrix} & \begin{pmatrix} 3843.64 \\ 5116. \\ 1. \end{pmatrix} \\ \begin{pmatrix} 3794.68 \\ 5113.52 \\ 1. \end{pmatrix} & \begin{pmatrix} 3809.17 \\ 5118.85 \\ 1. \end{pmatrix} & \begin{pmatrix} 3823.69 \\ 5124.19 \\ 1. \end{pmatrix} & \begin{pmatrix} 3838.24 \\ 5129.54 \\ 1. \end{pmatrix} \\ \begin{pmatrix} 3789.37 \\ 5127.01 \\ 1. \end{pmatrix} & \begin{pmatrix} 3803.84 \\ 5132.34 \\ 1. \end{pmatrix} & \begin{pmatrix} 3818.34 \\ 5137.69 \\ 1. \end{pmatrix} & \begin{pmatrix} 3832.86 \\ 5143.04 \\ 1. \end{pmatrix} \end{pmatrix}$$

Die Funktion ProjectObjectpoints[], die diese Berechnung ausführt, erstellt eine

`Association-Liste`, welche `Key-Value` Paare speichert, mit den Parametern:

$$< |\text{„Sensorpunkte“} \rightarrow \text{Sensorpunkteliste},$$
$$\text{„Objektpunkte“} \rightarrow \text{Objektpunkteliste},$$
$$\text{„Indizes“} \rightarrow \text{Urgitter}| >;$$

5.4 Initiale Berechnung der Kameramatrix

Wurden drei verschiedene Schachbrettbilder erstellt, kann anhand dieser Sensorpunkte die initiale Kameramatrix berechnet werden. Dafür muss zuerst für jede projizierte Schachbrettebene eine Homografiematrix berechnet werden. Mit diesen Homografiematrizen lässt sich das Bild des absoluten Kegelschnitts und daraus die Kameramatrix K bestimmen.

5.4.1 Berechnung der Homografien

Eine Homografie beschreibt eine projektive Transformation des Schachbrettgitters $S_{m \times n}$ auf die Sensorpunkte, die in einer Ebene liegen. Zur Erinnerung sind die Gleichungen eines Punktes für die Homografiematrix noch einmal aufgelistet.

$$\begin{aligned} h_{11} \cdot P_x + h_{12} \cdot P_y + h_{13} - (h_{31} \cdot P_x + h_{32} \cdot P_y + h_{33}) \cdot P'_x &= 0 \\ h_{21} \cdot P_x + h_{22} \cdot P_y + h_{23} - (h_{31} \cdot P_x + h_{32} \cdot P_y + h_{33}) \cdot P'_y &= 0 \end{aligned}$$

Die in der `Acossiation`-Liste gespeicherten Werte für die Sensorpunkte und Indizes werden für diese Berechnung benötigt. Der Algorithmus stellt für jeden Punkt die beiden Gleichungen auf und fügt diese einer Koeffizientenmatrix hinzu.

```
pointPx' = projizierteObjektpunkte[[i, j, 1]];
pointPy' = projizierteObjektpunkte[[i, j, 2]];
pointPx = Indizes[[i, j, 1]];
pointPy = Indizes[[i, j, 2]];
row1 = {pointPx, pointPy, 1, 0, 0, 0, -(pointPx*pointPx'), -(pointPy*
    pointPx'), -(pointPx')};
row2 = {0, 0, 0, pointPx, pointRCc, 1, -(pointPx*pointPy'), -(pointPy*
    pointPy'), -(pointPy')};
AppendTo[HCoefficientMatrix, row1];
AppendTo[HCoefficientMatrix, row2];
```

Damit ergibt sich für ein Schachbrettquadrat eine Koeffizientengleichung mit neun Gleichungen und acht Unbekannten. Diese Koeffizientenmatrix wird mit einer SVD SingularValueDecomposition[] zerlegt. Die gesuchte Lösungsmatrix besteht hierbei aus den Werten der letzten Spalte der Matrix $V^T = (v_1\ v_2\ v_3\ ...\ v_n)$.

```
VMatrix = Last[SingularValueDecomposition[N[CoefficientMatrix]]];
{h11, h12, h13, h21, h22, h23, h31, h32, h33} =  Last[Transpose[VMatrix]];
Loesungsmatrix = {{h11, h12, h13}, {h21, h22, h23}, {h31, h32, h33}};
```

5.4.2 Bestimmung des absoluten Kegelschnitts

Im weiteren Verlauf zur Bestimmung der Kameramatrix wird die ω-Matrix für das Bild des absoluten Kegelschnitts berechnet. Dabei muss pro Homografiematrix $h = \begin{bmatrix} h1 & h2 & h3 \end{bmatrix}$ die Formel

$$h_1^T \omega h_2 = 0 \tag{5.4}$$

$$h_1^T \omega h_1 - h_2^T \omega h_2 = 0 \tag{5.5}$$

gelten. Für jede Homografiematrix werden zwei Gleichungen aufgestellt und einer Koeffizientenmatrix angehängt.

```
row1 = {h11 * h12, h11 * h22 + h21 * h12, h11 * h32 + h31 * h12, h21 * h22,
    h21 * h32 + h31 * h22, h31 * h32};
```

```
row2 = {h11^2 - h12^2, h11 * h21 + h21 * h11 - h12 * h22 - h22 * h12, h11 *
    h31 + h31 * h11 - h12 * h32 - h32 * h12, h21^2 - h22^2,
      h21 * h31 + h31 * h21 - h22 * h32 - h32 * h22, h31^2 - h32^2};
AppendTo[Koeffizientenmatrix, row1];
AppendTo[Koeffizientenmatrix, row2];
```

Mit der SVD kann das Gleichungssystem gelöst werden, wobei in der Lösung die Werte der ω-Matrix entnommen werden kann. Mit der Ermittlung der ω-Matrix lässt sich die Kameramatrix mit der Invertierung der Cholesky Zerlegung berechnen.

$$\omega = (KK^T)^{-1}$$

```
CameraMatrix = Inverse[CholeskyDecomposition[omegaMatrix]];
```

Um zu zeigen, dass die ω-Matrix positiv definit ist, können die Eigenwerte der ω-Matrix mit der Funktion Eigenvalues[] berechnet werden. Sind alle Eigenwerte der Matrix positiv, ist auch die Matrix positiv definit. Die intrinsischen Kameraparameter können nun aus der Kameramatrix entnommen werden. Wird die Kameramatrix mit den Werten der `CANON 6D` ermittelt, entspricht die errechnete Kameramatrix der Projektionsmatrix.

Die ermittelte Kameramatrix in mm: $\begin{pmatrix} 9. & 0. & 12. \\ 0. & 9. & 18. \\ 0. & 0. & 1. \end{pmatrix}$

5.4.3 Beobachtungen der Algorithmen unter verschiedenen Bedingungen

Durch die Modellierung sind Störungen, die bei der Lokalisierung und Typisierung der Schachbrettpunkte auftreten, nicht enthalten. Da auch die Kameraparameter bekannt sind, können verschiedene Eingaben getätigt werden, um den Algorithmus unter verschiedenen Bedingungen zu beobachten.

- Was passiert, wenn z. B . drei von fünf Sensorpunkten identisch sind?

Sind drei von fünf Sensorpunkten identisch, gibt es dennoch drei unterschiedliche Homografien, wodurch sich die Kameramatrix trotzdem korrekt ermittelt lässt.

- Was passiert, wenn eines der drei Schachbretter nur um $90°$ gedreht und nicht verschoben wurde?

Wird ein Schachbrett um $D_{(90,x)}$ oder um $D_{(90,y)}$ gedreht und um die Brennweite $f = 9\,mm$ in z-Richtung verschoben, entsteht das Bildviereck:

$$\text{Sensorpunkte} : \left(\begin{pmatrix} 3200. \\ 4800. \\ 1. \end{pmatrix} \begin{pmatrix} 3800. \\ 4800. \\ 1. \end{pmatrix} \begin{pmatrix} 3200. \\ 4800. \\ 1. \end{pmatrix} \begin{pmatrix} 3680. \\ 4800. \\ 1. \end{pmatrix} \right)$$

Anhand der Bildpunkte lässt sich erkennen, dass das Viereck ausgeartet und damit das 3. Axiom nicht erfüllt ist. In diesem Fall ist die berechnete ω-Matrix nicht positiv definit oder die falschen Werte werden berechnet.

Auch das Hinzufügen weiterer Bilder des Schachbretts ändert nichts an dem Ergebnis. Besteht die Schachbrettebene jedoch aus mehr als nur einem Quadrat, existieren mindestens vier Punkte, sodass keine drei Punkte auf einer Geraden liegen und die Kameramatrix kann berechnet werden.

$$\text{Sensorpunkte} : \begin{pmatrix} \begin{pmatrix} 3200. \\ 4800. \\ 1. \end{pmatrix} & \begin{pmatrix} 9866.67 \\ 4800. \\ 1. \end{pmatrix} & \begin{pmatrix} 16533.3 \\ 4800. \\ 1. \end{pmatrix} \\ \begin{pmatrix} 3200. \\ 4800. \\ 1. \end{pmatrix} & \begin{pmatrix} 4964.71 \\ 4800. \\ 1. \end{pmatrix} & \begin{pmatrix} 6729.41 \\ 4800. \\ 1. \end{pmatrix} \end{pmatrix} \tag{5.6}$$

Auch eine Verschiebung des 2x2 Schachbretts ermöglicht eine Bestimmung der Kameramatrix. Eine Drehung $D_{(90,z)}$ um die z-Achse stellt kein Problem bei der Berechnung dar.

- Was passiert, wenn eines der drei Schachbretter weder gedreht noch verschoben wird?

Die Berechnung der Kameramatrix war in diesem Fall erfolgreich.
Diese Fälle sind zwar in der Theorie möglich, in der Praxis jedoch kaum umsetzbar.

- Es könnten zwar drei gleiche Bilder verwendet werden, dies ist jedoch nicht notwendig, da es das Ergebnis nicht beeinflusst
- Schon eine minimale Abweichung der 90° reicht aus, um die Kameramatrix wieder zuverlässig bestimmen zu können, z. B. bei der Drehung $D_{90.0000001,x}$
- Eine frontale Schachbrettaufnahme ohne eine Drehung und Translation ist nahezu unmöglich, da es immer eine minimale Drehung oder Translation gibt.

Mit dieser Anwendung können interessante Fragen untersucht werden. Im realistischen Fall ist es schwieriger, diese Theorien zu untersuchen.

5.5 Initiale Rekonstruktion der Objektpunkte

Wie in Kapitel 4.2 beschrieben, werden für die Rekonstruktion der Objektpunkte zuerst die Fluchtpunkte berechnet um daraus mit den Richtungsvektoren der Fluchtpunkte zum Projektionszentrum den Normalenvektor zu bestimmen. Im Anschluss werden die Objektpunkte durch das Lösen eines nichtlinearen Gleichungssystem rekonstruiert. Die Parameter, die für die Rekonstruktion notwendig sind, sind die lokalisierten Sensorpunkte, die Gitterkonstante d und der Normalenvektor der Ebene.

```
1 Fluchtpunkte = calcFluchtpunkte[Sensorpunkte];
2 NormVektor = calcNormVector[Kameramatrix, Fluchtpunkte];
3 Rekonstruktionspunkte = calcObjectpoints[Sensorpunkte, Objektpunkte,
    Kameramatrix, NormVektor, Gitterkonstante];
4 ExtrinsicParameters = calcExtrinsic[NormVector, Rekonstruktionspunkte,
    Kameramatrix, Gitterkonstante];
```

5.5.1 Berechnung des Normalenvektor der Ebene

Für die Berechnung der Fluchtpunkte, mit der der Normalenvektor der Ebene bestimmt werden kann, wird zuerst für jede Punktreihe derselben geometrischen Lage der Normalenvektor $\vec{n} = (a, b, c)$ der $\mathbb{R}^3$-Ebene, welche von den Vektoren aufgespannt wird, bestimmt. Um den Normalenvektor, der senkrecht zu allen Punkten einer Punktreihe steht, zu berechnen, wird eine Koeffizientenmatrix wie in Gleichung 4.21 aufgestellt und mit der SVD, die die Werte für a, b und c ermittelt, gelöst. Der Fluchtpunkt ergibt sich durch die Ermittlung des Schnittpunkts aller Geraden der gleichen geometrischen Lage.

Die Berechnung des Normalenvektor erfolgt über die Bestimmung der Richtungsvektoren $RV_1 = \overrightarrow{O_k F_v}$ und $RV_2 = \overrightarrow{O_k F_u}$.

```
RichtungsvektorEins = {FluchtpunktEins[[1]], FluchtpunktEins[[2]], 0 }
    - {Kameramatrix[[1,3]], Kameramatrix[[2,3]], -Kameramatrix[[1,1]]};
RichtungsvektorZwei = {FluchtpunktZwei[[1]], FluchtpunktZwei[[2]], 0 }
    - {Kameramatrix[[1,3]], Kameramatrix[[2,3]], -Kameramatrix[[1,1]]};
```

Anschließend wird das Kreuzprodukt Cross[RichtungsvektorEins, RichtungsvektorZwei]; berechnet, der die Lösung für den Normalenvektor der Schachbrettebene liefert. Mit dem Beispiel 5.3 ergibt sich der Normalenvektor in normierter Form: $n = \{0.20791, -0.153016, 0.966105\}$.

