
Stereokalibrierung und Szenenrekonstruktion aus heterogenen Bildquellen in Bezug auf Kameras gleicher und unterschiedlicher Auflösungen

Erarbeitet von Studenten und Studentinnen
im Rahmen der Abschlussarbeit
Masterarbeit der Fakultät

Studiengang Semester Max Mustermann 222222

Betreut von: Prof. Dr. Mustermann

[Disclaim here](#)



Fakultät XYZ der Hochschule Furtwangen
Sommersemester - Wintersemester 2015

Inhaltsverzeichnis

1 Einleitung, Motivation - motivation	4
2 Basis Transformationen	5
2.1 Die Transformation von Koordinaten schließt die Transformation der Basis mit ein	5
2.2 Beispielhafte Vorführung von einem definierten Welt- zu einem synthetischen Kamera-koordinatensystem	5
2.3 Übersicht über die benötigten Koordinatensysteme und Transformationen für die Ste- reobildanalyse	11
2.4 Aufbau der Koordinatensysteme	11
3 Homographien in der Ebene	18
3.1 Homographie zwischen den Abbildungen eines Quaders einer Ausgangskamera und einer dazu um das Projektionszentrum rotierten Kamera	19
3.2 Abbildungsunterschiede von Rotationen um ein Projektionszentrum und Rotation um einen beliebigen Drehpunkt von Punkten in der Ebene	26
3.3 Punkte in unterschiedlichen Ebenen	28
3.4 Epipolargeometrie als Grundlage der Stereokalibrierung und Szenenrekonstruktion . .	29
3.5 Geometrische Erläuterung der Fundamentalmatrix und der Essentiellen Matrix	29
4 Minimalbeispiel 3D-Stereokalibrierung und Szenenrekonstruktion bei Kameras gleicher Auflösung	30
4.1 Vorgehen: Projektion eines Quaders in zwei verschiedenen transformierte Kameras	30
4.2 Berechnung der Projektionsmatrizen	32
4.3 Transformation der Weltpunkte in Koordinaten der Koordinatensysteme von beiden Kameras	32
4.4 Berechnung der projizierten Punkte auf den beiden Bildebenen	33
4.5 Umrechnung von Bildkoordinaten in Sensorkoordinaten	34
4.6 Ermitteln der Fundamentalmatrix mit Hilfe des 8-Point-Algorithms	34
4.7 Berechnen der Epipole und Epipolgeraden mit der Fundamentalmatrix	35
4.7.1 Konstruktion der Epipole und der Epipolgeraden auf Grundlage der Epipolar-geometrie	36
4.8 Ermitteln der Essentiellen Matrix über die Fundamentalmatrix	38
4.8.1 Ermitteln der Essentiellen Matrix mit normierten Bildkoordinaten und dem 8- Point-Algorithm	38
4.9 Ermitteln der externen Kameraparameter mit Hilfe der Essentiellen Matrix	39
4.10 Punktreakonstruktion durch Triangulation	40
4.10.1 Rektifizierung	40
5 Minimalbeispiel 3D-Stereokalibrierung und Szenenrekonstruktion bei Kameras unter- schiedlicher Auflösung	41
5.1 Vorgehen: Projektion eines Quaders in zwei verschiedenen transformierte Kameras mit unterschiedlichen Auflösungen	41
5.2 Berechnung der Projektionsmatrizen	41
5.3 Transformation der Weltpunkte in Koordinaten der Koordinatensysteme von beiden Kameras	41
5.4 Berechnung der projizierten Punkte auf den beiden Bildebenen	41
5.5 Behauptung 1: Kameras unterschiedlicher Auflösung haben keine Auswirkung auf die Ermittlung der externen Kameraparameter	41

5.6	Ermitteln der Fundamentalmatrix mit Hilfe des 8-Point-Algorithms	41
5.7	Ermitteln der Essentiellen Matrix über die Fundamentalmatrix	41
5.8	Ermitteln der externen Kameraparameter mit Hilfe der Essentiellen Matrix	41
5.9	Behauptung 2: Durch Rektifizierung der Bilder kann eine reelle Triangulation erfolgen	41
5.10	Rektifizierung	41
5.11	Erstellen einer Tiefenkarte aus zwei rektifizierten Bildern	41
5.12	Punkterekonstruktion durch Triangulation	41
5.13	Vergleich der rekonstruierten Szenen	41
6	3D-Stereokalibrierung und Szenenrekonstruktion mit reellen Daten und Kameras gleicher Auflösung	42
6.1	Vorgehen	42
6.2	Aufbau der Set-Ups	46
6.3	Unterschiede im Algorithmus im Vergleich zum Minimalbeispiel	48
6.3.1	normalisierung der eingehenden Daten und Berechnung der Fundamentalmatrix über den normalized 8-Point-Algorithm	48
6.3.2	Ermitteln der Essentiellen Matrix und Einführung des singularity-constraints .	53
6.4	Rekonstruktion der externen Kameraparameter	54
6.5	Szenenrekonstruktion mit Hilfe der Sampson-approximation	56
6.5.1	andere Ansätze für Triangulationsverfahren	56
7	3D-Stereokalibrierung und Szenenrekonstruktion mit reellen Daten und Kameras unterschiedlicher Auflösung	57
7.1	Vorgehen	57
7.2	Aufbau der Set-Ups	57
7.3	Was bedeuten andere Auflösungen für die Belichtung auf dem Sensor	57
7.4	Rekonstruktion der Szene ohne Rektifizierung	57
7.5	Rekonstruktion der Szenen mit Rektifizierung	57
8	Aufbauprojekt- Algorithmus zur Punktesortierung in verzeichneten Schachbrettbildern	58
8.1	Algorithmus zur Punktesortierung in verzeichneten Schachbrettbildern	58
8.1.1	Vorläufiges Klassendiagramm	59
8.1.2	Beispiele	61
8.1.3	Weiteres Vorgehen/ Was fehlt noch	66
9	C3 - ExampleHeader2	69
10	C3 - ExampleHeader3	70
11	Fazit - Conclusion	71
12	Nächste Schritte - next steps	72
13	Protocol - 10.11.2015	73
14	Abkürzungsverzeichnis - List of Abbreviations	74

1 Einleitung, Motivation - motivation

Hier auch erwähnen in welchem Kameramodell wir uns befinden (LITERATUR FINDEN DIE NICHT HZ IST!!)

Ziel ist es auch bestimmte unklarheiten in der Literatur wie Einheiten in rechnungen wie fundamental und essentielle matrix aus der welt zu schaffen.

expliziet darauf eingehen, was es bedeutet zwei gleiche und zwei unterschiedliche Resolutions der Kameras zu besitzen gerade in Bezug auf die Ermittlung von F und E und der Szenenrekonstruktion(LITERATUR FINDEN DIE NICHT HZ IST!!)

2 Basis Transformationen

Um die mathematischen Vorgehensweisen dieser Arbeit verständlicher zu machen, werden in den folgenden Kapiteln zunächst ein paar Grundlegende mathematische Operationen vorgestellt, welche das Grundgerüst der Stereokalibrierung und Szenenrekonstruktion bilden. Als aller erstes werden anhand praxisnaher Beispiele die Notwendigkeit der Basis Transformation und damit einhergehend der Spezialfall einer Transformation von einem Koordinatensystem wie zum Beispiel $K_w = (O, \hat{e}_1, \hat{e}_2, \hat{e}_3)$ in ein anderes Koordinatensystem $K_c = (O_c, \hat{c}_1, \hat{c}_2, \hat{c}_3)$ aufgezeigt. Anzumerken ist, dass grundlegend von orthogonalen Koordinatensystemen ausgegangen wird, sollte dies nicht der Fall sein, wird darauf nochmal explizit hingewiesen.

2.1 Die Transformation von Koordinaten schließt die Transformation der Basis mit ein

Es sei V ein n -Dimensionaler Vektorraum über einem Skalar beziehungsweise Körper K . K stellt in diesem Beispiel das Skalar \mathbb{R} dar, welches alle reellen Zahlen mit einschließt. Zur Veranschaulichung wird der Vektorraum $n = 3$ sprich V^3 also ein 3-Dimesnionaler Raum gewählt, dessen Basis mit $\beta = [\vec{b}_1, \vec{b}_2, \vec{b}_3]$ bezeichnet wird.[1] (FORTFÜHREN)

2.2 Beispielhafte Vorführung von einem definierten Welt- zu einem synthetischen Kamerakoordinatensystem

Anhand eines Beispiels wird die Transformation der Koordinaten noch genauer Veranschaulicht. Es soll kompakt in einer Symbolischen Schreibweise Punkte aus einem Weltkoordinatensystem $K_w = (O, \hat{e}_1, \hat{e}_2, \hat{e}_3)$ in ein Kamerakoordinatensystem $K_c = (O_c, \hat{c}_1, \hat{c}_2, \hat{c}_3)$ überführt werden, welches eine Verschiebung und Rotation zum Weltkoordinatesystem aufweist. Beispielhaft wird dies in Abbildung 2.1. aufgezeigt.