Bestimmung des Normalenvektors einer gedrehten Ebene

Um nun zu prüfen, ob der errechnete Normalenvektor aus den Sensorpunkten auch dem Normalenvektor der Schachbrettebene im Raum entspricht, wird dieser aus der mit der Drehmatrix D gedrehten Ebene berechnet. Für die Berechnung des Normlanvektors ist die Orientierung und nicht die Verschiebung der Ebene relevant. Damit kann eine gegebene Ebene in allgemeiner Koordinatenform (AKF) $Ax + By + Cz + d = 0$ mit $d = 0$ auch wie folgt geschrieben werden.

$$\begin{bmatrix} A & B & C \end{bmatrix} \cdot \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = 0$$

$$N \cdot x = 0$$

Desweiteren kann die Einheitsmatrix $D^T D$ eingefügt werden, da diese Gleichung nicht verändern wird.

$$N \cdot D^T D \cdot x = 0$$

$$(N \cdot D^T) \cdot Dx \;=\; 0$$

Der Normalenvektor der gedrehten Ebene ergibt sich also aus der Gleichung

$$\begin{aligned} N &= (N \cdot D^T) \\ &= \begin{bmatrix} A & B & C \end{bmatrix} \cdot D^T. \end{aligned}$$

Da der ursprüngliche Normalenvektor in Richtung der z-Ebene zeigt, gilt

$$N \;=\; \begin{bmatrix} 0 & 0 & 1 \end{bmatrix} \cdot D^T. \tag{5.7}$$

```
1 NVektor = {0,0,1,0}.Transpose[DrehMatrix];
2 NVektor = Transpose[DrehMatrix][[3]];
3 NVektor = NVektor / Norm[NVektor];
```

Um die beiden Normalenvektoren zu vergleichen, werden beide Vektoren normiert. Der Normalenvektor, der sich aus der Drehmatrix ergibt, ist folgender: NVektor der Ebene genormt : $\{0.207912, -0.153016, 0.966105, 0.\}$. Das Ergebnis stimmt mit dem Normalenvektor, der über die Richtungsvektoren bestimmt wurden, überein, was zeigt, dass die Berechnung des Normalenvektors der Ebene über die Sensorpunkte funktioniert.

5.5.2 Rekonstruktion der Objektpunkte

Für die Berechnung der Objektpunkte wird der Normalenvektor der Ebene, der im vorherigen Schritt berechnet wurde, die Sensorpunkte, die bereits zu Beginn lokalisiert wurden und die Gitterkonstante d des Schachbretts, die nachgemessen werden kann, benötigt. Folgende Gleichungen müssen für die Rekonstruktion der Objektpunkte gelten, wobei a und b zwei neben-

einander liegende Punkte sind.

$$n \cdot (s \cdot \overrightarrow{O_k a} - t \cdot \overrightarrow{O_k b}) = 0$$
$$||s \cdot \overrightarrow{O_k a} - t \cdot \overrightarrow{O_k b}|| = d$$

In Mathematica wird das nichtlineare Gleichungssystem nummerisch über die Funktion Nsolve gelöst. Das Projektionszentrum O_k ergibt sich aus den Werten h_1, h_2 und f aus der Kameramatrix.

```
a = N[{m[[i, 1]], m[[i, 2]], 0}
    - {Kameramatrix[[1, 3]], Kameramatrix[[2,3]], -Kameramatrix[[1,1]]}];
b = N[{m[[i + 1, 1]], m[[i + 1, 2]], 0}
    - {Kameramatrix[[1, 3]], Kameramatrix[[2, 3]], -Kameramatrix[[1, 1]]}];
SolVector = Last[{s, t} /. NSolve[ Norm[(s * a) - (t * b)] ==
   Gitterkonstante && NormVector.(s * a - t * b) == 0, {s, t}]];
```

Die Lösung der Nsolve-Funktion beinhaltet jeweils einen positiven und negativen Wert für s und t. Die Modellrechnung bezieht sich auf das Modell der Lochkamera in Positivlage, weswegen der positive Wert der Lösung verwendet wird, der sich im letzten Teil Last[] der Lösungsmenge befindet. Im letzten Schritt muss noch der Punkt A und der Punkt B berechnet werden.

$$s \cdot a = A$$
$$t \cdot b = B$$

```
A = {SolVector[[1]]*a[[1]], SolVector[[1]]*a[[2]], SolVector[[1]]*a[[3]]};
B = {SolVector[[2]]*b[[1]], SolVector[[2]]*b[[2]], SolVector[[2]]*b[[3]]};
```

Wie Abbildung 5.4 zeigt, ist die Berechnung der Objektpunkte über das Lösen des nichtlinearen Gleichungssystems erfolgreich.

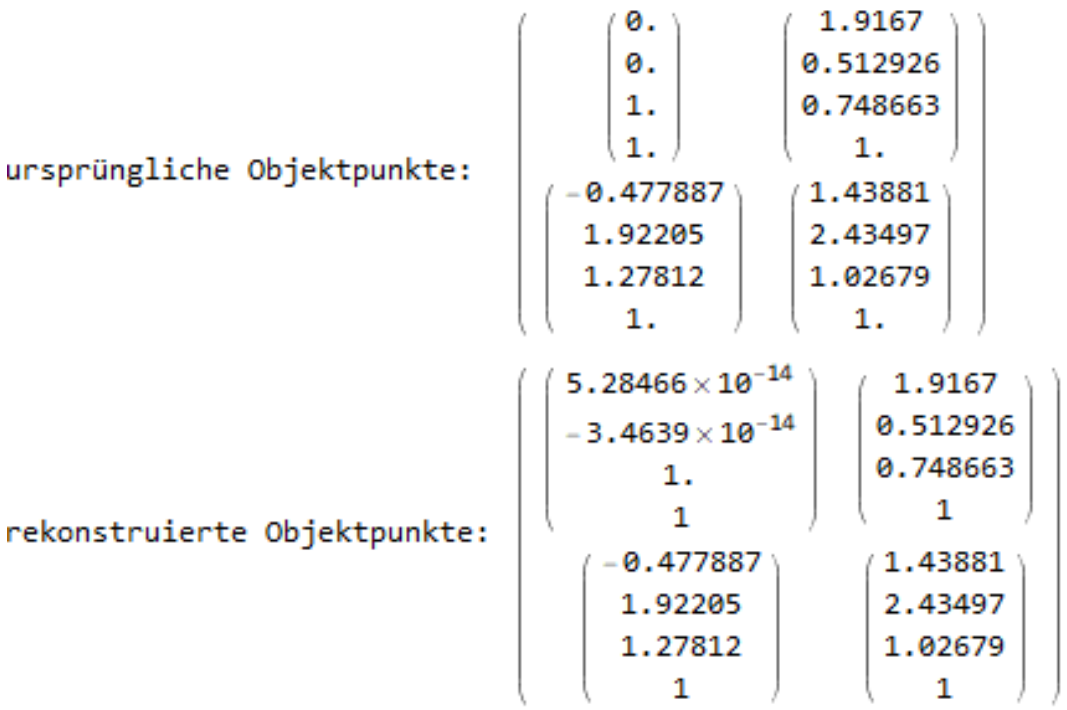


Abbildung 5.4: Rekonstruktion der Objektpunkte

Prüfung, ob die Punkte komplanar sind

Um festzustellen, ob die Rekonstruktion der Objektpunkte erfolgreich war, kann geprüft werden, ob alle Objektpunkte in einer Ebene liegen. Hierfür wird der Ausdruck aufgestellt:

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 \\ X_2 & Y_2 & Z_2 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ X_n & Y_n & Z_n & 1 \end{bmatrix} \cdot \begin{bmatrix} A \\ B \\ C \\ D \end{bmatrix} = 0$$

Mit 20 Objektpunkten ergibt sich ein überbestimmtes Gleichungssystem, das mit der SVD gelöst werden kann. Werden die Lösungen für A, B, C und D in den Ausdruck der linearen Form

$$X_1 \cdot A + Y_1 \cdot B + Z_1 \cdot C + D \cdot 1 = 0$$

eingesetzt und beträgt die Lösung nicht exakt 0, befinden die einzelnen Punkte nicht exakt in der Ebene.

5.6 Initiale Berechnung der extrinsischen Kameraparameter

Für die Berechnung der extrinsischen Kameraparameter werden die Daten aus 5.3 verwendet → $\alpha = 6$, $\beta = 9$ und $\gamma = 13$ und $t = (0, 0, 4000)$.
Der Translationsvektor ist der erste Objektpunkt und kann aus den rekonstruierten Objektpunkten entnommen werden.

```
t = Objektpunkte[[1,1]];
```

Der Winkel α ist der Winkel zwischen $\hat{z} = (0, 0, 1)$ und der Projektion des Normalenvektors auf die y-z-Ebene.

$$\alpha = \measuredangle(\hat{z}, n')$$

```
zDach = {0,0,1};
n_projiziert = {0, NormVector[[2]], NormVector[[3]]};
alphaBerechnet = VectorAngle[zDach, n_projiziert];
```

β ergibt sich aus der Berechnung des Winkels zwischen n und $\hat{n}$.

$$\beta = \measuredangle(n, n')$$

```
betaBerechnet = VectorAngle[normVector, n_projiziert];
```

Der Winkel γ lässt sich berechnen, indem der Winkel zwischen zwei rekonstruierten Objektpunkt-Verbindungsvektoren $R_A R_B$ und den Objektpunkten $G_A G_B$, die nach einer Drehung um die bereits berechneten Winkel

α, β und der Translation mit t entstehen.

$$G_A = (D_{(alphaberechnet,x)} \circ D_{(betaberechnet,y)} \cdot (0,0,0,1)) + t$$
$$G_B = (D_{(alphaberechnet,x)} \circ D_{(betaberechnet,y)} \cdot (0,d,0,1)) + t$$

Die Berechnung von γ ist damit:

$$\gamma = \measuredangle(\overrightarrow{G_A G_B}, \overrightarrow{R_A R_B})$$

```
A = ((D(alpha).D(beta)).{0,0,0,0}) + t;
B= ((D(alpha).D(beta)).{0,Gitterkonstante,0,0}) + t;
VektorAB = Drop[B–A, −1];
```

Für den Vektor $\overrightarrow{R_A R_B}$ werden die beiden nebeneinanderliegenden Objektpunkte verwendet.

```
VektorABObjektpunkte = Drop[Objektpunkte[[2,1]] − Objektpunkte[[1,1]], −1];
gammaBerechnet = VectorAngle[VektorABObjektpunkte, VektorAB];
```

Werden die Winkel mit den Formeln berechnet, ergibt sich folgendes für die extrinsischen Parameter:

Winkel α: 9. Grad

Winkel α: 12. Grad

Winkel γ: 18. Grad

Translationsvektor t: (0, 0, 4000, 1)

Die Ergebnisse zeigen, dass die Berechnungen der extrinsischen Kameraparameter erfolgreich waren. Im Anhang 7.3 befindet sich die Ausgabe der *Mathematica*-Datei, bei der die Kamerakalibrierung für die Gleichung 5.3 ausgeführt wird.

5.7 Untersuchung der Algorithmen mit gestörten Sensorpunkten

Bisher wurden ungestörte Sensorpunkte verwendet, um zu prüfen, ob die Algorithmen der Kamerakalibrierung und die Rekonstruktion der Objektpunkte funktionieren. Diese Anwendung geht davon aus, dass die Schachbretteckpunkte bereits lokalisiert wurden. Um Schachbretteckpunkte zu lokalisieren, wurde ein `FindPoints`-Algorithmus in einem Forschungsprojekt namens `ADEKKA` von T. Schneider, S. Piontek, P. Hafen und N. Hottong [21] entwickelt, der sich auf die Methode von Chen und Zhang [22] stützt und Schachbretteckpunkte im Sub-Pixelbereich lokalisiert. In der Realität werden die Sensorpunkte durch verschiedene Faktoren wie beispielsweise Sensorrauschen, Über- oder Unterbelichtung meist gestört, weshalb die Lokalisierung der Schachbretteckpunkte im Subpixelbereich daneben liegen kann. Um einen Realitätsfall nachzustellen und den Algorithmus für den Bündelblockausgleich zu testen, werden minimale Störungen eingebaut, die die Kamerakalibrierung beeinflussen. RandomVariate[NormalDistribution [0, sd]] simuliert eine kontinuierliche Wahrscheinlichkeitsverteilung mit der (Gaußschen) Verteilung mit Mittelwert 0 und Standardabweichung sd [23]. In diesem Fall wurde eine Störung um $\frac{1}{4}$ Pixel gewählt.

```
sd = PixelPitch/1000/4;
QuadratGestoert = Map[{RandomVariate[NormalDistribution[0, sd]],
    RandomVariate[NormalDistribution[0, sd]], 0} + # &, Sensorpunkte];
```

Abbildung 5.5 zeigt mit den roten Punkten die ungestörten und mit dem blauen Punkten die gestörten Sensorkoordinaten. Je größer der Abstand des Objekts zur Kamera ist, desto größer wird auch der Abstand der gestörten zu den ungestörten Punkten. Schon bei einer minimalen Abweichung von $\frac{1}{4}$-Pixeln, ändern sich die Einträge der Kameramatrix. Aus den

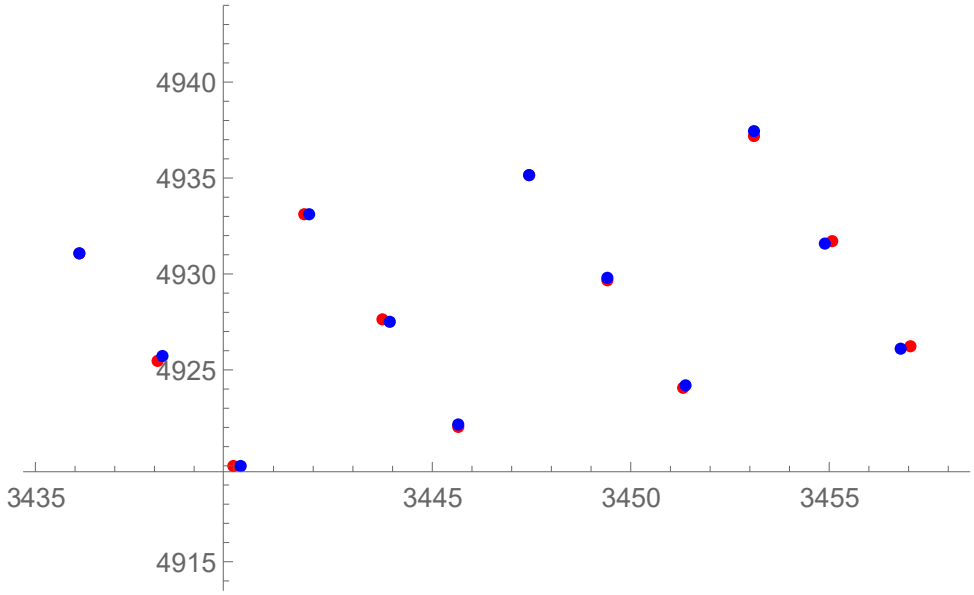


Abbildung 5.5: Störungen der Sensorpunkte bei $z = 10$ Meter

zuvor eingegebenen Werte für $f = 9$, $H = (12, 18)$, werden mit gestörten Punkten folgende Werte berechnet:

Brennweite: 9.1066 mm. Der Hauptpunkt liegt bei : H (11.9127, 18.0774)

Die ermittelte Kameramatrix in mm:
$$\begin{pmatrix} 9.1066 & 0.0197619 & 11.9127 \\ 0. & 9.1066 & 18.0774 \\ 0. & 0. & 1. \end{pmatrix}$$

Auch die Ermittlung der Objektpunkte wird durch die gestörten Punkte beeinflusst. Abbildung 5.6 zeigt die Berechnung der Fluchtpunkte mit gestörten Sensorpunkten. Zu sehen ist, dass die blauen Sensorpunkte nicht direkt auf einer Punktreihe liegen, weswegen eine Ausgleichslösung der Gerade mit der SVD bestimmt wurde. Auch der Schnittpunkt der Geraden wird mit einer Ausgleichslösung ermittelt. Die ursprünglichen Objektpunkten entsprechen nicht mehr den rekonstruierten Objektpunkten.
Wird die initial berechnete Kameramatrix K auf die rekonstruierten Ob-

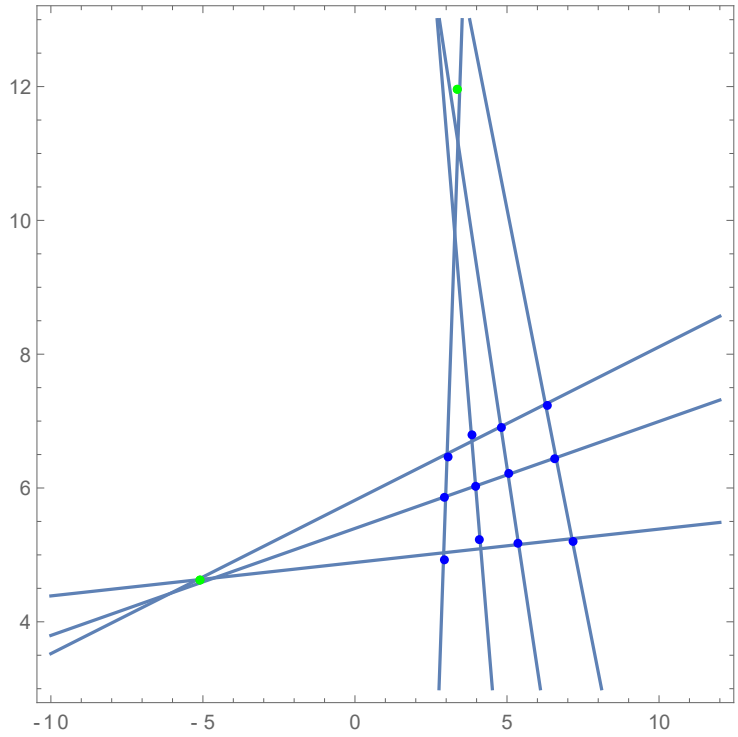


Abbildung 5.6: Rekonstruktion der Objektpunkte mit gestörten Sensorpunkten

jektpunkte R_k angewandt, ergeben sich reprojizierte homogene Punkte $P = (x, y, 1)^T$ auf dem Sensor. Mit diesen wird der Reprojektionsfehler R berechnet, indem die reprojizierten Punkte $RP_{SP}^{m\,x\,n}$ mit den ursprünglich lokalisierten Punkten $L_{SP}^{m\,x\,n}$ verglichen werden. Die Summe der Abstände zwischen einem reprojizierten Punkt und dem zugehörigen lokalisierten Punkt im Quadrat ergibt den Reprojektionsfehler R, der im nächsten Schritt mit dem Bündelblockausgleich minimiert wird.

$$R = \sum_{m\,x\,n} \|L_{SP}^{m\,x\,n} - RP_{SP}^{m\,x\,n}\|^2$$

Die initial berechnete Kameramatrix, die die rekonstruierten Objektpunkte projiziert, ist jene, die auch mit den Störpunkten berechnet wurde.

5.8 Bündelblockausgleich

Gesucht sind die intrinsischen und extrinsischen Parameter für die der Reprojektionsfehler möglichst minimal ist.

$$min\left(\sum_{jk} \|P(f,h1,h2)\cdot((D(\alpha,\beta,\gamma)\cdot s_{m\,\mathrm{x}\,n})+t)-b_{jk}\|^2\right)$$

Um das beste Ergebnis, das möglichst nahe an den eingegeben Werten liegt - der sogenannten `Ground Truth` - zu erzielen, werden verschiedene Herangehensweisen des Minimierungsalgorithmus getestet. Für die Prüfung der `Ground Truth` wird hierbei der Objektreferenzfehler O_{RF} eingeführt. Dieser vergleicht die ursprünglichen Objektpunkte, die nach der Drehung und Translation der eingegebenen Werte entstanden sind, mit jenen Objektpunkten, die nach Anwendung der Drehung und Translation der optimierten Werte erzeugt wurden.

$$O_{RF} \;=\; \sum \|((D(\alpha,\beta,\gamma)\cdot s_{m\,\mathrm{x}\,n})+t)-((D(\alpha_{opt},\beta_{opt},\gamma_{opt})\cdot s_{m\,\mathrm{x}\,n})+t_{opt})\|^2$$

Für die Untersuchung der Algorithmen, werden dieselben gestörten Sensorpunkte verwendet, um das die Algorithmen auch dahingehend vergleichen zu können. Anfangs wird davon ausgegangen, dass für f dieselbe Brennweite existiert, weswegen der Mittelwert $\overline{f}$ genommen wird.

$$\overline{f} \;=\; \frac{f_x + f_y}{2}$$

Außerdem wird davon ausgegangen, dass der Scherungsfaktor $skew = 0$ beträgt, da dieser nur in seltenen Fällen abweicht. Abbildung 5.7 zeigt die Auswirkungen eines Bildes in unterschiedlichen Fällen.

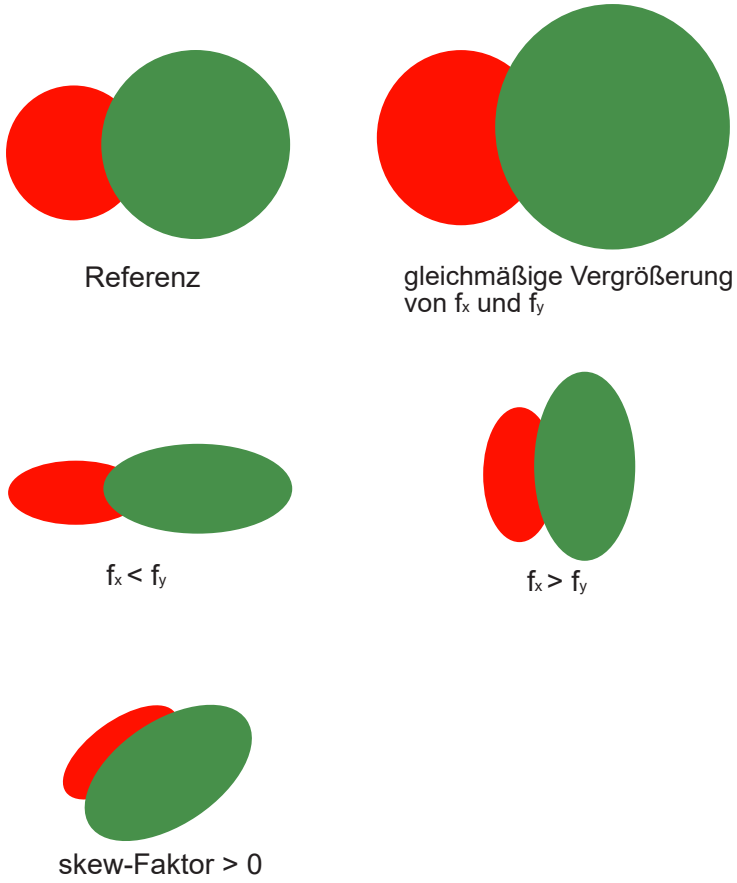


Abbildung 5.7: Auswirkungen verschiedener Kameraparameter

Dahingehend ändert sich die zuvor mit Störpunkten initial berechnete Kameramatrix:

Die ermittelte Kameramatrix in mm:
$$\begin{pmatrix} 9.1066 & 0. & 11.9127 \\ 0. & 9.1066 & 18.0774 \\ 0. & 0. & 1. \end{pmatrix}$$

In *Mathematica* sind die zwei Algorithmen FindMinimum und NMinimize zur Mininimierung implementiert.

```
FindMinimum [ f,{ x,x0 }];
```

sucht nach einem lokalen Minimum in f, ausgehend vom Startwert $x = x_0$ [24]. Folgende Algorithmen, die eine Minimierung durchführen sind implementiert: ConjugateGradient, PrincipalAxis, LevenbergMarquardt, Newton, QuasiNewton, InteriorPoint, und LinearProgramming. Die Methode

selbst wird, wenn nicht anders angegeben, von *Mathematica* selbst gewählt. Diese kann auch manuell eingegeben werden. Als Startwerte für den Minimierungsalgorithmus werden die berechneten Parameter verwendet.

```
x = FindMinimum[MinimizeFunctionWithOneF[Sensorpunkte, Urgitter, f1, h1, h2
    , alpha, beta, gamma, t1, t2, t3], {f1, Kameraparameter[[1]]}, {h1,
    Kameraparameter[[4]]}, {h2, Kameraparameter[[5]]}, {alpha,
    alphaBerechnet}, {beta, betaBerechnet}, {gamma, gammaBerechnet}, {t1, t
    [[1]]}, {t2, t[[2]]}, {t3, t[[3]]}, MaxIterations -> 100000,
    AccuracyGoal -> 6];
```

Um ein gutes Ergebnis zu erzielen, werden die Parameter MaxIterations –> 100000 und AccuracyGoal –> 6, Method–>"LevenbergMarquardt" ergänzt. Der Standardwert für die maximale Anzahl an Iterationen liegt bei 100 Durchläufen, was für die Berechnung nicht ausreicht. Das Genauigkeitsziel `AccuracyGoal`, welches angibt, wie viele effektive Stellen der Genauigkeit im Endergebnis gesucht werden soll, sollte bei 6 liegen und für die Methode, nach der optimiert werden sollte, wird der Levenberg-Marquardt Algorithmus gewählt (vgl. Hardly und Zissermann [7]).
Der Objektreferenzfehler beträgt mit den initial berechneten Werten für die extrinsischen Parameter zu Beginn

```
Objektreferenzfehler: 66947.8.
```

Mit diesen Werten und ausgehend von einer Kameramatrix, bei der der `skew`-Faktor gleich Null und für f_x und f_y der Mittelwert genommen wird, wird die Lösung, die sich auf der nächsten Seite befindet, berechnet. Abbildung 5.8 zeigt eine Minimierung des Reprojektionsfehlers von 0.352529 zu $2.11595 \cdot 10^{-11}$, jedoch auch eine Entfernung der berechneten Parametern zur `Ground Truth`. Die initial berechneten Werte liegen näher an der `Ground Truth` als die Werte, die durch den Bündelblockausgleich berechnet wurden. Zu erkennen ist das durch den Objektreferenzfehler, der sich von 66947.8 auf 524320 erhöht hat.