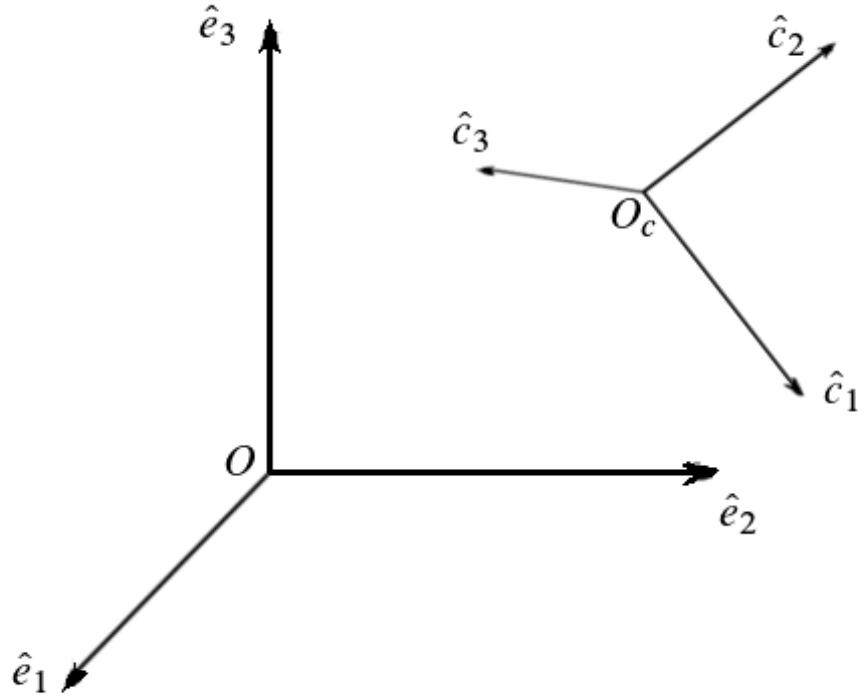


Abbildung 2.1: Weltkoordinatensystem $K_w = (O, \hat{e}_1, \hat{e}_2, \hat{e}_3)$ und Kamerakoordinatensystem $K_c = (O_c, \hat{c}_1, \hat{c}_2, \hat{c}_3)$

Zunächst wird wie im vorherigen Abschnitt beschrieben eine Koordinatisierung von Punkten im Weltkoordinatensystem vorgenommen. Ein Punkt P wird dann wie folgt beschrieben.

$$P = O + p_1\hat{e}_1 + p_2\hat{e}_2 + p_3\hat{e}_3 \quad (2.1)$$

$$\rightsquigarrow (P)_{K_w} = (p_1, p_2, p_3)^T = \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} \quad (2.2)$$

Dieses Koordinatentupel wird dann zum Zweck der Einführung homogener Objekte projektiv erweitert.

$$(P)_{K_w} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix} = \left\{ k \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix} \in \mathbb{R}^4 \mid k \in \mathbb{R} \right\} \quad (2.3)$$

Im Weltkoordinatensystem gilt des Weiteren:

$$\begin{bmatrix} \lambda p_1 \\ \lambda p_2 \\ \lambda p_3 \\ 1 \end{bmatrix} = \begin{bmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{bmatrix} \text{ für } \lambda \neq 0 \quad (2.4)$$

Ein Punkt bezüglich des Kamerakoordinatensystem wird im Vergleich wie folgt beschrieben.

$$K_c = (O_c, \hat{c}_1, \hat{c}_2, \hat{c}_3) \quad (2.5)$$

$${}_c P = O_c + {}_c p_1 \hat{c}_1 + {}_c p_2 \hat{c}_2 + {}_c p_3 \hat{c}_3 \quad (2.6)$$

$$(P)_{K_c} = \begin{pmatrix} {}_c p_1 \\ {}_c p_2 \\ {}_c p_3 \end{pmatrix} \quad (2.7)$$

Zwischen diesen beiden Koordinatensystemen K_w und K_c soll nun eine Beziehung hergestellt werden. Folgende Beziehungsgleichungen werden hierzu aufgestellt.

$$O_c = O + o_{c,1} \hat{e}_1 + o_{c,2} \hat{e}_2 + o_{c,3} \hat{e}_3 \quad (2.8)$$

$$\hat{c}_1 = (c_1)_1 \hat{e}_1 + (c_1)_2 \hat{e}_2 + (c_1)_3 \hat{e}_3 \quad (2.9)$$

$$\hat{c}_2 = (c_2)_1 \hat{e}_1 + (c_2)_2 \hat{e}_2 + (c_2)_3 \hat{e}_3 \quad (2.10)$$

$$\hat{c}_3 = (c_3)_1 \hat{e}_1 + (c_3)_2 \hat{e}_2 + (c_3)_3 \hat{e}_3 \quad (2.11)$$

Diese Beziehungsgleichungen können nun in Gleichung 2.1 eingesetzt werden.

$$\begin{aligned} P = O &+ (o_{c,1} + {}_c p_1 (c_1)_1 + {}_c p_2 (c_2)_1 + {}_c p_3 (c_3)_1) \cdot \hat{e}_1 \\ &+ (o_{c,2} + {}_c p_1 (c_1)_2 + {}_c p_2 (c_2)_2 + {}_c p_3 (c_3)_2) \cdot \hat{e}_2 \\ &+ (o_{c,3} + {}_c p_1 (c_1)_3 + {}_c p_2 (c_2)_3 + {}_c p_3 (c_3)_3) \cdot \hat{e}_3 \end{aligned} \quad (2.12)$$

Hieraus lässt sich ein Gleichungssystem aufstellen und lösen in Form von:

$$\begin{aligned} p_1 &= o_{c,1} + (o_{c,1} + {}_c p_1 (c_1)_1 + {}_c p_2 (c_2)_1 + {}_c p_3 (c_3)_1) \\ \rightsquigarrow p_1 - o_{c,1} &= (o_{c,1} + {}_c p_1 (c_1)_1 + {}_c p_2 (c_2)_1 + {}_c p_3 (c_3)_1) \end{aligned} \quad (2.13)$$

Zu bemerken ist, dass wenn $(P)_{K_c}$ gegeben ist, erhält man auf diese Weise direkt $(P)_{K_w}$. Wenn jedoch $(P)_{K_w}$ gegeben ist, so muss das LGS

$$\begin{bmatrix} (c_1)_1 & (c_2)_1 & (c_3)_1 \\ (c_1)_2 & (c_2)_2 & (c_3)_2 \\ (c_1)_3 & (c_2)_3 & (c_3)_3 \end{bmatrix} \begin{pmatrix} {}_c p_1 \\ {}_c p_2 \\ {}_c p_3 \end{pmatrix} = \begin{pmatrix} p_1 - o_{c,1} \\ p_2 - o_{c,2} \\ p_3 - o_{c,3} \end{pmatrix} \quad (2.14)$$

gelöst werden. Wenn K_c ein kartesisches Koordinatensystem ist, so ist die entstehende koeffizientenmatrix M orthogonal und es gilt $M_c^{-1} = M_c^T$.

$$M_c^T = \begin{bmatrix} (c_1)_1 & (c_1)_2 & (c_1)_3 \\ (c_2)_1 & (c_2)_2 & (c_2)_3 \\ (c_3)_1 & (c_3)_2 & (c_3)_3 \end{bmatrix} \quad (2.15)$$

$$\rightsquigarrow \begin{pmatrix} {}_c p_1 \\ {}_c p_2 \\ {}_c p_3 \end{pmatrix} = M_c^T \begin{pmatrix} p_1 - o_{c,1} \\ p_2 - o_{c,2} \\ p_3 - o_{c,3} \end{pmatrix} \quad (2.16)$$

Handelt es sich um kein kartesisches Koordinatensystem, so muss lediglich die Inverse M_c^{-1} anstatt M_c^T gebildet und wie gehabt verfahren werden. Im Folgenden wird nun noch einmal kompakt und in einer symbolischen Schreibweise die Transformation von Welt- in Kamerakoordinaten festgehalten. Einmal wird wie bereits angefangen in Spaltenvektoren verfahren, einmal wird das selbe Verfahren mit Zeilenvektoren dargestellt. Grund hierfür ist, dass es beide Ansätze funktionieren und zum Beispiel das

Programm *MatLab* mit Spaltenvektoren verfährt, während im entstandenen Code dieser Arbeit mit Zeilenvektoren gearbeitet wurde. Als Hilfestellung und zur Erinnerung gilt, dass $(\hat{e}_1, \hat{e}_2, \hat{e}_3)$ durch Rotation R aus $(\hat{e}_1, \hat{e}_2, \hat{e}_3)$ entstanden ist. Da es sich aber in einem nicht-kartesischen Koordinatensystem nicht um eine orthogonale Drehmatrix handelt wird die Rotation R in diesem Beispiel als Transformationsmatrix C gekennzeichnet. Somit soll gezeigt werden, dass die Transformation unabhängig der Definition ihres ausgehenden Koordinatensystem allgemein formuliert werden kann. Es gilt für die Transformation von Welt- in Kamerakoordinaten ausgehend von Spaltenvektoren folgendes.

$$\begin{pmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \\ O_c \end{pmatrix} = \begin{bmatrix} c_{11} & c_{21} & c_{31} & 0 \\ c_{12} & c_{22} & c_{32} & 0 \\ c_{13} & c_{23} & c_{33} & 0 \\ o_{c,1} & o_{c,2} & o_{c,3} & 1 \end{bmatrix} \begin{pmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \\ O \end{pmatrix} \quad (2.17)$$

$$C^T = C^{-1} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \quad (2.18)$$

$$C = \begin{bmatrix} c_{11} & c_{21} & c_{31} \\ c_{12} & c_{22} & c_{32} \\ c_{13} & c_{23} & c_{33} \end{bmatrix} \quad (2.19)$$

Dementsprechend muss für die Rücktransformation von Kamera in Weltkoordinaten nur wieder die transformierte beziehungsweise die Inverse C^T gebildet werden. Des Weiteren muss der Verschiebungsvektor ebenfalls mit dieser Inversen Multipliziert werden, um diesen ebenfalls in das Zielkoordinatensystem zu überführen.

$$\rightsquigarrow \begin{pmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \\ O \end{pmatrix} = \begin{bmatrix} c_{11} & c_{21} & c_{31} & 0 \\ c_{12} & c_{22} & c_{32} & 0 \\ c_{13} & c_{23} & c_{33} & 0 \\ -(o_{c,1}, o_{c,2}, o_{c,3})C & & & 1 \end{bmatrix} \begin{pmatrix} \hat{e}_1 \\ \hat{e}_2 \\ \hat{e}_3 \\ O_c \end{pmatrix} \quad (2.20)$$

Die Formel 2.21 zeigt die selbe Transformation nur werden die Koordinatensysteme als Zeilenvektoren dargestellt.

$$(\hat{e}_1, \hat{e}_2, \hat{e}_3, O_c) = (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) \cdot \begin{bmatrix} c_{11} & c_{21} & c_{31} & o_{c,1} \\ c_{12} & c_{22} & c_{32} & o_{c,2} \\ c_{13} & c_{23} & c_{33} & o_{c,3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.21)$$

Daraus folgt, dass für den Fall der Rücktransformation gilt:

$$\rightsquigarrow (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) = \begin{bmatrix} c_{11} & c_{12} & c_{13} & & \\ c_{21} & c_{22} & c_{23} & - \begin{pmatrix} o_{c,1} \\ o_{c,2} \\ o_{c,3} \end{pmatrix} C^T & \\ c_{31} & c_{32} & c_{33} & & \\ 0 & 0 & 0 & & 1 \end{bmatrix} (\hat{e}_1, \hat{e}_2, \hat{e}_3, O_c) \quad (2.22)$$