-----START DER OPTIMIERUNG DER WERTE------

Die 'Ground Truth'- Werte für die Kameramatrix: $\begin{pmatrix} 2400 & 0 & 3200 \\ 0 & 2400 & 4800 \\ 0 & 0 & 1 \end{pmatrix}$

Die 'Ground Truth'- Werte für die extrinsischen Parameter:

α: 9 Grad, β: 12 Grad, γ: 18 Grad, t = $\begin{pmatrix} 1000 \\ 500 \\ 4000 \end{pmatrix}$.

Initiale KameraMatrix: $\begin{pmatrix} 2429.02 & 5.26984 & 3176.71 \\ 0. & 2427.84 & 4820.64 \\ 0. & 0. & 1. \end{pmatrix}$

Initiale Werte für α: 8.9786, β: 12.3901, γ: 18.128, t: $\begin{pmatrix} 1040.14 \\ 466.192 \\ 4053.52 \\ 1 \end{pmatrix}$

Objektreferenzfehler: 66947.8

-----START DER OPTIMIERUNG FINDMINIMUM MIT f = (f1 + f2) / 2, skew = 0------ nach LevenbergMarquardt

-----OPTIMIERTE WERTE------

Reprojektionsfehler nach Minimierung: 2.2289×10^{-11}

Die berechnete Kameramatrix: $\begin{pmatrix} 2276.1 & 0 & 3216.16 \\ 0 & 2276.1 & 4775.69 \\ 0 & 0 & 1 \end{pmatrix}$

Berechnete Werte nach Minimierung für α: 8.28947, β: 11.3704, γ: 18.1198, t: $\begin{pmatrix} 973.911 \\ 540.989 \\ 3796.82 \end{pmatrix}$

Objektreferenzfehler: 524320.

Abbildung 5.8: Bündelblockausgleich mit $\overline{f}$ und $s = 0$

Damit sich die Werte des Bündelblockausgleichs der `Ground Truth` nähern, wird nun untersucht, ob sich der Algorihtmus mit Erhöhung der Freiheitsgrade durch Hinzunahme des *skew*-Faktors und durch Lösen der f_x und f_y Bedingung den ursprünglichen Werten nähert. Folglich muss der Algorithmus elf Parameter optimieren.

-----START DER OPTIMIERUNG DER WERTE------

Die 'Ground Truth'- Werte für die Kameramatrix: $\begin{pmatrix} 2400 & 0 & 3200 \\ 0 & 2400 & 4800 \\ 0 & 0 & 1 \end{pmatrix}$

Die 'Ground Truth'- Werte für die extrinsischen Parameter:

α: 9 Grad, β: 12 Grad, γ: 18 Grad, t = $\begin{pmatrix} 1000 \\ 500 \\ 4000 \end{pmatrix}$.

Initiale KameraMatrix: $\begin{pmatrix} 2429.02 & 5.26984 & 3176.71 \\ 0. & 2427.84 & 4820.64 \\ 0. & 0. & 1. \end{pmatrix}$

Initiale Werte für α: 8.9786, β: 12.3901, γ: 18.128, t: $\begin{pmatrix} 1040.14 \\ 466.192 \\ 4053.52 \\ 1 \end{pmatrix}$

Objektreferenzfehler: 66 947.8

-----START DER FINDMINIMUM OPTIMIERUNG MIT f1 != f2, skew != 0------ nach LevenbergMarquardt

-----OPTIMIERTE WERTE------

Reprojektionsfehler nach Minimierung: 3.10177×10^{-11}

Die berechnete Kameramatrix: $\begin{pmatrix} 2395.87 & 8.24497 & 3176.98 \\ 0 & 2393.01 & 4819.17 \\ 0 & 0 & 1 \end{pmatrix}$

Berechnete Werte nach Minimierung für α: 8.56928, β: 11.9259, γ: 18.2139, t: $\begin{pmatrix} 1038.86 \\ 469.562 \\ 4001.24 \end{pmatrix}$

Objektreferenzfehler: 29 074.

Abbildung 5.9: Bündelblockausgleich mit 11 Freiheitsgraden

Abbildung 5.9 zeigt eine Verbesserung bei der Berechnung der Kameraparameter mit elf Freiheitsgraden, die sich in der Nähe der `Ground Truth` befinden. Der Objektreferenzfehler hat sich auf 29074 verringert.
Die Funktion

```
1 NMinimize[{f,cons},{x,y}];
```

sucht nach dem globalen Minimum von f unter gegebenen Einschränkungen. Das Problem bei der Verwendung des Algorithmus ist, dass dieser einen Mindestwert und einen Höchstwert der Parameter benötigt und so immer einer Einschränkung unterliegt. Theoretisch kann davon ausgegan-

gen werden, dass die initiale Berechnung der Kameraparameter nur minimal abweicht und so eine Einschränkung der Winkel bei ± 5 und bei den Parametern in *mm*-Angaben bei ± 100 eingegeben werden kann.

```
x = NMinimize[{MinimizeFunction[Sensorpunkte, Urgitter, f1, f2, skewFaktor,
    h1, h2, alpha, beta, gamma, t1, t2, t3],
f1Berechnet - 100 <= f1 <= f1Berechnet + 100 &&
f2Berechnet - 100 <= f2 <= f2Berechnet + 100 &&
skewFaktorBerechnet <= skewFaktor <= skewFaktorBerechnet + 100 &&
h1Berechnet - 100 <= h1 <= h1Berechnet + 100 &&
h2Berechnet - 100 <= h2 <= h2Berechnet + 100 &&
alphaBerechnet - 5 <= alpha <= alphaBerechnet + 5 && betaBerechnet - 5 <=
    beta <= betaBerechnet + 5 && gammaBerechnet - 5 <= gamma <=
    gammaBerechnet + 5 && t[[1]] - 100 <= t1 <= t[[1]] + 100 && t[[2]] - 100
    <= t2 <= t[[2]] + 100 && t[[3]] - 100 <= t3 <= t[[3]] + 100},
{f1, f2, skewFaktor, h1, h2, alpha, beta, gamma, t1, t2, t3}, MaxIterations
    -> 100000, AccuracyGoal -> 6];
```

Abbildung 5.10 zeigt das Ergebnis des NMinimize-Algorithmus. Zu sehen ist, dass die optimierten Werte sich ebenfalls von der `Ground Truth` entfernen. Auch der Objektreferenzfehler liegt nun bei 167780 und ist damit höher als der Anfangswert. Unter diesem Aspekt und durch die gegebene Einschränkung der NMinimize[]-Funktion ist der FindMinimum-Algorithmus mit elf Freiheitsgraden der, der sich am nächsten der `Ground Truth` nähert. Die Ausgabedatei 7.4 zeigt die Berechnung der Kameramatrix mit gestörten Punkten und den Vergleich der zu optimierenden Algorithmen.

-----START DER OPTIMIERUNG DER WERTE------

Die 'Ground Truth'- Werte für die Kameramatrix: $\begin{pmatrix} 2400 & 0 & 3200 \\ 0 & 2400 & 4800 \\ 0 & 0 & 1 \end{pmatrix}$

Die 'Ground Truth'- Werte für die extrinsischen Parameter:

α: 9 Grad, β: 12 Grad, γ: 18 Grad, t = $\begin{pmatrix} 1000 \\ 500 \\ 4000 \end{pmatrix}$.

Initiale KameraMatrix: $\begin{pmatrix} 2429.02 & 5.26984 & 3176.71 \\ 0. & 2427.84 & 4820.64 \\ 0. & 0. & 1. \end{pmatrix}$

Initiale Werte für α: 8.9786, β: 12.3901, γ: 18.128, t: $\begin{pmatrix} 1040.14 \\ 466.192 \\ 4053.52 \\ 1 \end{pmatrix}$

Objektreferenzfehler: 66 947.8

-----START DER NMINIMIZE OPTIMIERUNG MIT f1 != f2, skew != 0------

-----OPTIMIERTE WERTE------

Reprojektionsfehler nach Minimierung: 6.07551×10^{-11}

Die berechnete Kameramatrix: $\begin{pmatrix} 2431.06 & 9.4582 & 3156.2 \\ 0 & 2427.05 & 4839.76 \\ 0 & 0 & 1 \end{pmatrix}$

Berechnete Werte nach Minimierung für α: 8.51086, β: 12.4498, γ: 18.2177, t: $\begin{pmatrix} 1075.07 \\ 435.972 \\ 4065.97 \end{pmatrix}$

Objektreferenzfehler: 167 780.

Abbildung 5.10: Bündelblockausgleich mit der NMinimize-Funktion

5.9 Portierung der Berechnungen in *Python*

Dieses Kapitel beschäftigt sich mit der Frage, ob sich die in *Mathematica* geschriebene Anwendung in *Python* portieren lässt. Hierfür werden zuerst die Grundzüge von *Python* erläutert. Anschließend wird auf die Berechnung eingegangen und ein Fazit gezogen.

5.9.1 Einführung in *Python*

Python ist eine universelle, vielseitige und leistungsstarke Open-Source-Programmiersprache, die wegen ihrer Einfachheit prägnant und leicht zu lesen ist [25]. Die Grundfunktionen von Python können mithilfe verschiedener Bibliotheken erweitert werden. So gibt es beispielsweise die Bibliothek `numpy` [26], bei der nummerische Berechnungen in ihrer Performance optimiert, durchgeführt werden können. Auch die Bibliothek `sympy` [27], die wie *Mathematica* auch symbolische Berechnungen durchführt, kann eingefügt werden. Grundlegend ist jede Bibliothek, die hinzugefügt werden kann, open-source, was die Algorithmen transparent gestaltet. Jede Bibliothek hat dabei eine hervorragende Online-Dokumentation und durch die große Python-Community existieren viele Foren, in denen Fragen beantwortet werden. Es gibt eine Python-Console, die es ermöglicht Code interaktiv auszuführen, welche jedoch eher dazu gedacht ist, Befehle auszuführen oder Code zu testen, ohne dafür eine Datei erstellen zu müssen [28]. *Python* bietet ebenfalls die Möglichkeit eine modularisierte Anwendung aufzubauen. Zusammengehörige Funktionen können zur Übersichtlichkeit in eine Datei geschrieben und in anderen Dateien wieder aufgerufen werden. Dateien, die in Python geschrieben wurden, werden mit der Endung `.py` markiert. Als IDE für Python wurde die Software *PyCharm* von JetBrains verwendet.

Vorteil dieser IDE ist der intelligente Code-Editor, der eine Versionskontrolle wie `GIT` zulässt und in einer kostenfreien Version erhältlich ist [29]. In Python können Listen, wie sie in *Mathematica* verwendet werden, mit `Arrays` erstellt werden. Um eine Matrizenrechnungen miteinzubeziehen, wurde die Bibliothek `sympy` eingebunden, bei der Matrizen, die ebenfalls mehrdimensional sein können, mit `m = Matrix([ ])` erstellt werden. Die Indizierung in *Python* startet bei 0.

5.9.2 Berechnungen in *Python*

Für die Matrizenberechnungen und die Berechnungen von Gleichungssystemen wurde die Bibliothek `sympy` verwendet. Erstellt werden die Schachbrettpunkte mit einer mehrdimensionalen Matrix.

```
1 S = Matrix([[[0, 0, 0, 1], [25, 0, 0, 1]], [[0, 25, 0, 1],
2     [25, 25, 0, 1]]])
```

Diese bietet zwar die symbolische Berechnung an, zur Optimierung der Laufzeit wird dennoch die nummerische Berechnung gewählt. Dahingehend entstehen kleine Rundungsfehler, wie zum Beispiel bei der Projektion einer Schachbrettebene auf den Sensor. Mit der Einbindung der `matplotlib`-Bibliothek können diese Sensorpunkte grafisch dargestellt werden. Eine Grafik wird in Abbildung 5.11 dargestellt. Die Kameramatrix, welche mit den gerundeten Werten berechnet wird, ist dann folgende:

$$\begin{bmatrix} 2399.97349396187 & -0.00167995955581467 & 3200.01438882414 \\ -5.16536851036104 \cdot 10^{-13} & 2399.9720093794 & 4799.99197383291 \\ -1.07612023905873 \cdot 10^{-16} & -6.2602223797275 \cdot 10^{-17} & 1.0 \end{bmatrix}$$

Dieser leichte Rundungsfehler lässt sich auch nicht mit der symbolischen Rechnung verhindern, da die Singulärwertszerlegung nur nummerisch be-

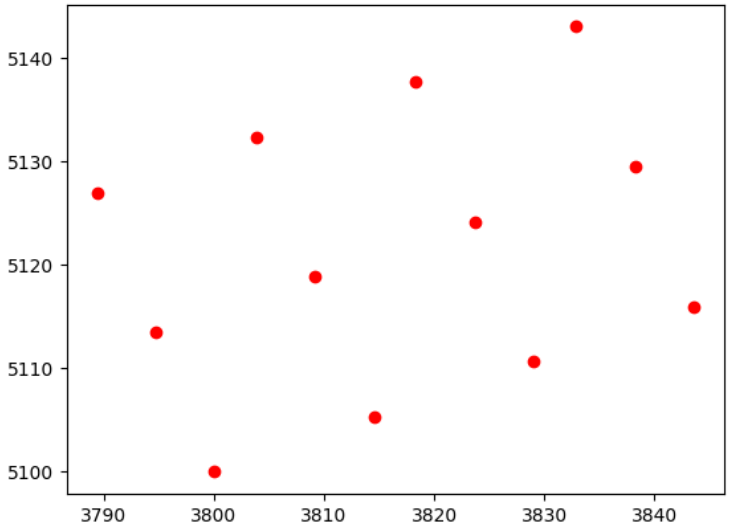


Abbildung 5.11: Projizierte Sensorpunkte in *Python*

rechnet werden kann. Um die Singulärwertszerlegung in Python durchzuführen, wurden verschiedene Bibliotheken getestet, in der die Funktion SVD bereits implementiert ist. Die für die allgemeinen Matrizenrechnungen benutzte Bibliothek `sympy` bietet die Funktion matrix.singular_value_decomposition(), welche die Lösungen für die V-Matrix jedoch nicht sortiert und bei einer geringen Anzahl an Gleichungen nicht die beste Ausgleichslösung berechnet. Deswegen wird die SVD, die in der `numpy`-Bibliothek implementiert ist, mit der Funktion v_matrix = np.linalg.svd(matrix, full_matrices =True)[2] berechnet. Die gespeicherten Sensorpunkte sind aber in einer `sympy`-Matrix gespeichert, weswegen diese erst in ein `numpy`-Array abgeändert werden muss. Nach Berechnung der SVD kann das Ergebnis wieder in eine `sympy`-Matrix übertragen werden.

```
1 def calculate_svd(matrix):
2     matrix = np.array(matrix).astype(np.float64)
3     v_matrix = np.linalg.svd(matrix, full_matrices=True)[2]
4     print(v_matrix)
5     solution_vector = v_matrix[-1]
6     return sym.Matrix(solution_vector).T
```

Die gesuchte Lösungsmatrix setzt sich hierbei aus den Werten der letzten Zeile der Matrix V zusammen. Die Rekonstruktion der Objektpunkte über das Lösen des nichtlinearen Gleichungssystems erfolgt auch über das Lösen mit einer in `sympy` implementierten solve-Funktion, welche eine längere Berechnungszeit hat, als die solve-Funktion in *Mathematica*.

```
s, t = sp.symbols("s t", real=True)
eq1 = sp.Eq(((s * a) - (t * b)).norm(), grid_constant)
eq2 = sp.Eq(norm_vector.T @ ((s * a) - (t * b)), 0)
solution_vector = sp.solve([eq1, eq2], [s, t])[1]
```

Mit den minimalen Rundungsfehlern ergibt sich für den ersten Objektpunkt und damit auch für den Translationsvektor t der Wert:

$$t = [999.975602721404,\ 500.013168982086,\ 3999.95415962813]$$

und für die extrinsischen Parameter beispielsweise $\alpha = 8.99988611849556$ Grad.
Werden die Sensorpunkte minimal gestört, entsteht auch hier ein Reprojektionsfehler, der minimiert werden kann. Mit der Einbindung der `scipy`-Bibliothek kann ein Bündelblockausgleich durchgeführt werden.

```
solution = least_squares(minimize_function, x0, ftol=1e-2, xtol=1e-2, args
    =(grid, checkerboard))
```

Hierbei ergeben sich bei nicht gestörten Sensorpunkten die Werte

```
array([ f1 = 2399.9,  f2 = 2399.9, s = -1.67995956e-03,  h1 = 3.2000,
        h2 = 4799.999,  alpha = 8.9998, beta = 11.999, gamma =  18.000,
        t1 = 9.9997, t2 = 500.000, t3 = 3999.95416].
```

Werden auch hier die Sensorpunkte um $\frac{1}{4}$-Pixel gestört ergeben sich die initial berechneten Werte:

$$
\begin{aligned}
K &= \begin{bmatrix} 2427.864720 & 1.8020 & 3184.75702 \\ 0 & 2429.4586 & 4808.5049 \\ 0 & 0 & 1 \end{bmatrix} \\
alpha &= 9.12649622 \\
beta &= 12.4285553 \\
gamma &= 18.08541108 \\
t &= 1026.3935,\ 486.2952,\ 4050.3521
\end{aligned}
$$

mit folgenden optimierten Werten nach Durchführung des Bündelblockausgleichs:

```
x: array([ f1 = 2388.70568,  f2 = 2388.07841, -7.09920890e-01,
    h1 = 3206.112, h2 = 4796.59186,  alpha = 8.908792,
     beta = 12.0118525,  gamma = 17.933867,
     t1 = 9898.3584,  505.692948,  3981.27373])
```

Die optimierten Werte liegen näher an der `Ground Truth` als die initial berechneten Werte, was zeigt, dass dieser Algorithmus erfolgreich ist.

5.9.3 Fazit zur Portierung der Berechnungen

Auch in *Python* entstand eine modularisierte Anwendung zur Berechnung der Kamerakalibrierung, bei der auch der Bündelblockausgleich implementiert wurde. Die Schwierigkeit in *Python* besteht jedoch in dem Wechsel der verschiedenen Bibliotheken. Bei einigen Funktionen, wie zum Beispiel dem Bündelblockausgleich, musste zwischen `sympy` und `numpy` gewechselt werden, da in `numpy` leistungsstärkere Algorithmen implementiert sind, in `sympy` jedoch die Symbolik (vor allem im Bezug auf das Lösen von Glei-

chungen) implementiert ist.

Die print-Befehle werden derzeit in *Python* in einer Konsole ausgegeben und sind dahingehend schwerer zu lesen. Abbildung 5.12 zeigt einen Ausschnitt der Konsole, in der die Kameramatrix ausgegeben wurde.

```
 Die Matrix ist positiv definit:  True

Die ermittelte Kameramatrix:

[  2531.82325913705    9.18657006708408  3125.80552198361|
|                                                        |
|5.40091526016058e-13  2539.9707851899   4841.46645909373|
|                                                        |
|1.11555358397992e-16         0                1.0      ]

 Die ermittelte Kameramatrix in mm:

[  9.50961383311302    0.0344496377515653  11.7217707074385|
|                                                         |
|5.40091526016058e-13   9.50961383311302   18.1554992216015|
|                                                         |
|1.11555358397992e-16          0                 1.0      ]

 Brennweite:  9.50961383311302 mm.
 Der Hauptpunkt liegt bei: ( 11.7217707074385 , 18.1554992216015 ).
```

Abbildung 5.12: Konsolenausgabe in *Python*

Der Vorteil gegenüber *Mathematica* ist der Abbruch des Programmdurchlaufs in *Python*, wenn ein Fehler auftritt, zum Beispiel, wenn die ω-Matrix nicht positiv definit ist. In *Mathematica* gibt es diesen Abbruch jedoch nicht, was teilweise zu sehr langen Berechnungszeiten und keinem Ergebnis führt. An einigen Stellen kann die *Python*-Anwendung noch verbessert werden. Teile dieser Verbesserung könnte die Optimierung der Laufzeit sein und die grafische Aufbesserung der Ergebnisse. Auch die Möglichkeit zur objektorientierten Programmierung ist gegeben und sollte untersucht werden. Die entstandene *Python*-Anwendung zeigt aber, dass eine Portierung der Berechnungen in *Python* möglich ist.

6 Fazit und Ausblick

Die Aufgabenstellung, die Algorithmen und deren Robustheit anhand der Modellrechnung zu überprüfen, wurde erfolgreich behandelt. Neben der Berechnung der Kameramatrix wurde die Rekonstruktion der Objektpunkte genauer betrachtet. Der Bündelblockausgleich, der die Minimierung des Reprojektionsfehlers behandelt, wurde ebenfalls implementiert, untersucht und so angepasst, dass sich die Werte der `Ground Truth` nähern. In diesem Projekt entstand dabei eine *Mathematica*- und eine *Python*-Anwendung, die nachvollziehbar und modular aufgebaut wurde, sodass diese sich für eine Weiterentwicklung eignet. Teile dieser Weiterentwicklung könnte die Hinzunahme der Verzeichnung in das Kameramodell sein, um auch den Algorithmus der Korrektur zu untersuchen und prüfen. Auch die Frage, ob durch Hinzunahme mehrerer Kameras K_1, K_2, K_n die Ermittlung der Lage der 3D-Objekte präzisiert werden kann, ist Grundstein künftiger Projekte.

7 Anhang

7.1 Singulärwertszerlegung

Die Singulärwertszerlegung ist ein Verfahren zur Bestimmung einer Ausgleichslösung. Abbildung 7.1 zeigt eine Ausgleichslösung der Geraden einer Punktreihe, bei der die Punkte nicht eindeutig auf einer Geraden liegen.

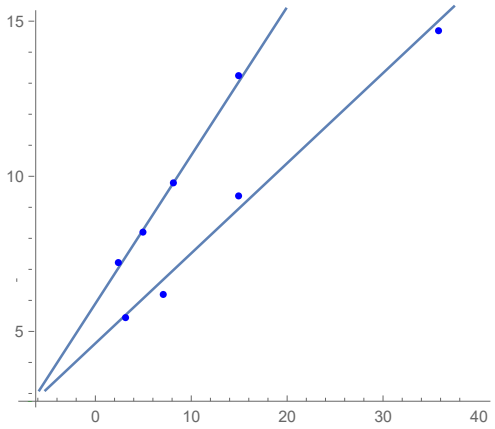


Abbildung 7.1: Bestimmung der Geraden mit einer SVD

Ein Gleichungssystem der Form $A \cdot x = 0 \mid A \in \mathbb{R}^{mxn}$ lässt sich lösen, falls der Rang der Matrix $A \leq n$ ist. Ist der Rang der Matrix $A > n$, wird für $x \in \mathbb{R}^n$, gilt das Gleichungssystem als überbestimmt und es wird derjenige Wert gesucht, für den $\|Ax\|$ minimal ist. Ist x die gesuchte Lösung, so sind auch alle Vielfachen $k \cdot x$ Lösungen und die Lösungsmenge kann beschränkt werden auf $x = \|1\|$. Die Singulärwertszerlegung ist ein mathematisches Verfahren, das bei der Problemlösung hilft und bei der eine nummerische

Matrix $A \in \mathbb{R}^{mxn}$ zerlegt werden kann.

$$A \cdot x = U \cdot D \cdot V^T \cdot x \tag{7.1}$$

$$A \cdot x = U^{mxn} \cdot D^{nxn} \cdot V^{nxn} \cdot x \tag{7.2}$$

Für die Matrix $U = \begin{bmatrix} u_1 & u_2 & u_3 & ... & u_n \end{bmatrix}$ gilt, $\|u_i\| = 1$ und durch die Orthogonalität der Spalten gilt auch $u_i \cdot u_j = 0$ für $i \neq j$. Damit kann der Ausdruck 7.2 geschrieben werden als

$$\|UDV^T x\| \; = \; \|DV^T x\|$$

und die Gleichung

$$\|V^T x\| \; = \; \|x\|$$

aufgestellt werden, da V orthogonal ist und somit auch V^T.
$V^T x$ wird nun definiert mit

$$z \; = \; V^T x. \tag{7.3}$$

Damit wird nach dem Minimum des Betrags $\|Dz\|$ unter der Bedingung $\|z\| = 1$ gesucht. Da D eine sortierte diagonale Matrix

$$D \; = \; \begin{bmatrix} s_1 & & & \\ & s_2 & & \\ & & ... & \\ & & & s_n \end{bmatrix}$$

ist, deren kleinster Eintrag s_n ist, liefert

$$y = \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix} \tag{7.4}$$

den Minimalwert von $\|Dz\|$. Der Ausdruck 7.3 wird nach x umgestellt

$$x = V \cdot z \tag{7.5}$$

Wird die Gleichung 7.4 in Gleichung 7.5 eingesetzt, ergibt sich

$$x = V \cdot \begin{pmatrix} 0 \\ \vdots \\ 1 \end{pmatrix}$$

Der Wert für x entspricht damit gleich der letzten Spalte von V (vgl. Hardly und Zisserman [7]). Auf diese Weise lassen sich überbestimmte Gleichungssysteme, bei der es mehr Gleichungen als Unbekannte gibt, lösen.

7.2 Installationsanweisungen

7.2.1 Installation von *Mathematica*

Damit die Applikation in *Mathematica* auf dem Computer funktioniert, muss die Software *Mathematica* und die IDE Eclipse heruntergeladen werden. *Mathematica* ist nur lizenziert verfügbar, weshalb ein Aktivierungsschlüssel benötigt wird.
Um mit Eclipse und Mathematica zu arbeiten, wurde von Katharina Ußling eine Installationsanleitung erstellt. Diese Anleitung befindet sich auf

dem angehängt USB-Stick dieser Arbeit (*Installationsanleitung_IDE.pdf*).

7.2.2 Installation von *Python*

Für die *Python*-Installation wurde Version 3 verwendet, welche unter dem Link `https://www.python.org/downloads/` verfügbar ist. Öffnet sich nach dem Download der Windows Installer, sollte zur Einfachheit beachtet werden, dass die Checkbox *Add Python to PATH* angekreuzt ist. Damit wird bestätigt, dass der Pfad zum Python-Interpreter zu den Umgebungsvariablen von Windows hinzugefügt wird. Damit ist Python nicht nur lokal, sondern auch global auf dem Computer ausführbar. Um zu prüfen, ob Python auf dem Windows-PC installiert wurde, kann in der Eingabeaufforderung der Befehl `py` ausgeführt werden, der bei erfolgreicher Installation die derzeitige Python-Version angibt. Eine einfache Möglichkeit, um benötigte Bibliotheken wie `sympy` herunterzuladen, ist das Packetverwaltungssystem *PIP*, das mittels dem Befehl python get−pip.py installiert werden kann. Damit kann mit der Eingabeaufforderung pip install Paketname jedes Python-Paket heruntergeladen werden.