Als Zwischenfazit lässt sich festhalten, dass sich die Matrix zur Beschreibung der Koordinatentransformation aus C und einem Spalten- beziehungsweise Zeilenvektor der Form $\vec{v} = \begin{pmatrix} o_{c,1} \\ o_{c,2} \\ o_{c,3} \end{pmatrix}$ oder $\vec{v} = (o_{c,1} \ o_{c,2} \ o_{c,3})$ zusammensetzt.

$$\begin{bmatrix} c_{11} & c_{21} & c_{31} & 0 \\ c_{12} & c_{22} & c_{32} & 0 \\ c_{13} & c_{23} & c_{33} & 0 \\ -(o_{c,1}, o_{c,2}, o_{c,3})C & & & 1 \end{bmatrix} \quad (2.23)$$

$$\begin{bmatrix} c_{11} & c_{21} & c_{31} & \\ c_{12} & c_{22} & c_{32} & -\begin{pmatrix} o_{c,1} \\ o_{c,2} \\ o_{c,3} \end{pmatrix} C \\ c_{13} & c_{23} & c_{33} & \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.24)$$

Da im implementierten Code dieser Arbeit mit Zeilenvektoren gearbeitet wurde, wird nochmal in symbolischer Schreibweise sämtliche Schritte der Koordinatentransformation im folgenden zusammengefasst.

$$(K) \begin{bmatrix} \begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ 0 \end{pmatrix}_k & \begin{pmatrix} O_c \\ 1 \end{pmatrix}_k \end{bmatrix} = (K_c) \quad (2.25)$$

$$\rightsquigarrow (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) = \begin{bmatrix} C & \vec{v} \\ 0 & 0 & 0 & 1 \end{bmatrix} (\hat{c}_1, \hat{c}_2, \hat{c}_3, O_c) \quad (2.26)$$

Sei im Weltkoordinatensystem ein Punkt $P = (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = (\hat{c}_1, \hat{c}_2, \hat{c}_3, O_c) \begin{pmatrix} cp_1 \\ cp_2 \\ cp_2 \\ 1 \end{pmatrix}$ im Kamerakoordinatensystem, so liefert die Transformation von Welt- in Kamerakoordinatensystem folgendes

$$(\hat{c}_1, \hat{c}_2, \hat{c}_3, O_c) \begin{bmatrix} C & \vec{v} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} cp_1 \\ cp_2 \\ cp_2 \\ 1 \end{pmatrix} = (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} \quad (2.27)$$

Aus der Eindeutigkeit der Koordinatendarstellung folgt:

$$\begin{bmatrix} C & \vec{v} \\ 0 & 0 & 0 & 1 \end{bmatrix} \left(\begin{pmatrix} P_{kc} \\ 1 \end{pmatrix}_k \right) = \left(\begin{pmatrix} P_k \\ 1 \end{pmatrix}_k \right) \quad (2.28)$$

Und umgekehrt resultiert aus der Transformation von Kamera- in Weltkoordinaten wie bereits gezeigt:

$$\rightsquigarrow (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) = \begin{bmatrix} c_{11} & c_{12} & c_{13} & \\ c_{21} & c_{22} & c_{23} & -\begin{pmatrix} o_{c,1} \\ o_{c,2} \\ o_{c,3} \end{pmatrix} C^T \\ c_{31} & c_{32} & c_{33} & \\ 0 & 0 & 0 & 1 \end{bmatrix} (\hat{c}_1, \hat{c}_2, \hat{c}_3, O_c) \quad (2.29)$$

Zur Probe kann folgende Gleichung aufgestellt und auf ihren Wahrheitswert geprüft werden

$$\begin{bmatrix} C^T & -\begin{pmatrix} o_{c,1} \\ o_{c,2} \\ o_{c,3} \end{pmatrix} C^T \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C & o_{c,1} \\ & o_{c,2} \\ 0 & 0 & 1 & o_{c,3} \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.30)$$

In allgemeiner symbolischer Form gilt also

$$\begin{pmatrix} \left(\begin{pmatrix} P_{kc} \\ 1 \end{pmatrix}_k \right) \\ \vec{v} \end{pmatrix} = \begin{bmatrix} C & \vec{v} \\ 0 & 0 & 0 & 1 \end{bmatrix}^{-1} \begin{pmatrix} \left(\begin{pmatrix} P_k \\ 1 \end{pmatrix}_k \right) \\ \vec{v} \end{pmatrix} = \begin{bmatrix} C^{-1} & -\left(C^{-1} \right) \vec{v} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.31)$$

Und falls

$$(\hat{c}_1, \hat{c}_2, \hat{c}_3, O) = (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) \begin{bmatrix} C & \vec{v} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.32)$$

gilt, so ergibt sich

$$\begin{pmatrix} cp_1 \\ cp_2 \\ cp_3 \\ 1 \end{pmatrix} = \begin{bmatrix} C^{-1} & -C^{-1}(O_c)_K \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} \quad (2.33)$$

Somit wurde die symbolischen Transformationsformeln für die Objekte $(O, \hat{e}_1, \hat{e}_2, \hat{e}_3)$ der Koordinatensysteme festgehalten. Jetzt zurück zur Transformation der Koordinatentupel. Aus Gleichung 2.14 folgt:

$$C \begin{pmatrix} cp_1 \\ cp_2 \\ cp_3 \end{pmatrix} = \begin{pmatrix} p_1 - o_{c,1} \\ p_2 - o_{c,2} \\ p_3 - o_{c,3} \end{pmatrix} | C^T \quad (2.34)$$

$$cp := \begin{pmatrix} cp_1 \\ cp_2 \\ cp_3 \end{pmatrix} = C^T \begin{pmatrix} p_1 \\ p_2 \\ p_3 \end{pmatrix} - C^T \begin{pmatrix} o_{c,1} \\ o_{c,2} \\ o_{c,3} \end{pmatrix} \mid \text{projektive Erweiterung} \quad (2.35)$$

$$\begin{bmatrix} cp \\ 1 \end{bmatrix} = \begin{bmatrix} C^T & -C^T o_c \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} P \\ 1 \end{bmatrix} \quad (2.36)$$

Und umgekehrt gilt

$$\begin{bmatrix} P \\ 1 \end{bmatrix} = \begin{bmatrix} C & O_c \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} cp \\ 1 \end{bmatrix} \quad (2.37)$$

Daraus lässt sich also die folgende Aussage ableiten.

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \left[\begin{pmatrix} c_1 \\ 0 \end{pmatrix}_k \quad \begin{pmatrix} c_2 \\ 0 \end{pmatrix}_k \quad \begin{pmatrix} c_3 \\ 0 \end{pmatrix}_k \quad O_c \right] \begin{bmatrix} cp_1 \\ cp_2 \\ cp_3 \\ 1 \end{bmatrix} \quad (2.38)$$

$$\begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ 1 \end{pmatrix} = \left[cp_1 \begin{pmatrix} c_1 \\ 0 \end{pmatrix} +_c p_2 \begin{pmatrix} c_2 \\ 0 \end{pmatrix} +_c p_3 \begin{pmatrix} c_3 \\ 0 \end{pmatrix} + \begin{pmatrix} O_c \\ 1 \end{pmatrix} \right] \quad (2.39)$$

2.3 Übersicht über die benötigten Koordinatensysteme und Transformationen für die Stereobildanalyse

Nachdem die mathematische Grundlage der Basis - beziehungsweise Koordinatentransformation erläutert wurde, muss diese nun für den Fall der Stereokalibrierung und 3D-Szenerekonstruktion entsprechend erweitert werden. Es müssen folgende Überlegungen gemacht werden. Zum einen muss geklärt werden, wie viele Transformationen nötig sind, um von einem 3D-Objekt in Weltkoordinaten auf ein projiziertes 2D-Bild dieses Objektes auf dem Sensor zu kommen. Zum anderen müssen die entsprechenden Basen dieser Koordinatensysteme definiert werden.

2.4 Aufbau der Koordinatensysteme

Insgesamt werden für die Stereoanalyse fünf verschiedene Koordinatensysteme Definiert. Das Weltkoordinatensystem, die Kamerakoordinatensysteme, Das Bildebenenkoordinatensystem und das Sensorkoordinatensystem. Grundlegend wird erst einmal festgelegt, welche Orientierungen die Koordinatensysteme haben sollen. Die Arbeit und auch die entstandenen Algorithmen basieren auf rechtsdrehenden Systemen. Die Möglichkeit linksdrehende Systeme zu benutzen sind aus dem entstandenen Algorithmus nicht ausgeschlossen, jedoch ist es für die spätere Deutung und Interpretation der Resultate wichtig im Vorhinein definiert zu haben, wie die Orientierung der einzelnen Koordinatensysteme ist. Das Weltkoordinatensystem wird mit $K_w = (\hat{e}_1, \hat{e}_2, \hat{e}_3, O)$ definiert, um die Notation des Beispiels im vorherigen Abschnitts einzuhalten und Verwirrung zu vermeiden. Des Weiteren wird festgehalten, dass das Weltkoordinatensystem gleich dem Koordinatensystem von einer der beiden Kameras entspricht. Die Bildebene ist gleich einer Ebene im 3D-Raum und wird mit E bezeichnet. Das Projektionszentrum auf der Bildebene bekommt die Notation Z zugeordnet. Die Kamerakoordinatensysteme sind wie das Weltkoordinatensystem kartesische rechtsdrehende Systeme. Für die Erklärung der Projektionen reicht es wenn wir zunächst den Weg von einem Objekt im 3D-Raum auf sein projiziertes Bild in einer der beiden Kameras genauer betrachten. Wie im Abschnitt zuvor wird dieses Kamerakoordinatensystem mit $K_c = (\hat{c}_1, \hat{c}_2, \hat{c}_3, O_c)$ definiert. Des Weiteren soll gelten, dass $O_c = Z$, sprich der Ursprung des Kamerakoordinatensystems Deckungsgleich mit dem Projektionszentrum auf der Bildebene ist und somit auch $\langle \hat{c}_1, \hat{c}_2 \rangle + P = E$. P bezeichnet hier den Hauptpunkt der Bildebene.. Für die Wahl der Kamerakoordinatenachsen wird folgendes Schema verfolgt: $\hat{c}_1 \cdot \hat{n} = 0$, $\hat{c}_2 \cdot \hat{n} = 0$, $\hat{c}_3 = \pm \hat{n}$. Die Bildebene selbst bekommt auch ein eigenes Koordinatensystem zugewiesen, welches sich aber nicht mehr auf einen 3D-Raum sondern auf die 2D-Ebene bezieht. Es soll $K_b = (\hat{b}_1, \hat{b}_2, O_b)$ mit $O_b = P$, $\hat{b}_1 = \hat{c}_1$,

$\hat{b}_2 = \hat{c}_2$ gelten. Formuliert man die Bildebene in Hess'scher Normalform so gilt $E : \hat{n} \cdot (\vec{x} - \vec{p}) = 0$. Als letztes kommt noch das Sensorkoordinatensystem mit $K_s = (\vec{u}, \vec{v}, O_s)$. Dieses Koordinatensystem ist an die Geometrie der Pixel und des Sensors angepasst und daher muss es sich nicht unbedingt um ein kartesisches Koordinatensystem handeln.