7.3 Beispiel einer Ausgabe der *Mathematica*-Datei bei ungestörten Punkten

Auf der nächsten Seite befindet sich die Ausgabedatei, die bei der Anwendung erzeugt wurde.

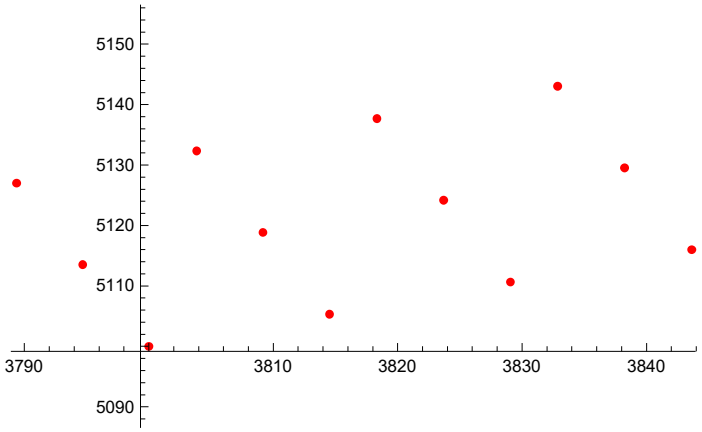
In[388]:=
```
Get["Calibration`CalibrationMain`"]
```
erhalte
```
Winkelliste =
  {{9, 12, 18, {1000, 500, 4000}}, {23, 34, 3, {0, 0, 5000}}, {12, 4, 13, {0, 0, 7000}}};
f = 9; H = {12, 18}; (* in millimeters*)
Gitterkonstante = 25; (* in millimeter*)
Zeilen = 3; Spalten = 4;
pixelPitch = 3.75; (*camera model: canon eos 6D in millimeter*)
```
leite ab
```
(*"Ungestoert" gibt die Liste der Punkte an,die nicht gestoert wurde,
"Gestoert" gibt die Liste der Punkte an,die gestoert wurden*)
Quadrat = "Ungestoert";
startCalibration[Winkelliste,
  Gitterkonstante, Zeilen, Spalten, Quadrat, pixelPitch, f, H];
```

-----START DER BERECHNUNG DER PROJIZIERTEN OBJEKTPUNKTEN------

3x4-ProjektionsMatrix:
$$\begin{pmatrix} 2400. & 0. & 3200. & 0. \\ 0. & 2400. & 4800. & 0. \\ 0. & 0. & 1. & 0. \end{pmatrix}$$

-----Start der Berechnung der Sensorpunkte------

Drehmatrix:
$$\begin{pmatrix} 0.930274 & -0.302264 & 0.207912 & 0. \\ 0.336145 & 0.929297 & -0.153016 & 0. \\ -0.14696 & 0.212235 & 0.966105 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$$

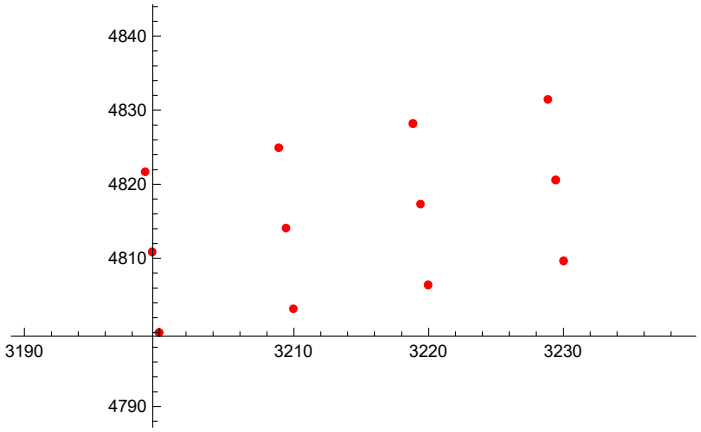


Sensorpunkte, Ungestoert :
$$\begin{pmatrix} \begin{pmatrix} 3800. \\ 5100. \\ 1. \end{pmatrix} & \begin{pmatrix} 3814.52 \\ 5105.32 \\ 1. \end{pmatrix} & \begin{pmatrix} 3829.06 \\ 5110.66 \\ 1. \end{pmatrix} & \begin{pmatrix} 3843.64 \\ 5116. \\ 1. \end{pmatrix} \\ \begin{pmatrix} 3794.68 \\ 5113.52 \\ 1. \end{pmatrix} & \begin{pmatrix} 3809.17 \\ 5118.85 \\ 1. \end{pmatrix} & \begin{pmatrix} 3823.69 \\ 5124.19 \\ 1. \end{pmatrix} & \begin{pmatrix} 3838.24 \\ 5129.54 \\ 1. \end{pmatrix} \\ \begin{pmatrix} 3789.37 \\ 5127.01 \\ 1. \end{pmatrix} & \begin{pmatrix} 3803.84 \\ 5132.34 \\ 1. \end{pmatrix} & \begin{pmatrix} 3818.34 \\ 5137.69 \\ 1. \end{pmatrix} & \begin{pmatrix} 3832.86 \\ 5143.04 \\ 1. \end{pmatrix} \end{pmatrix}$$

normierter Normalenvektor der Ebene: {0.207912, -0.153016, 0.966105, 0.}

-----Start der Berechnung der Sensorpunkte------

Drehmatrix:
$$\begin{pmatrix} 0.827901 & -0.0433885 & 0.559193 & 0. \\ 0.26637 & 0.907808 & -0.323931 & 0. \\ -0.493585 & 0.417135 & 0.763133 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$$

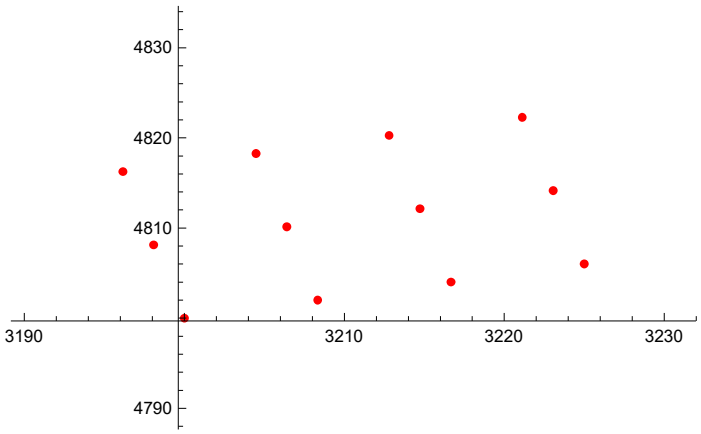


Sensorpunkte, Ungestoert :
$$\begin{pmatrix} \begin{pmatrix} 3200. \\ 4800. \\ 1. \end{pmatrix} & \begin{pmatrix} 3209.96 \\ 4803.2 \\ 1. \end{pmatrix} & \begin{pmatrix} 3219.97 \\ 4806.42 \\ 1. \end{pmatrix} & \begin{pmatrix} 3230.03 \\ 4809.66 \\ 1. \end{pmatrix} \\ \begin{pmatrix} 3199.48 \\ 4810.87 \\ 1. \end{pmatrix} & \begin{pmatrix} 3209.42 \\ 4814.1 \\ 1. \end{pmatrix} & \begin{pmatrix} 3219.4 \\ 4817.34 \\ 1. \end{pmatrix} & \begin{pmatrix} 3229.44 \\ 4820.59 \\ 1. \end{pmatrix} \\ \begin{pmatrix} 3198.96 \\ 4821.7 \\ 1. \end{pmatrix} & \begin{pmatrix} 3208.88 \\ 4824.94 \\ 1. \end{pmatrix} & \begin{pmatrix} 3218.84 \\ 4828.2 \\ 1. \end{pmatrix} & \begin{pmatrix} 3228.86 \\ 4831.48 \\ 1. \end{pmatrix} \end{pmatrix}$$

normierter Normalenvektor der Ebene: {0.559193, -0.323931, 0.763133, 0.}

-----Start der Berechnung der Sensorpunkte------

Drehmatrix:
$$\begin{pmatrix} 0.971997 & -0.224403 & 0.0697565 & 0. \\ 0.234167 & 0.949815 & -0.207405 & 0. \\ -0.0197134 & 0.217932 & 0.975765 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$$



Sensorpunkte, Ungestoert :
$$\begin{pmatrix} \begin{pmatrix} 3200. \\ 4800. \\ 1. \end{pmatrix} & \begin{pmatrix} 3208.33 \\ 4802.01 \\ 1. \end{pmatrix} & \begin{pmatrix} 3216.67 \\ 4804.01 \\ 1. \end{pmatrix} & \begin{pmatrix} 3225. \\ 4806.02 \\ 1. \end{pmatrix} \\ \begin{pmatrix} 3198.08 \\ 4808.13 \\ 1. \end{pmatrix} & \begin{pmatrix} 3206.4 \\ 4810.14 \\ 1. \end{pmatrix} & \begin{pmatrix} 3214.73 \\ 4812.15 \\ 1. \end{pmatrix} & \begin{pmatrix} 3223.06 \\ 4814.15 \\ 1. \end{pmatrix} \\ \begin{pmatrix} 3196.16 \\ 4816.26 \\ 1. \end{pmatrix} & \begin{pmatrix} 3204.48 \\ 4818.26 \\ 1. \end{pmatrix} & \begin{pmatrix} 3212.8 \\ 4820.27 \\ 1. \end{pmatrix} & \begin{pmatrix} 3221.12 \\ 4822.27 \\ 1. \end{pmatrix} \end{pmatrix}$$

normierter Normalenvektor der Ebene: {0.0697565, -0.207405, 0.975765, 0.}

-----START DER BERECHNUNG DER HOMOGRAFIEN------

Homographiematrix: : $\begin{pmatrix} 0.0000692757 & -1.81923\times10^{-6} & 0.597481 \\ 3.98341\times10^{-6} & 0.000127713 & 0.801883 \\ -5.77672\times10^{-9} & 8.34254\times10^{-9} & 0.000157232 \end{pmatrix}$

Homographiematrix: : $\begin{pmatrix} 0.0000141272 & 0.0000426668 & 0.5547 \\ -0.0000599742 & 0.00014495 & 0.83205 \\ -1.7112\times10^{-8} & 1.44616\times10^{-8} & 0.000173344 \end{pmatrix}$

Homographiematrix: : $\begin{pmatrix} 0.0000562057 & 3.93278\times10^{-6} & 0.5547 \\ 0.0000115738 & 0.0000823539 & 0.83205 \\ -4.88171\times10^{-10} & 5.39673\times10^{-9} & 0.000173344 \end{pmatrix}$

-----START DER BERECHNUNG DER KAMERA MATRIX------

Eigenvektor der omega-Matrix: $\begin{pmatrix} 1. \\ 2.56148\times10^{-8} \\ 3.77923\times10^{-9} \end{pmatrix}$

Die ermittelte Kameramatrix: $\begin{pmatrix} 2400. & -1.09938\times10^{-7} & 3200. \\ 0. & 2400. & 4800. \\ 0. & 0. & 1. \end{pmatrix}$

Die ermittelte Kameramatrix in mm: $\begin{pmatrix} 9. & -4.12268\times10^{-10} & 12. \\ 0. & 9. & 18. \\ 0. & 0. & 1. \end{pmatrix}$

Brennweite: 9. mm. Hauptpunkt liegt bei : H(12.18.)

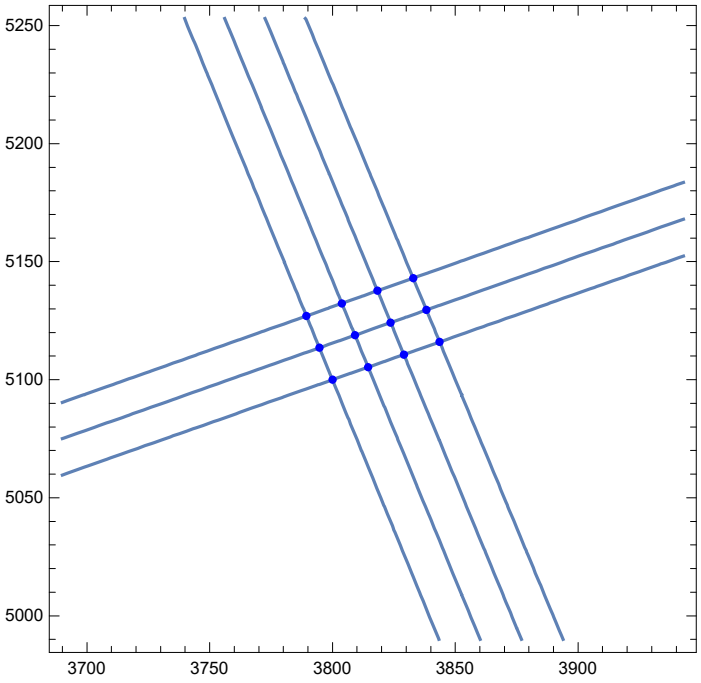
-----START DER REKONSTRUKTION DER OBJEKTPUNKTE------

-----START DER BERECHNUNG DER FLUCHTPUNKTE------

Loesung - Schnittpunkt der Projektionspunkte horizontal: {-11992.2, -689.563, 1.}

Loesung - Schnittpunkt der Projektionspunkte vertikal: {-218.066, 15308.7, 1.}

Zeichnung der Geraden des Schachbretts:



-----START DER BERECHNUNG DES NORMALENVEKTORS------

Berechnung des Richtungsvektoren 1: {-3418.07, 10508.7, 2400.}

Berechnung des Richtungsvektoren 2: {-15192.2, -5489.56, 2400.}

genormter Normalenvektor der beiden Richtungsvektoren: {0.207912, -0.153016, 0.966105}

-----START DER BERECHNUNG DER REKONSTRUKTIONSPUNKTE------

ursprüngliche Objektpunkte:
$$\begin{pmatrix} \begin{pmatrix}1000.\\500.\\4000.\\1.\end{pmatrix} & \begin{pmatrix}1023.26\\508.404\\3996.33\\1.\end{pmatrix} & \begin{pmatrix}1046.51\\516.807\\3992.65\\1.\end{pmatrix} & \begin{pmatrix}1069.77\\525.211\\3988.98\\1.\end{pmatrix} \\ \begin{pmatrix}992.443\\523.232\\4005.31\\1.\end{pmatrix} & \begin{pmatrix}1015.7\\531.636\\4001.63\\1.\end{pmatrix} & \begin{pmatrix}1038.96\\540.04\\3997.96\\1.\end{pmatrix} & \begin{pmatrix}1062.21\\548.443\\3994.28\\1.\end{pmatrix} \\ \begin{pmatrix}984.887\\546.465\\4010.61\\1.\end{pmatrix} & \begin{pmatrix}1008.14\\554.868\\4006.94\\1.\end{pmatrix} & \begin{pmatrix}1031.4\\563.272\\4003.26\\1.\end{pmatrix} & \begin{pmatrix}1054.66\\571.676\\3999.59\\1.\end{pmatrix} \end{pmatrix}$$

rekonstruierte Objektpunkte:
$$\begin{pmatrix} \begin{pmatrix}1000.\\500.\\4000.\\1\end{pmatrix} & \begin{pmatrix}1023.26\\508.404\\3996.33\\1\end{pmatrix} & \begin{pmatrix}1046.51\\516.807\\3992.65\\1\end{pmatrix} & \begin{pmatrix}1069.77\\525.211\\3988.98\\1\end{pmatrix} \\ \begin{pmatrix}992.443\\523.232\\4005.31\\1\end{pmatrix} & \begin{pmatrix}1015.7\\531.636\\4001.63\\1\end{pmatrix} & \begin{pmatrix}1038.96\\540.04\\3997.96\\1\end{pmatrix} & \begin{pmatrix}1062.21\\548.443\\3994.28\\1\end{pmatrix} \\ \begin{pmatrix}984.887\\546.465\\4010.61\\1\end{pmatrix} & \begin{pmatrix}1008.14\\554.868\\4006.94\\1\end{pmatrix} & \begin{pmatrix}1031.4\\563.272\\4003.26\\1\end{pmatrix} & \begin{pmatrix}1054.66\\571.676\\3999.59\\1\end{pmatrix} \end{pmatrix}$$

-----START DER BERECHNUNG DER EXTRINSISCHEN WERTE------

Winkel α: 9. Grad

Winkel β: 12. Grad

Winkel γ: 18. Grad

Translationsvektor t:
$$\begin{pmatrix}1000.\\500.\\4000.\\1\end{pmatrix}$$

-----START DER BERECHNUNG DES REPROJEKTIONSFEHLERS------

3x4 Projektionsmatrix:
$$\begin{pmatrix}2400. & 0 & 3200. & 0\\ 0. & 2400. & 4800. & 0\\ 0. & 0. & 1. & 0\end{pmatrix}$$

Reprojektionsfehler: 4.20268×10^{-8}

Reprojektionsfehler quadratisch: 1.47278×10^{-16}

-----START DER BERECHNUNG DES FEHLERS ZWISCHEN
URSPRÜNGLICHEN UND BERECHNETEN OBJEKTPUNKTEN------

quadratischer Fehler zwischen ursprünglichen
Objektpunkten und rekonstruierten Objektpunkten: 1.74886×10^{-11}

7.4 Beispiel einer Ausgabe der *Mathematica*-Datei bei gestörten Punkten

Auf der nächsten Seite befindet sich die Ausgabedatei, bei der mit gestörten Punkten gerechnet wird. In diesem sind noch einmal alle Algorithmen implementiert, die die Optimierung des Bündelblockausgleichs durchführen.

In[19]:=
```
Get["Calibration`CalibrationMain`"]
```
erhalte
```
Winkelliste =
  {{9, 12, 18, {1000, 500, 4000}}, {23, 34, 3, {0, 0, 2000}}, {12, 4, 13, {0, 0, 3000}}};
f = 9; H = {12, 18}; (* in millimeters*)
Gitterkonstante = 25; (* in millimeter*)
Zeilen = 3; Spalten = 4;
pixelPitch = 3.75; (*camera model: canon eos 6D in millimeter*)
```
leite ab
```
(*"Ungestoert" gibt die Liste der Punkte an,die nicht gestoert wurde,
"Gestoert" gibt die Liste der Punkte an,die gestoert wurden*)
Quadrat = "Gestoert";
startCalibration[Winkelliste,
  Gitterkonstante, Zeilen, Spalten, Quadrat, pixelPitch, f, H];
```

-----START DER BERECHNUNG DER PROJIZIERTEN OBJEKTPUNKTEN------

3x4-ProjektionsMatrix: $\begin{pmatrix} 2400. & 0. & 3200. & 0. \\ 0. & 2400. & 4800. & 0. \\ 0. & 0. & 1. & 0. \end{pmatrix}$

-----Start der Berechnung der Sensorpunkte------

Drehmatrix: $\begin{pmatrix} 0.930274 & -0.302264 & 0.207912 & 0. \\ 0.336145 & 0.929297 & -0.153016 & 0. \\ -0.14696 & 0.212235 & 0.966105 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$

normierter Normalenvektor der Ebene: {0.207912, -0.153016, 0.966105, 0.}

-----Start der Berechnung der Sensorpunkte------

Drehmatrix: $\begin{pmatrix} 0.827901 & -0.0433885 & 0.559193 & 0. \\ 0.26637 & 0.907808 & -0.323931 & 0. \\ -0.493585 & 0.417135 & 0.763133 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$

normierter Normalenvektor der Ebene: {0.559193, -0.323931, 0.763133, 0.}

-----Start der Berechnung der Sensorpunkte------

Drehmatrix: $\begin{pmatrix} 0.971997 & -0.224403 & 0.0697565 & 0. \\ 0.234167 & 0.949815 & -0.207405 & 0. \\ -0.0197134 & 0.217932 & 0.975765 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$

normierter Normalenvektor der Ebene: {0.0697565, -0.207405, 0.975765, 0.}

-----START DER BERECHNUNG DER HOMOGRAFIEN------

Homographiematrix: : $\begin{pmatrix} 0.0000689623 & -2.26488\times10^{-6} & 0.597481 \\ 3.56514\times10^{-6} & 0.000127107 & 0.801883 \\ -5.85812\times10^{-9} & 8.22497\times10^{-9} & 0.000157232 \end{pmatrix}$

Homographiematrix: : $\begin{pmatrix} 0.0000354678 & 0.000106845 & 0.5547 \\ -0.000149713 & 0.000362645 & 0.83205 \\ -4.27348\times10^{-8} & 3.62089\times10^{-8} & 0.000173344 \end{pmatrix}$

Homographiematrix: : $\begin{pmatrix} 0.000131152 & 9.31365\times10^{-6} & 0.5547 \\ 0.0000270108 & 0.000192367 & 0.83205 \\ -1.13766\times10^{-9} & 1.26354\times10^{-8} & 0.000173344 \end{pmatrix}$

-----START DER BERECHNUNG DER KAMERA MATRIX------

Eigenvektor der omega-Matrix: $\begin{pmatrix} 1. \\ 2.55266\times10^{-8} \\ 3.88813\times10^{-9} \end{pmatrix}$

Die ermittelte Kameramatrix: $\begin{pmatrix} 2429.02 & 5.26984 & 3176.71 \\ 0. & 2427.84 & 4820.64 \\ 0. & 0. & 1. \end{pmatrix}$

Die ermittelte Kameramatrix in mm: $\begin{pmatrix} 9.1066 & 0.0197619 & 11.9127 \\ 0. & 9.1066 & 18.0774 \\ 0. & 0. & 1. \end{pmatrix}$

emt

Brennweite: 9.1066 mm. Der Hauptpunkt liegt bei : H(11.9127 , 18.0774)

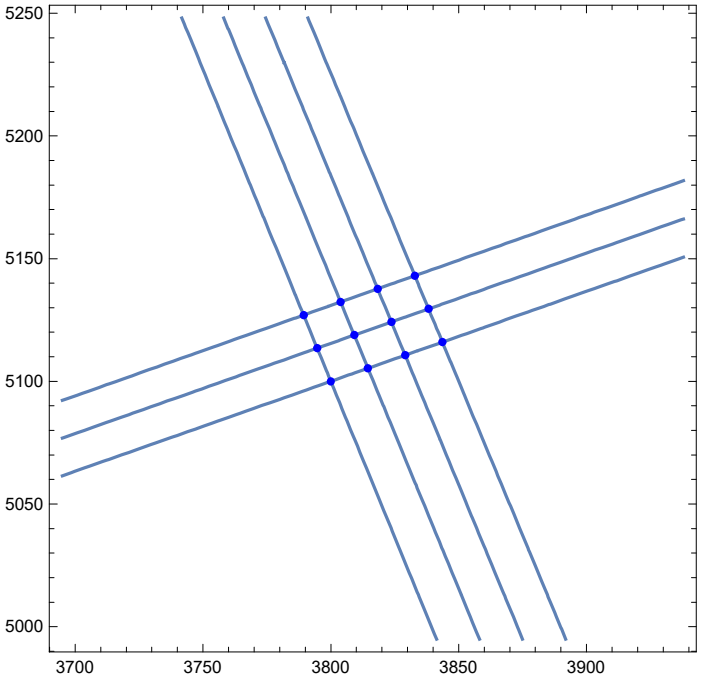
-----START DER REKONSTRUKTION DER OBJEKTPUNKTE------

-----START DER BERECHNUNG DER FLUCHTPUNKTE------

Loesung - Schnittpunkt der Projektionspunkte horizontal: {-11541.4, -523.787, 1.}

Loesung - Schnittpunkt der Projektionspunkte vertikal: {-245.101, 15377.3, 1.}

Zeichnung der Geraden des Schachbretts:



-----START DER BERECHNUNG DES NORMALENVEKTORS------

Berechnung des Richtungsvektoren 1: {-3421.81, 10556.7, 2429.02}

Berechnung des Richtungsvektoren 2: {-14718.1, -5344.43, 2429.02}

genormter Normalenvektor der beiden Richtungsvektoren: {0.214567, -0.152431, 0.964741}

-----START DER BERECHNUNG DER REKONSTRUKTIONSPUNKTE------

-----START DER BERECHNUNG DER EXTRINSISCHEN WERTE------

Winkel α: 8.9786 Grad

Winkel β: 12.3901 Grad

{1032.5, 489.343, 4058.39, 1}{1040.14, 466.192, 4053.52, 1}

{0., 24.6937, 3.90164}{-7.64279, 23.1512, 4.86461}

Winkel γ: 18.128 Grad

Translationsvektor t: $\begin{pmatrix} 1040.14 \\ 466.192 \\ 4053.52 \\ 1 \end{pmatrix}$

-----START DER BERECHNUNG DES REPROJEKTIONSFEHLERS------

3x4 Projektionsmatrix: $\begin{pmatrix} 2428.43 & 0 & 3176.71 & 0 \\ 0. & 2428.43 & 4820.64 & 0 \\ 0. & 0. & 1. & 0 \end{pmatrix}$

Reprojektionsfehler: 2.05617

Reprojektionsfehler quadratisch: 0.352529

-----START DER BERECHNUNG DES FEHLERS ZWISCHEN
URSPRÜNGLICHEN UND BERECHNETEN OBJEKTPUNKTEN------

Objektreferenzfehler: 66 947.8

-----START DER OPTIMIERUNG DER WERTE------

Die gewählte Gitterkonstante beträgt 25 mm.

Die 'Ground Truth'- Werte für die Kameramatrix: $\begin{pmatrix} 2400 & 0 & 3200 \\ 0 & 2400 & 4800 \\ 0 & 0 & 1 \end{pmatrix}$

Die 'Ground Truth'- Werte für die extrinsischen

Parameter: α: 9 Grad, β: 12 Grad, γ: 18 Grad, t = $\begin{pmatrix} 1000 \\ 500 \\ 4000 \end{pmatrix}$.