Nachdem die einzelnen Koordinatensysteme definiert sind, soll zunächst wieder symbolisch die Projektion eines Objekts aus dem 3D-Raum auf den 2D-Sensor durchgerechnet werden. Für die Transformation der Weltkoordinatenachsen in Kamerakoordinatenachsen gilt: $C\hat{e}_i = \hat{c}_i$, wobei C eine 3x3-Rotationsmatrix darstellt. Es kann also wie bereits bekannt eine Matrix M aufgestellt werden, welche das Weltkoordinatensystem in das Kamerakoordinatensystem überführt. Der Verschiebevektor \vec{v} besteht aus den Koordinaten des Projektionszentrums Z . Es gilt also $\vec{v} = Z \rightsquigarrow \vec{v} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}$.

$$(\hat{e}_1, \hat{e}_2, \hat{e}_3)[D] = (\hat{c}_1, \hat{c}_2, \hat{c}_3) \rightsquigarrow \hat{c}_1 = D_{11}\hat{e}_1 + D_{21}\hat{e}_2 + D_{31}\hat{e}_3 \quad (2.40)$$

$$\begin{bmatrix} D \end{bmatrix} = \left[\begin{pmatrix} \hat{c}_1 \\ \hat{c}_2 \\ \hat{c}_3 \end{pmatrix}_K \right] \quad (2.41)$$

$$(\hat{c}_1, \hat{c}_2, \hat{c}_3, O_c) = (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) \cdot \begin{bmatrix} [D] & z_1 \\ 0 & z_2 \\ 0 & z_3 \\ 0 & 1 \end{bmatrix} \quad (2.42)$$

des weiteren müssen die Transformierten Koordinaten noch mit einer entsprechenden Projektion auf die Kamera projiziert werden. Hierzu muss eine Projektionsmatrix der Form

$$K_{c_1} [\pi]_{K_{c_2}} = \begin{pmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & \zeta & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2.43)$$

aufgestellt werden. Zur Probe dient

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{pmatrix} \zeta X \\ \zeta Y \\ \zeta Z \\ Z \end{pmatrix} = \begin{pmatrix} \zeta \frac{X}{Z} \\ \zeta \frac{Y}{Z} \\ \zeta \\ 1 \end{pmatrix} \quad (2.44)$$

Im folgenden wird die Bedeutung hinter ζ hergeleitet. Einmal in Bezug der Projektion von Kamera- in Kamerakoordinaten. Danach in Bezug auf die Projektion von Kamera- in Bildebenekoordinaten. In der Literatur wird ζ meist mit f für *focal length* dargestellt und die Bildebene auch *Focal plane* genannt [2]. Hierzu wird später bei der Projektion auf die 2D Bildebene noch genaueres erläutert.

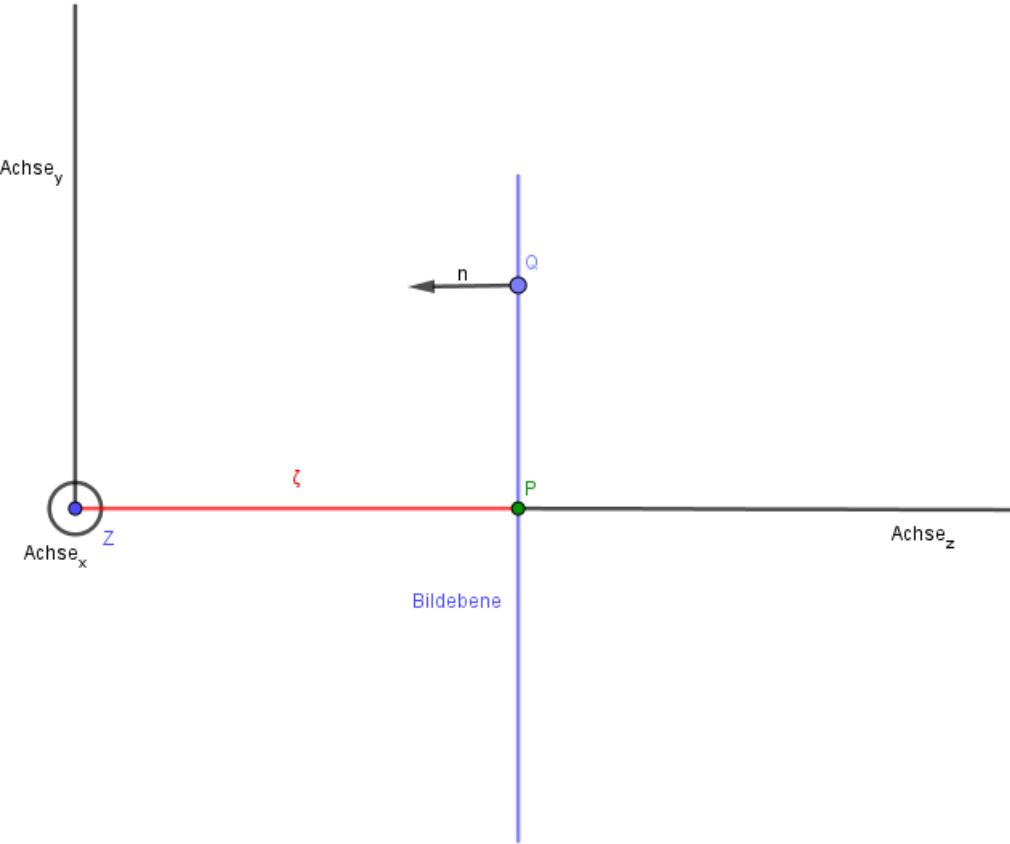


Abbildung 2.2: In blau ist die Bildebene dargestellt auf ihr befinden sich die Punkte Q und P . Z liegt nicht auf der Ebene, Das Projektionszentrum liegt hinter der Bildebene und somit auch hinter dem Sensor. n ist die Normale der Bildebene

Wie in Abbildung 2.2 ersichtlich kann für die Findung von ζ , welche den Abstand des Projektionszentrums zur Bildebene beschreibt, folgende Gleichung aufgestellt werden

$$\vec{p} + |\vec{QZ} \cdot \hat{n}| \vec{n} = \vec{Z} \quad (2.45)$$

Q ist ein beliebiger Punkt auf der Ebene. Setze

$$\vec{QZ} \cdot \hat{n} = \zeta \quad (2.46)$$

$$\vec{p} = \vec{Z}\zeta\hat{n} \quad (2.47)$$

Wählt man nun $\hat{c}_3 = \hat{n}$ so kann daraus geschlossen werden, dass $\vec{p} = \vec{Z} - \zeta\hat{n}$. Für die folgende Projektion der Koordinaten im 3D-Kamerakoordinatensystems auf das 2D-Bildebenenkoordinatensystem muss eine Projektionsmatrix der Form

$$K_b [\pi]_{K_c} = \begin{pmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2.48)$$

aufgestellt werden. Es gilt allgemein dass $O_B = Z - \zeta\hat{n} = \vec{p}$, $\hat{b}_1 = \hat{c}_1$, $\hat{b}_2 = \hat{c}_2$ ist. Zur Probe ob die Projektionsmatrix stimmt wird folgendes gerechnet.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{pmatrix} \zeta X \\ \zeta Y \\ Z \end{pmatrix} = \begin{bmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{pmatrix} \zeta \frac{X}{Z} \\ \zeta \frac{Y}{Z} \\ 1 \end{pmatrix} \quad (2.49)$$

Der Übergang von Kamerakoordinaten in Bildkoordinaten wird im folgenden nochmal bezogen auf das in dieser Arbeit definierte Modell beschrieben.

$$(\hat{b}_1, \hat{b}_2, O_B) = (\hat{c}_1, \hat{c}_2, \hat{c}_3, O_c) \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \zeta \\ 0 & 0 & 1 \end{bmatrix} \quad (2.50)$$

$$\text{Sei } (X)_B = \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix}$$

$$\rightsquigarrow (x)_c = \begin{pmatrix} x_1 \\ x_2 \\ \zeta \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & \zeta \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} \quad (2.51)$$

Für die Umkehrung muss hier mit der Pseudoinversen gearbeitet werden. Um die Projektionsmatrix welche die Bildebenenkoordinaten wieder in Kamerakoordinaten projiziert zu finden muss folgende Rechenoperation durchgeführt werden.

$$\begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ \zeta \\ 1 \end{pmatrix} \quad (2.52)$$

$$\rightsquigarrow {}_{K_c} [\pi]_{K_b} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & \zeta & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.53)$$

Um die Projektionsmatrix allgemeiner zu formulieren und die Bedeutung und hinter ζ und dessen Zusammenhang mit dem in der Literatur benutzten f genauer zu erläutern, wird die Projektionsmatrix der Photogrammetrie in Bezug auf ein Lochkameramodell[2] hier noch einmal genauer betrachtet und mit dem hergeleiteten Modell verglichen. Nehmen wir an es gilt $\zeta = f$, dann gilt für die Projektion von Punkten das selbe wie in Gleichung 2.49 nur mit f statt ζ .

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{pmatrix} \zeta X \\ \zeta Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{pmatrix} \zeta \frac{X}{Z} \\ \zeta \frac{Y}{Z} \\ 1 \end{pmatrix} \quad (2.54)$$

$$\rightsquigarrow \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{pmatrix} f \frac{X}{Z} \\ f \frac{Y}{Z} \\ 1 \end{pmatrix} \quad (2.55)$$

Zum Vergleich dient die Definition im Buch von *Hartley & Zisserman*[2], welche der selbst hergeleiteten entspricht.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (2.56)$$

Die komplette Projektionsmatrix K lautet in der Literatur wie folgt[2]

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.57)$$

Um zu verstehen was die Bedeutung hinter α_x und α_y ist, sollte zunächst noch geklärt werden, dass im hergeleiteten Ansatz dieser Arbeit davon ausgegangen wird, dass die *focal length* ζ beziehungsweise f in x - sowie in y -Richtung einheitlich ist. Dies ist jedoch nicht immer der Fall. Bei den bekannten und viel genutzten CCD-Kamerachips trifft es zu, dass die Bildkoordinaten und Sensorkoordinaten einheitliche Quadrate bilden. Jedoch gibt es ebenfalls Chips, bei denen es nicht-quadratische Pixel gibt[2]. Wird so ein Chip genutzt, so sind die Werte für f nicht mehr identisch, weshalb beide f meist einen Index besitzen und es steht dann jeweils f_x und f_y in der Matrix. Dies soll aufzeigen, dass es sich auch wenn die Werte identisch sein sollte, es sich trotzdem um zwei Verschiedene *focal length*- Einheiten handelt. Sind die Pixel nicht quadratisch, wird auf die Werte f_x und f_y jeweils eine Skalierung m_x und m_y drauf multipliziert so dass $\alpha_x = f_x \cdot m_x$ und $\alpha_y = f_y \cdot m_y$ entspricht.[2]. x_0 und y_0 bilden einen Verschiebungsvektor. Sie beinhalten die Definition wo sich der Hauptpunkt auf der Bildebene befindet. x_0 und y_0 sind definiert als $x_0 = p_x \cdot m_x$ und $y_0 = p_y \cdot m_y$. Die Variable s wird dem sogenannten *skew-Faktor* zugeordnet, welcher nur dann zum Einsatz kommt, sollte die optische Achse nicht orthogonal auf den Chip auftreffen. Sprich wenn der Chip geneigt in der Kamera montiert wurde[2]. Die komplette Kameramatrix $P = KM = K[C|t]$ beziehungsweise wie sie in der Literatur zitiert wird $P = K[R|t]$ [2] besteht aus der Matrixmultiplikation der hergeleiteten Transformationsmatrix M welche die externen Kameraparameter repräsentiert und der Projektionsmatrix K , welche die internen Kameraparameter repräsentiert. P beinhaltet also sowohl die internen als auch die externen Kameraparameter.

Zuletzt folgt nun noch die Transformation der Bildebenekoordinaten in die Sensorkoordinaten, welches mit $K_s = (\vec{u}, \vec{v}, O_s)$ definiert ist. \vec{u} und \vec{v} beinhalten wie bereits erwähnt die Information über die Geometrische Beschaffung der Pixel und muss sich dem entsprechend nicht zwangsläufig ein kartesisches Koordinatensystem sein. Das Koordinatensystem wird also folgendermaßen definiert

$$\vec{u} = u_1 \hat{b}_1 + u_2 \hat{b}_2 \quad (2.58)$$

$$\vec{v} = v_1 \hat{b}_1 + v_2 \hat{b}_2 \quad (2.59)$$

$$O_s = O_B + p_1 \hat{b}_1 + p_2 \hat{b}_2 \quad (2.60)$$

$$(\vec{u}, \vec{v}, O_s) = (\hat{b}_1, \hat{b}_2, O_B) \cdot \begin{bmatrix} u_1 & v_1 & p_1 \\ u_2 & v_2 & p_2 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.61)$$

Anhand eines Beispiels wird nun symbolisch durchgerechnet was das für die Bildebenekoordinaten bedeutet.

$$\text{Es sei } (X)_S = \begin{pmatrix} a \\ b \\ 1 \end{pmatrix}$$

$$\rightsquigarrow x = a\vec{u} + b\vec{v} + O_S \quad (2.62)$$

$$= a(u_1\hat{b}_1 + u_2\hat{b}_2) + b(v_1b_1 + v_2b_2) + O_B + p_1b_1 \quad (2.63)$$

$$\mapsto (X)_B = \begin{bmatrix} p_1 + av_1 + bv_1 \\ p_2 + au_2 + bu_2 \end{bmatrix} \quad (2.64)$$

$$(X)_S = \begin{bmatrix} \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix}^{-1} & -M^{-1} \begin{pmatrix} p_1 \\ p_2 \end{pmatrix} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ \\ \end{bmatrix}_B \quad (2.65)$$

$_{K_s} [\pi]_{K_c}$ wird also dementsprechend wie folgt dargestellt

$$_{K_s} [\pi]_{K_c} = \begin{bmatrix} M^{-1} & -M \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}^{-1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -\zeta & 0 & 0 & 0 \\ 0 & -\zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -\zeta M^{-1} & -M \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}^{-1} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad (2.66)$$

Zur Verdeutlichung folgen nun noch zwei Beispiele. Es werden \vec{u} und \vec{v} , sowie p_1 und p_2 mit Werten versehen. p entspricht symbolisch einem *Pixelpitch*-Wert. Mit *Pixelpitch* wird der direkte Abstand der Pixel auf Bildsensoren zwischen Pixelmitte zu Pixelmitte bezeichnet. Wir definieren also

$$\vec{u} = 1pb_1$$

$$\vec{v} = 2pb_2$$

$$p_1 = 15, p_2 = 20$$

Für die Projektionsmatrix ergibt sich dann

$$O_S = O_B - \vec{u} - \vec{v} \rightsquigarrow O_S = O_B - 15b_1 - 20b_2 \quad (2.67)$$

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \rightsquigarrow M^{-1} = \begin{bmatrix} \frac{1}{p} & 0 \\ 0 & \frac{1}{2p} \end{bmatrix} \quad (2.68)$$

$$[\pi] = \begin{bmatrix} \frac{-\zeta}{p} & 0 & 15 & 0 \\ 0 & \frac{-\zeta}{p} & 20 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.69)$$

Zum Nachvollziehen gibt es hier nochmal ein anderes Beispiel.

$$\vec{v} = 1pb_1 + 2pb_2$$

$$\vec{u} = 1pb_1$$

$$p_1 = 10, p_2 = 5$$

$$M = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix} \rightsquigarrow M^{-1} = \begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & \frac{1}{2} \end{bmatrix} \quad (2.70)$$

$${}_{K_s} [\pi]_{K_c} = \begin{bmatrix} \frac{-\zeta}{p} & \frac{-\zeta}{2p} & 10 & 0 \\ 0 & \frac{-\zeta}{p} & 5 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.71)$$

$$= [K|0] \quad (2.72)$$

Zusammengefasst wird hier noch einmal die symbolische Darstellung von ${}_{K_s} [\pi]_K$ aufgezeigt

$${}_{K_s} [\pi]_K = {}_{K_s} [\pi]_{K_c} \cdot \begin{bmatrix} [C]^{-1} & -[C]^{-1}Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.73)$$

Um sicherzugehen, dass die Herleitung der Projektionsmatrix von Bildebenenkoordinaten auf Sensorskoordinaten stimmen kann, gibt es hier noch einmal zum Vergleich die Darstellung aus *Hartley&Zisserman*[2]. Zu beachten ist hier, dass im *Hartley&Zisserman* R für $[C]^{-1}$ steht[2].

$$[K|0] \begin{bmatrix} R & -RZ \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.74)$$

$$[KR] - KRZ = KR[I_{3x3}] - Z \quad (2.75)$$

3 Homographien in der Ebene

Im vorherigen Kapitel wurde ausführlich dargelegt, wie Koordinatensysteme ineinander überführt werden. Die folgenden Unterkapitel sollen anhand eines Beispiels zeigen, dass die in (Link Kapitel) gezeigten Transformationen in Matrizen zusammenfassen lassen. Im ersten Beispiel wird davon ausgegangen, dass die intrinsischen und extrinsischen Kameraparameter unbekannt sind und nur die Bildpunkte von beiden Kamerakoordinatensystemen bekannt sind. Des Weiteren wird festgelegt, dass sich alle Bildpunkte auf einer Ebene befinden und somit die selbe Tiefe z besitzen. Die behauptung ist nun, dass es möglich ist eine 3×3 -Homographiematrix zu ermitteln, welche die Punkte von Kamera eins in die Punkte von Kamera zwei und umgekehrt überführen kann. Eine Homographie ist eine projektive Transformation zwischen zwei Ebenen. Dabei bleiben Kollinearitäten und die Reihenfolge von Punkten auf Geraden(z.B Schnittpunkte mit anderen Geraden) erhalten. Aufgrund der Ebenenannahme, kann solch eine projektive Transformation durch eine 3×3 -Homographiematrix ausgedrückt werden[3]. Sprich die entstehende projektive Transformation projiziert jede Figur in eine Figur gleicher projektiver Entsprechung[2]. Die Homographie ist eine allgemeine projektive Transformation, welche Abbildungen einer Ebene in eine andere beschreiben, dabei können Punkte in Punkte oder Gerade in Geraden überführt werden, wobei das Doppelverhältnis erhalten bleibt. Sind die Punkte A', B', C' und D' die projektiven Bilder eines Systems von vier kollinearen Punkten, so ist $(A', B', C', D') = (A, B, C, D)$ [4]. Schon Euler hatte Bewegungen untersucht, er hatte im Prinzip gezeigt, dass eine ebene Kongruenzabbildung eine Rotation, eine Translation oder eine Translation gefolgt von einer Spiegelung ist. Möbius nahm den Eulerschen Terminus affine Transformation wieder auf, um solche Transformationen zu benennen, die die Parallelität erhalten, aber nicht abstandstreu sind. Die allgemeinste Transformation, die Möbius studierte, war die Homographie, die er Kollineation nannte[4]

Es seien $x = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ die homogenen Koordinaten eines Punktes der projektiven Ebene und $x' = \begin{pmatrix} x'_1 \\ x'_2 \\ x'_3 \end{pmatrix}$ die Punkte des projektiv transformierten Punktes. Dann gilt

$$x' = Hx \quad (3.1)$$

$$Hx = \begin{bmatrix} h_1^T \cdot x \\ h_2^T \cdot x \\ h_3^T \cdot x \end{bmatrix} \quad (3.2)$$

$$\rightsquigarrow x' = Hx = \begin{bmatrix} h_{11}x_1 + h_{12}x_2 + h_{13}x_3 \\ h_{21}x_1 + h_{22}x_2 + h_{23}x_3 \\ h_{31}x_1 + h_{32}x_2 + h_{33}x_3 \end{bmatrix} \quad (3.3)$$

$$\rightsquigarrow H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (3.4)$$

Dabei müssen die Koeffizienten so geartet sein, dass die zugehörige Transformation umkehrbar ist. [2][4]. Sprich es muss gelten dass wenn

$$x' = Hx \quad (3.5)$$

$$x = H^{-1}x' \quad (3.6)$$

Danach wird dann die Beziehung von Punkten im Raum aufgezeigt, welche sich nicht auf einer Ebene befinden. Die Relationen dieser Punkte zueinander lassen sich mit der so genannten Epipolaregeometrie beschreiben. Mehr dazu ab Kapitel (link zu Kapitel 3.3).