Initiale KameraMatrix: $\begin{pmatrix} 2429.02 & 5.26984 & 3176.71 \\ 0. & 2427.84 & 4820.64 \\ 0. & 0. & 1. \end{pmatrix}$

Initiale Werte für α: 8.9786, β: 12.3901, γ: 18.128, t: $\begin{pmatrix} 1040.14 \\ 466.192 \\ 4053.52 \\ 1 \end{pmatrix}$

-----START DER OPTIMIERUNG FINDMINIMUM MIT f = (f1 + f2) / 2, skew = 0------ nach LevenbergMarquardt

---Werte werden berechnet---

-----OPTIMIERTE WERTE------

Reprojektionsfehler nach Minimierung: 2.2289×10^{-11}

Die berechnete Kameramatrix: $\begin{pmatrix} 2276.1 & 0 & 3216.16 \\ 0 & 2276.1 & 4775.69 \\ 0 & 0 & 1 \end{pmatrix}$

Berechnete Werte nach Minimierung für α: 8.28947, β: 11.3704, γ: 18.1198, t: $\begin{pmatrix} 973.911 \\ 540.989 \\ 3796.82 \end{pmatrix}$

-----START DER BERECHNUNG DER PROJIZIERTEN OBJEKTPUNKTEN------

3x4-ProjektionsMatrix: $\begin{pmatrix} 2276.1 & 0 & 3216.16 & 0 \\ 0 & 2276.1 & 4775.69 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

-----Start der Berechnung der Sensorpunkte------

Drehmatrix: $\begin{pmatrix} 0.931755 & -0.304902 & 0.197151 & 0. \\ 0.334771 & 0.931638 & -0.141345 & 0. \\ -0.140577 & 0.197699 & 0.970131 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$

normierter Normalenvektor der Ebene: {0.197151, -0.141345, 0.970131, 0.}

-----START DER BERECHNUNG DES FEHLERS ZWISCHEN
URSPRÜNGLICHEN UND BERECHNETEN OBJEKTPUNKTEN------

Objektreferenzfehler: 524 320.

-----START DER FINDMINIMUM OPTIMIERUNG
MIT f1 != f2, skew != 0------ nach LevenbergMarquardt

---Werte werden berechnet---

-----OPTIMIERTE WERTE------

Reprojektionsfehler nach Minimierung: 3.10177×10^{-11}

Die berechnete Kameramatrix: $\begin{pmatrix} 2395.87 & 8.24497 & 3176.98 \\ 0 & 2393.01 & 4819.17 \\ 0 & 0 & 1 \end{pmatrix}$

Berechnete Werte nach Minimierung für α: 8.56928, β: 11.9259, γ: 18.2139, t: $\begin{pmatrix} 1038.86 \\ 469.562 \\ 4001.24 \end{pmatrix}$

-----START DER BERECHNUNG DER PROJIZIERTEN OBJEKTPUNKTEN------

3x4-ProjektionsMatrix: $\begin{pmatrix} 2395.87 & 8.24497 & 3176.98 & 0 \\ 0 & 2393.01 & 4819.17 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

-----Start der Berechnung der Sensorpunkte------

Drehmatrix: $\begin{pmatrix} 0.929393 & -0.305819 & 0.206647 & 0. \\ 0.338325 & 0.929667 & -0.145789 & 0. \\ -0.147528 & 0.205409 & 0.967493 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$

normierter Normalenvektor der Ebene: {0.206647, -0.145789, 0.967493, 0.}

-----START DER BERECHNUNG DES FEHLERS ZWISCHEN URSPRÜNGLICHEN UND BERECHNETEN OBJEKTPUNKTEN------

Objektreferenzfehler: 29074.

-----START DER NMINIMIZE OPTIMIERUNG MIT f1 != f2, skew != 0------

---Werte werden berechnet---

-----OPTIMIERTE WERTE------

Reprojektionsfehler nach Minimierung: 6.07551×10^{-11}

Die berechnete Kameramatrix: $\begin{pmatrix} 2431.06 & 9.4582 & 3156.2 \\ 0 & 2427.05 & 4839.76 \\ 0 & 0 & 1 \end{pmatrix}$

Berechnete Werte nach Minimierung für α: 8.51086, β: 12.4498, γ: 18.2177, t: $\begin{pmatrix} 1075.07 \\ 435.972 \\ 4065.97 \end{pmatrix}$

-----START DER BERECHNUNG DER PROJIZIERTEN OBJEKTPUNKTEN------

3x4-ProjektionsMatrix: $\begin{pmatrix} 2431.06 & 9.4582 & 3156.2 & 0 \\ 0 & 2427.05 & 4839.76 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$

-----Start der Berechnung der Sensorpunkte------

Drehmatrix: $\begin{pmatrix} 0.92754 & -0.305277 & 0.215584 & 0. \\ 0.339492 & 0.929441 & -0.144517 & 0. \\ -0.156255 & 0.207234 & 0.965732 & 0. \\ 0. & 0. & 0. & 1. \end{pmatrix}$

Eidesstattliche Erklärung

Ich versichere, dass ich die vorstehende Arbeit selbstständig verfasst und hierzu keine anderen als die angegebenen Hilfsmittel verwendet habe. Alle Stellen der Arbeit die wörtlich oder sinngemäß aus fremden Quellen entnommen wurden, sind als solche kenntlich gemacht.

Die Arbeit wurde bisher in gleicher oder ähnlicher Form in keinem anderen Studiengang als Prüfungsleistung vorgelegt oder an anderer Stelle veröffentlicht.

Ich bin mir bewusst, dass eine falsche Erklärung rechtliche Folgen haben kann.

Furtwangen, 28.02.2022, Sabine Schleise

Abbildungsverzeichnis

2.1 Intrinsische und extrinsische Kameraparameter (vgl. [4]) . 4
2.2 Extrinsische Kameraparameter (vgl. [2]) 4
3.1 Zweidimensionale Konstruktion der Perspektive 9
3.2 Projektionsbild eines Schachbrettmusters (vgl. [11]) 10
3.3 Das Lochkameramodell 11
3.4 Einfaches Lochkameramodell in Positivlage (vgl. [7]) 12
3.5 Koordinaten von H bezüglich des Bildkoordinatensystems . 13
3.6 Verzeichnung abhängig von der Blendenlage 15
4.1 Formen eines Kegelschnitts 18
4.2 Homografie 21
4.3 Lage der Objektpunkte 26
4.4 Geraden und Fluchtpunkte eines Schachbretts 28
4.5 Ermittlung der Richtungsvektoren 29
4.6 Lage der Objektpunkte (vgl. [15]) 30
4.7 Translationsvektor t 34
4.8 Drehung D mit dem Normalenvektor $\vec{n}$ 35
4.9 Projektion von n_0 in die y-z-Ebene 36
4.10 Bild mit Reprojektionsfehler 38
5.1 Lochkameramodell für die Modellierung 44
5.2 Ablauf bei der Projektion der Objektpunkte 45
5.3 Projiziertes Schachbrett aus den Werten der Gleichung 5.3 47
5.4 Rekonstruktion der Objektpunkte 57
5.5 Störungen der Sensorpunkte bei $z = 10$ Meter 61
5.6 Rekonstruktion der Objektpunkte mit gestörten Sensorpunkten 62
5.7 Auswirkungen verschiedener Kameraparameter 64
5.8 Bündelblockausgleich mit $\overline{f}$ und $s = 0$ 66
5.9 Bündelblockausgleich mit 11 Freiheitsgraden 67
5.10 Bündelblockausgleich mit der NMinimize-Funktion 69
5.11 Projizierte Sensorpunkte in *Python* 72
5.12 Konsolenausgabe in *Python* 75
7.1 Bestimmung der Geraden mit einer SVD 79

Abkürzungsverzeichnis

IDE Entwicklungsumgebung

AKG Allgemeine Koordinatengleichung

AKF Allgemeine Koordinatenform

SVD Singulärwertszerlegung

IAC Image of absolute Conic

APP Application Project

CAS Computeralgebrasystem

AC Absolute Conic

Literaturverzeichnis

[1] Autodesk. *Was ist Photogammetrie?* Eingesehen am 25.12.2021. [Online]. Available: https://www.autodesk.de/solutions/photogrammetry-software

[2] MathWorks. *What is Camera Calibration?* Eingesehen am 07.01.2022. [Online]. Available: https://de.mathworks.com/help/vision/ug/camera-calibration.html

[3] M. Schulze. (2013) *Vergleich verschiedener Programme zur Kalibrierung von Kameras.* Eingesehen am 20.01.2022. [Online]. Available: https://wwwpub.zih.tu-dresden.de/~photo/Abschlussarbeiten/2013/DA_Mandy_Schulze_2013_Kamerakalibrierung.pdf

[4] U. College London. *Intrinsic camera parameters calibration.* Eingesehen am 17.01.2022. [Online]. Available: https://mphy0026.readthedocs.io/en/latest/calibration/camera_calibration.html

[5] E. Rodner and H. Süße, *Bildverarbeitung und Objekterkennung.* Springer Fachmedien Wiesbaden, 2014.

[6] B. Möller, "*Ein Ansatz zur ikonischen Repräsentation von Bilddaten aktiver Kameras*," Doktorarbeit, Halle-Wittenberg, 2005, eingesehen am 07.01.2022. [Online]. Available: https://sundoc.bibliothek.uni-halle.de/diss-online/05/05H124/prom.pdf

[7] R. Hartley and A. Zissermann, *Multiple view geometry in computer vision.* Cambridge UK and New York: Cambridge University Press, 2003.

[8] O. Schreer, *Stereoanalyse und Bildsynthese.* Berlin Heidelberg: Springer-Verlag, 2005.

[9] L. Halbeisen and N. Hungerbühler. (2016) *Einführung in die projektive Geometrie.* Eingesehen am 03.01.2022. [Online]. Available: https://math.ch/TMU2016/ProjektiveGeometrie.pdf

[10] D. van Starten and O. Labs, "*Die Reele Projektive Ebene*," 2002, eingesehen am 03.01.2022. [Online]. Available: http://www.math.fu-berlin.de/altmann/LEHRE/xx19_WS_MathEntd/P2_vanStratenLabs.pdf

[11] J. L. Mundy and A. Zisserman, "*Appendix - Projective Geometry for Machine Vision*," 1992.

[12] *Verzeichnung*. Eingesehen am 06.02.2022. [Online]. Available: https://www.spektrum.de/lexikon/physik/verzeichnung/15218

[13] S. Liebscher, *Projektive Geometrie der Ebene: Ein klassischer Zugang mit interaktiver Visualisierung*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2017.

[14] T. Arens, F. Hettlich, C. Karpfinger, U. Kockelkorn, K. Lichtenegger, and H. Stachel, *Mathematik*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2018.

[15] J. Conrad, "*Entwicklung, Implementierung und Validierung eines Systems zur optimierten Kamerakalibrierung und Szenenrekonstruktion*," Bachelor Thesis, Hochschule Furtwangen.

[16] Studyflix. *Definitheit*. [Online]. Available: https://studyflix.de/mathematik/definitheit-1866

[17] J. J. Moré, "The Levenberg-Marquardt algorithm: Implementation and theory," in *Lecture Notes in Mathematics*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1978, pp. 105–116.

[18] *Wolfram Mathematica*. Eingesehen am 30.01.2022. [Online]. Available: https://www.wolfram.com/mathematica/

[19] *Wolfram Notebook-Technologie*. Eingesehen am 04.01.2022. [Online]. Available: https://www.wolfram.com/technologies/nb/index.de.html?footer=lang

[20] K. Ußling, "*Entwicklung einer modularisierten Anwendung zur Kamerakalibrierung mit einem CAS*," Bachelor Thesis, Hochschule Furtwangen.

[21] T. Schneider, S. Piontek, P. Hafen, and N. Hottong, "*Robust autodetection of camera lens distortion parameters: 3rd IEEE International Conference on Consumer Electronics*," 2013.

[22] D. Chen and G. Zhang. *A New Sub-Pixel Detector for X-Corners in Camera Calibration Cargets*. Eingesehen am 23.01.2022. [Online]. Available: http://wscg.zcu.cz/wscg2005/Papers_2005/Short/A43-full.pdf

[23] *RandomVariate*. Eingesehen am 10.01.2022. [Online]. Available: https://reference.wolfram.com/language/ref/RandomVariate.html

[24] *FindMinimum*. Eingesehen am 10.01.2022. [Online]. Available: https://reference.wolfram.com/language/ref/FindMinimum.html

[25] *Python - Programmiersprache*. Eingesehen am 22.07.2021. [Online]. Available: https://www.codecademy.com/catalog/language/python

[26] *Numpy*. Eingesehen am 04.01.2022. [Online]. Available: https://numpy.org/

[27] *SymPy*. Eingesehen am 04.01.2022. [Online]. Available: https://www.sympy.org/en/index.html

[28] L. Tagliafferi. *So arbeiten Sie mit der interaktiven Python-Konsole*. Eingesehen am 04.01.2022. [Online]. Available: https://www.digitalocean.com/community/tutorials/how-to-work-with-the-python-interactive-console

[29] *Jetbrains - PyCharm*. Eingesehen am 16.12.2021. [Online]. Available: https://www.jetbrains.com/de-de/pycharm/

[30] H. Scheid and W. Schwarz, *Elemente der Geometrie*. Berlin Heidelberg: Springer-Verlag, 2017.

[31] R. Frost. *Pixelpitch*. Eingesehen am 02.01.2022. [Online]. Available: https://slr-foto.de/pixel-pitch.htm