3.1 Homographie zwischen den Abbildungen eines Quaders einer Ausgangskamera und einer dazu um das Projektionszentrum rotierten Kamera

In diesem Beispiel wird die Abbildung eines Quadrats einer Kamera in die einer anderen Kamera transformiert. Danach wird eine Homographiematrix ermittelt und auf ihre Gültigkeit hin untersucht. Sprich es wird geschaut, ob sich die Punkte der einen Kamera in die Punkte der anderen nur mit Hilfe dieser Matrix überführen lassen und anders herum. Erwähnt sei außerdem, dass in diesem Beispiel nicht von überbestimmten Systemen ausgegangen wird. Für diese gilt ein leicht anderer Algorithmus, welcher auch in den folgenden Algorithmen dieser Arbeit, wie zum Beispiel bei der Findung der Fundamentalmatrix gebraucht wird und dort genauer besprochen wird. Die Kamerakoordinatensysteme unterscheiden sich durch eine Drehung um 180° um die \hat{e}_3 -Achse. Der Ursprung beider Kamerakoordinatensysteme entspricht dem jeweiligen Projektionszentrum. Es werden zwei Bilder der selben Szene mit diesen Kameras aufgenommen. Die Behauptung ist, dass sich die beiden entstandenen Bilder mit Einer Homographie ineinander überführen lassen.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (3.7)$$

Die zwei Kamerakoordinatensysteme werden in Abbildung 3.1 nochmal grafisch veranschaulich.

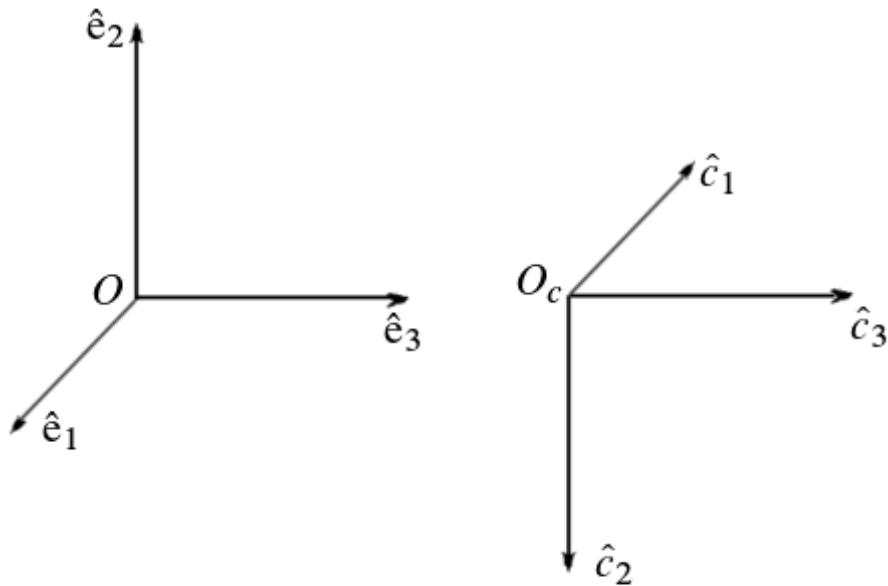


Abbildung 3.1: Weltkoordinatensystem $K = (O, \hat{e}_1, \hat{e}_2, \hat{e}_3)$ und Kamerakoordinatensystem $K_c = (O_c, \hat{c}_1, \hat{c}_2, \hat{c}_3)$.

Zur Übersichtlichkeit wurden die Koordinatensysteme in Abbildung 3.1 verschoben voneinander dargestellt. In Wirklichkeit sind die Ursprünge O und O_c deckungsgleich. Als nächstes werden die intrinsischen Kameraparameter festgelegt, hierbei gilt:

$${}_{K_c} [\pi]_{K_c} = \begin{pmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & \zeta & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.8)$$

$$\rightsquigarrow \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = {}_{K_c} [\pi]_{K_c} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \zeta X \\ \zeta Y \\ \zeta Z \\ Z \end{pmatrix} = \begin{pmatrix} \frac{\zeta}{Z} X \\ \frac{\zeta}{Z} Y \\ \zeta \\ 1 \end{pmatrix} \quad (3.9)$$

Für das Beispiel gilt zunächst die Bedingung $\zeta \neq 0$. Des Weiteren soll gelten, dass \hat{c}_3 gleich der Lotgeraden vom Sender zum Projektionszentrum entspricht und somit folgt, dass \hat{c}_1 in Sensorebene liegt und $\hat{c}_2 = \hat{c}_3 \times \hat{c}_1$ ist. Der Quader besteht aus den Punkten A, B, C, D und E in homogenen Weltkoordinaten. Das Koordinatensystem von Kamera eins ist gleich dem um 180° gedrehte Weltkoordinatensystem. Die Punkte in Weltkoordinaten lauten wie folgt.

$$(A)_K = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix}, (B)_K = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}, (C)_K = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \end{pmatrix}, (D)_K = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix}, (E)_K = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 2 \\ 1 \end{pmatrix} \quad (3.10)$$

Kamera zwei wird für die Aufnahme der Szene zusätzlich noch um 45° zu Kamera eins eingedreht. Für die Transformation der Weltkoordinatentupel in Kamera eins und Kamera zwei müssen also zuerst die Punkte in den jeweiligen Kamerakoordinatensystemen angegeben werden. Für die Transformation der Weltkoordinaten in die entsprechenden Kamerakoordinaten werden also zwei Matrizen T_1 und T_2 benötigt. T_1 bewirkt die Drehung der Punkte um 180° um die \hat{e}_3 -Achse. Um T_2 zu erhalten wird T_1 zusätzlich noch mit einer weiteren Transformationsmatrix, welche die Drehung um 45° beinhaltet, verrechnet. Die so erhaltenen Matrizen T_1 und T_2 können nun dazu verwendet werden, die Punkte Bezuglich des Weltkoordinatensystems in Punkte bezüglich der jeweiligen Kamerakoordinatensysteme zu transformieren. Im ersten Schritt erfolgt die Drehung um 180° um die \hat{e}_3 -Achse.

$$\begin{bmatrix} \cos(180) & -\sin(180) & 0 & 0 \\ \sin(180) & \cos(180) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) \quad (3.11)$$

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = T_1 \quad (3.12)$$

Im nächsten Schritt wird die Kamera um 45° zu Kamera eins eingedreht, es ergibt sich die folgende Matrix T_2 :

$$\cos(45) = \sin(45) = \frac{1}{\sqrt{2}} = a \quad (3.13)$$

$$\begin{bmatrix} a & 0 & a & 0 \\ 0 & 1 & 0 & 0 \\ -a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (\hat{e}_1, \hat{e}_2, \hat{e}_3, O) \quad (3.14)$$

$$= \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = T_2 \quad (3.15)$$

Sind die Matrizen T_1 und T_2 ermittelt, können nun die Weltkoordinatentupel in die Koordinatentupel von Kamera eins und Kamera zwei transformiert werden. Zunächst die Transformation der Punkte in die um 180° gedrehte Kamera eins.

$$\begin{pmatrix} (A)_{K_{c1}} \\ 1 \end{pmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix} \quad (3.16)$$

$$\begin{pmatrix} (B)_{K_{c1}} \\ 1 \end{pmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 2 \\ 1 \end{pmatrix} \quad (3.17)$$

$$\begin{pmatrix} (C)_{K_{c1}} \\ 1 \end{pmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ -1 \\ 2 \\ 1 \end{pmatrix} \quad (3.18)$$

$$\begin{pmatrix} (D)_{K_{c1}} \\ 1 \end{pmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 2 \\ 1 \end{pmatrix} \quad (3.19)$$

Danach die Transformation der Weltkoordinaten in die um 180° um die \hat{e}_3 -Achse gedrehte und zusätzlich um die \hat{e}_2 -Achse um 45° eingedrehte Kamera zwei mit Matrix T_2 .

$$\begin{pmatrix} (A)_{K_{c2}} \\ 1 \end{pmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2a \\ 0 \\ 2a \\ 1 \end{pmatrix} \quad (3.20)$$

$$\begin{pmatrix} (B)_{K_{c2}} \\ 1 \end{pmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ 0 \\ 3a \\ 1 \end{pmatrix} \quad (3.21)$$

$$\begin{pmatrix} (C)_{K_{c2}} \\ 1 \end{pmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ -1 \\ 3a \\ 1 \end{pmatrix} \quad (3.22)$$

$$\begin{pmatrix} (D)_{K_{c2}} \\ 1 \end{pmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2a \\ -1 \\ 2a \\ 1 \end{pmatrix} \quad (3.23)$$

Danach müssen die entstandenen Koordinatentupel von Kamera eins und Kamera zwei noch mit ihren jeweiligen Abbildungsmatrizen verrechnet werden. ζ bekommt in diesem Beispiel den Wert -1, das bedeutet laut der Definition der Kamerakoordinatensysteme, dass der Sensor sich hinter dem Projektionszentrum befindet. Das entstehende Bild ist somit um 180° gespiegelt auf dem Sensor abgebildet.

$${}_{K_{c1}} [\pi]_{K_{c1}} = {}_{K_{c2}} [\pi]_{K_{c2}} = \begin{pmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & \zeta & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.24)$$

Die vier Punkte werden jeweils in einer Matrix zusammengeschrieben, so folgt für die Koordinatentupel von Kamera eins folgendes:

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & -1 & -1 & 0 \\ 0 & 0 & -1 & -1 \\ 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 1 \\ -2 & -2 & -2 & -2 \\ 2 & 2 & 2 & 2 \end{pmatrix} \quad (3.25)$$

$$\simeq \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.26)$$

Und für die Koordinatentupel von Kamera zwei gilt:

$$\begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 2a & a & a & 2a \\ 0 & 0 & -1 & -1 \\ 2a & 3a & 3a & 2a \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -2a & -a & -a & -2a \\ 0 & 0 & 1 & 1 \\ -2a & -3a & -3a & -2a \\ 2a & 3a & 3a & 2a \end{pmatrix} \quad (3.27)$$

$$\approx \begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} & -1 \\ 0 & 0 & \frac{1}{3a} & \frac{1}{2a} \\ -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.28)$$

Die entstandenen Punkte beider Kameras wurden in Geogebra eingegeben und das Programm liefert die folgende beide Quadrate

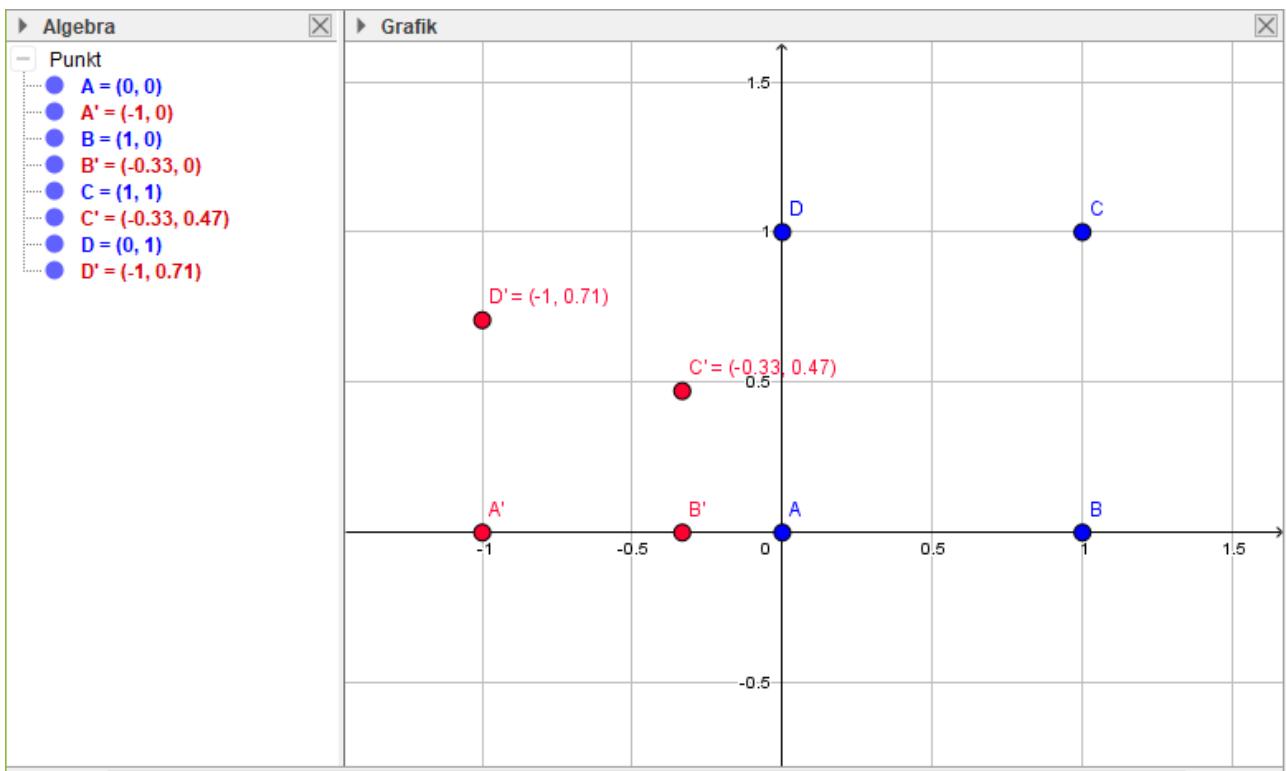


Abbildung 3.2: in blau ist die Abbildung des Quaders von Kamera eins und in rot die Abbildung des selben Quaders in Kamera zwei (BILD NOCHMAL ÜBERARBEITEN)

Die nun ermittelten Punkte sollen durch eine eigens aufgestellte Homographiematrix ineinander überführt werden. Um eine Homographiematrix mit $H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$ zu ermitteln werden die Punkte bei der Kameras in eine Koeffizientenmatrix welche sich nach folgendem Schema ergibt: (Tafelaufschrieb einbauen)

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} P_{K1} \end{bmatrix} = \begin{bmatrix} P_{K2} \end{bmatrix} \quad (3.29)$$

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad (3.30)$$

(3.31)

Hieraus ergeben sich die folgenden drei Gleichungssysteme

$$h_{11}x + h_{12}y + h_{13}z = \lambda x' \quad (3.32)$$

$$h_{21}x + h_{22}y + h_{23}z = \lambda y' \quad (3.33)$$

$$h_{31}x + h_{32}y + h_{33}z = \lambda z' \quad (3.34)$$

Da mit homogenen Koordinaten gearbeitet wird und somit z und $z' = 1$ sind, ergibt sich für die letzte Zeile $h_{31}x + h_{32}y + h_{33}z = 1$. Dieser Ausdruck kann dann in den ersten beiden Gleichungen für λ eingesetzt werden. Es ergeben sich pro Punktpaar jeweils zwei Gleichungen.

$$h_{11}x + h_{12}y + h_{13}z = (h_{31}x + h_{32}y + h_{33}z) \cdot x' \quad (3.35)$$

$$h_{21}x + h_{22}y + h_{23}z = (h_{31}x + h_{32}y + h_{33}z) \cdot y' \quad (3.36)$$

Für den Aufbau der Koeffizientenmatrix werden beide Ausdrücke noch nach Null aufgelöst

$$h_{11}x + h_{12}y + h_{13}z - (h_{31}x + h_{32}y + h_{33}z) \cdot x' = 0 \quad (3.37)$$

$$h_{21}x + h_{22}y + h_{23}z - (h_{31}x + h_{32}y + h_{33}z) \cdot y' = 0 \quad (3.38)$$

$$\rightsquigarrow h_{11}x + h_{12}y + h_{13}z - h_{31}x \cdot x' - h_{32}y \cdot x' - h_{33}z \cdot x' = 0 \quad (3.39)$$

$$\rightsquigarrow h_{21}x + h_{22}y + h_{23}z - h_{31}x \cdot y' - h_{32}y \cdot y' - h_{33}z \cdot y' = 0 \quad (3.40)$$

Für die Koeffizientenmatrix ergibt sich dann folgendes.

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 & 1 \cdot x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & x_1y'_1 & y_1y'_1 & 1 \cdot y'_1 \\ & & & \ddots & & & & & \\ & & & & \ddots & & & & \\ x_i & y_i & 1 & 0 & 0 & 0 & x_ix'_i & y_ix'_i & 1 \cdot x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & x_iy'_i & y_iy'_i & 1 \cdot y'_i \end{pmatrix} \cdot \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_i \end{pmatrix} = 0 \quad (3.41)$$

Wenn ein Idealfall, sprich wenn der Rang der Koeffizientenmatrix genau acht beträgt, vorliegt, so kann aus der Koeffizientenmatrix einfach der Nullraum berechnet werden. Dieser Nullraum entspricht dem Kern der Koeffizientenmatrix. Das Ergebnis ist ein Spaltenvektor mit 9 Einträgen, welche in die 3×3 -Homographiematrix übertragen werden können. [2][5]. Tritt nun der Fall ein, dass man ein überbestimmtes System besitzt, was auftritt wenn mehr als 9 Punktpaare durch eine Homographie ineinander überführt werden sollen, so kann nicht mehr der Nullraum für die Berechnung der Homographiematrix benutzt werden. Das resultierende Ergebnis würde zwei oder mehr voneinander unabhängige Lösungen bieten, aus denen man die reale Lösung erst noch herausfinden muss. Mehr zu diesem Verfahren wird bei der Berechnung der Fundamentalmatrix nochmal genauer erläutert.[2]

Für die Lösung überbestimmter Gleichungssysteme bietet sich die Singulärwertszerlegung an[2][6]. Das bedeutet es wird nicht derjenige Vektor x gesucht für den gilt $H \cdot x = 0$, sondern es wird derjenige Vektor x gesucht, für den $\| H \cdot x \|$ minimal wird.[2]

Definition der Singulärwertszerlegung: Eine Faktorisierung einer beliebigen Matrix $A \in \mathbb{R}^{m \times n}$ der Form $A = U \cdot S \cdot V^T$ mit orthogonalen Matrizen $U \in \mathbb{R}^{m \times m}$ und $V \in \mathbb{R}^{n \times n}$ sowie mit einer Diagonalmatrix

$$S = \begin{pmatrix} s_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \ddots \\ \vdots & \ddots & \ddots & \ddots & & \ddots \\ \vdots & \ddots & \ddots & \ddots & & \ddots \\ 0 & \dots & s_r & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & & \ddots \\ \vdots & & \ddots & \ddots & & \ddots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{pmatrix} \quad (3.42)$$

heißt Singulärwertszerlegung von A . Dabei gelte $s_1 \geq s_2 \geq \dots \geq s_r \geq 0$, und die Zahlen s_1 bis s_r werden als Singulärwerte von A bezeichnet.[6].

(OBEN ALLES NOCH UMFORMULIEREN)

Entsprechend dieser Methode wird eine Singulärwertszerlegung, kurz *SVD* der entstandenen Koeffizientenmatrix, welche gleich der Form von Matrix A und im fortlaufenden auch mit A bezeichnet wird, durchgeführt. Wir erhalten 3 Matrizen $U \cdot S \cdot V^T$. Durch die Zerlegung sind die diagonaleinträge von S in einer absteigenden Reihenfolge sortiert. Der kleinste Sigulärwert korrespondiert auf diese Weise mit der letzten Spalte von V . Somit gleichen die 9 Einträge der Homographiematrix gleich der letzten Spalte von V . Das Ergebnis für H hat dann die folgende Form

$$H = \begin{pmatrix} v_{19} & v_{29} & v_{39} \\ v_{49} & v_{59} & v_{69} \\ v_{79} & v_{89} & v_{99} \end{pmatrix} \quad (3.43)$$

Für das Minimalbeispiel mit den eigens erstellten reinen Punkten, welches in diesem Kapitel erstellt wurde, würde die Herleitung der Homographiematrix über die Ermittlung des Nullraumes der Koeffizientenmatrix genügen. Für die Matrix H ergibt sich aus den Werten der im Beispiel verwendeten Punkte

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & 1.41421 & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad (3.44)$$

Nun werden die Punkte aus Kamera eins und die Punkte aus Kamera zwei jeweils mit Hilfe von H ineinander überführt. Hierzu werden die Tiefenwerte z der Punkte vernachlässigt. Die Punkte liegen alle auf einer Ebene, die Tiefe ist somit für die Umrechnung der Punkte nicht relevant.

$$\begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} & -1 \\ 0 & 0 & \frac{1}{3a} & \frac{1}{2a} \\ -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} & -1 \\ 0 & 0 & \frac{1}{3a} & \frac{1}{2a} \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.45)$$

$$\begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.46)$$

Probe:

$$x' = H \cdot x \rightsquigarrow \begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} & -1 \\ 0 & 0 & \frac{1}{3a} & \frac{1}{2a} \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & 1.41421 & 0 \\ 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.47)$$

und umgekehrt:

$$x = H^{-1} \cdot x' \rightsquigarrow \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 0.5 & 0 & 0.5 \\ 0 & 0.707107 & 0 \\ 0.5 & 0 & 0.5 \end{pmatrix} \cdot \begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} & -1 \\ 0 & 0 & \frac{1}{3a} & \frac{1}{2a} \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.48)$$

HIER WEITER SCHREIBEN!!!! (WICHTIG NOCH VERBESSERN, nicht die Transformierte von H ergibt jeweils die umkehrfunktion sondern die INVERSE!!!!)

3.2 Abbildungsunterschiede von Rotationen um ein Projektionszentrum und Rotation um einen beliebigen Drehpunkt von Punkten in der Ebene

Im folgenden soll kurz eingeschoben werden, dass sie Abbildungen von Punkten in einer Ebene unterscheiden, sobald sich ihr Drehpunkt ändert. Zu diesem Zweck wurde eine kleine Simulation geschrieben, welche die Abbildung eines Objekts zeigt, wenn sich die Kamera um ihr Kamerazentrum dreht und im Vergleich hierzu wenn sie sich um einen definierten Drehpunkt dreht. In diesem Beispiel ist der rote Punkt der Drehpunkt. Die Frage die sich hinter dieser Simulation befand war, ob es bei beiden Drehungen zum selben oder unterschiedlichen Bildern kommt. Des Weiteren sollte geklärt werden, ob Punkte die bei der Drehung um das Projektionszentrum verdeckt bleiben auch bei einer Drehung um einen außerhalb der Kamera platzierten Drehpunkt ebenfalls verdeckt bleiben und ob eine gültige Homographiematrix errechnet werden kann. Abbildung 3.3 zeigt ein Quadrat mit vier Eckpunkten und seinem Mittelpunkt in rot.

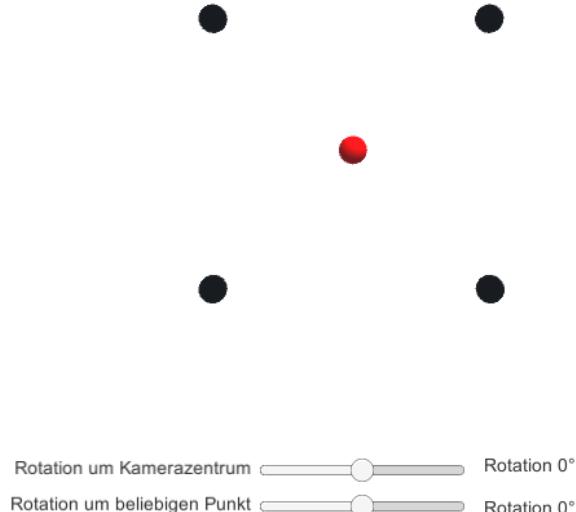


Abbildung 3.3: Objekt im Raum

Für die Simulation der Drehung wurden zwei Schieberegler implementiert mit dem sich die Kamera einmal um ihre eigene y-Achse in diesem Fall das Projektionszentrum dreht und einmal um den Drehpunkt, welche wie bereits gesagt der rote Mittelpunkt ist.

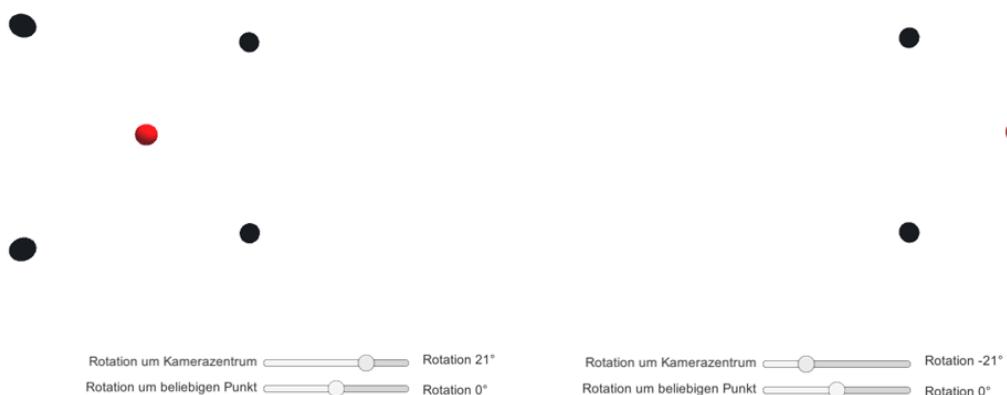


Abbildung 3.4: Drehung um das Projektionszentrum

Abbildung 4.3 zeigt jeweils die entstehende Abbildung wenn die Kamera um 20° beziehungsweise -20° um das Projektionszentrum gedreht wurde.



Abbildung 3.5: Drehung um einen Drehpunkt. In diesem Beispiel wurde der rote Punkt als Drehpunkt verwendet

Abbildung 3.4 zeigt die entstehenden Bilder, wenn die Kamera um 45° beziehungsweise -45° um den Drehpunkt gedreht wurde. Wie sich zeigt ist hier der Punkt welcher hinter dem roten Punkt platziert wurde sichtbar. Die Behauptung die nun noch zu beweisen gilt ist, dass die jeweiligen Abbildungen beide Drehungen durch jeweils eine Homographiematrix H dargestellt werden können, sofern sich die Punkte auf einer Ebene befinden. Ausgegangen wird immer vom Ursprungsbild in Abbildung 3.3. Das die Drehung um das Projektionszentrum eine gültige Homographiematrix hervorbringt wurde im vorherigen Unterkapitel (Link Kapitel 3.1) bereits gezeigt. Die Herangehensweise für das Beispiel um einen Drehpunkt ist die größtenteils die selbe. Das einzige was sich unterscheidet ist die Transformation der Punkte in ein Koordinatensystem von Kamera zwei. Diese besteht hier nicht nur aus einer Rotation, sondern zu allererst wird die Kamera zwei zum Drehpunkt verschoben, dann gedreht und zum Schluss wieder um den selben Translatationvektor zurückverschoben. Die Transformationsmatrix M hat besteht dann aus drei Transformationsmatrizen $M = V_1 \cdot R \cdot V_2$.

Beispiel aufzeigen (Code)

Beweis Homographie (beinhaltet sowohl Translation als auch Rotation) –
Drehung um ein Drehpunkt ist nichts weiter als eine Translation + Rotation und Rücktranslation. –
Punkte bleiben auf einer Ebene – alle Voraussetzungen für Homographien sind erfüllt

Sollte wie im zweiten Beispiel der grüne Punkte mit durch eine Homographie übermittelt werden, so ist die entstehende Homographie ungültig. Kamera zwei ist in diesem Beispiel um ihr Projektionszentrum gedreht.

– erklären warum homographien nur in der Ebene funktionieren

3.3 Punkte in unterschiedlichen Ebenen

Überleitung zur Epipolargeometry.

Warum kann hier keine Homographie benötigt werden

was muss hier genutzt werden?

3.4 Epipolargeometrie als Grundlage der Stereokalibrierung und Szenenrekonstruktion

(MEHR LITERATUR FINDEN ZUM VERGLEICHEN)

3.5 Geometrische Erläuterung der Fundamentalmatrix und der Essentiellen Matrix

4 Minimalbeispiel 3D-Stereokalibrierung und Szenenrekonstruktion bei Kameras gleicher Auflösung

4.1 Vorgehen: Projektion eines Quaders in zwei verschiedenen transformierte Kameras

Um den Mathematischen Vorgang der stereoskopischen Szenenrekonstruktion zu verdeutlichen, wurde ein Minimalszenario erstellt. Für dieses Minimalszenario wurde ein Objekt in diesem Falle ein Quader in ein zuvor Definiertes Weltkoordinatensystem gesetzt. Des Weiteren wurden zwei Kameras in die Szene platziert. Kamera 1 ist von der Lage Deckungsgleich mit dem Weltkoordinatensystem. Kamera 2 wurde verschoben und Rotiert, um so ein anderes Abbild des Objekts auf dem Sensor so produzieren. Für unsere Berechnungen der Szene sind äußere und innere Kameraparameter von uns festgelegt worden. Für eine Szenenrekonstruktion mit realen Bedingungen müssen diese zunächst ermittelt werden, um die aufgezeigten mathematischen Vorgänge anwenden zu können.

Für die Stereokamerakalibrierung wird eine der beiden Kameras relativ zur ersten Kamera um einen Vector \vec{v} verschoben und anschließend um einen Winkel α um die e_2 Achse (vgl. mit Dokumentation Koordinatensysteme) gedreht. Für die Rotation um e_2 gilt folgende Drehmatrix:

$$R = \begin{pmatrix} \cos(\alpha) & 0 & \sin(\alpha) \\ 0 & 1 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \quad (4.1)$$

Um die Transformationsmatrix M welche die Rotation und die Translation beinhaltet zu bekommen werden R und \vec{v} miteinander verrechnet, hierzu wird aus \vec{v} eine Matrix V gebildet.

$$V = \begin{pmatrix} 1 & 0 & 0 & -v_1 \\ 0 & 1 & 0 & -v_2 \\ 0 & 0 & 1 & -v_3 \end{pmatrix} \quad (4.2)$$

$$M = R^T \cdot V \quad (4.3)$$

$$M = \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -v_1 \\ 0 & 1 & 0 & -v_2 \\ 0 & 0 & 1 & -v_3 \end{pmatrix} \quad (4.4)$$

$$M = \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) & -v_1 \cos(\alpha) + v_3 \sin(\alpha) \\ 0 & 1 & 0 & -v_2 \\ \sin(\alpha) & 0 & \cos(\alpha) & -v_1 \sin(\alpha) - v_3 \cos(\alpha) \end{pmatrix} \quad (4.5)$$

Die entstandene Matrix M beschreibt die Transformation der der Kamera 2 und somit auch die Transformation des Koordinatensystems der Kamera 1 in das Koordinatensystem der Kamera 2.

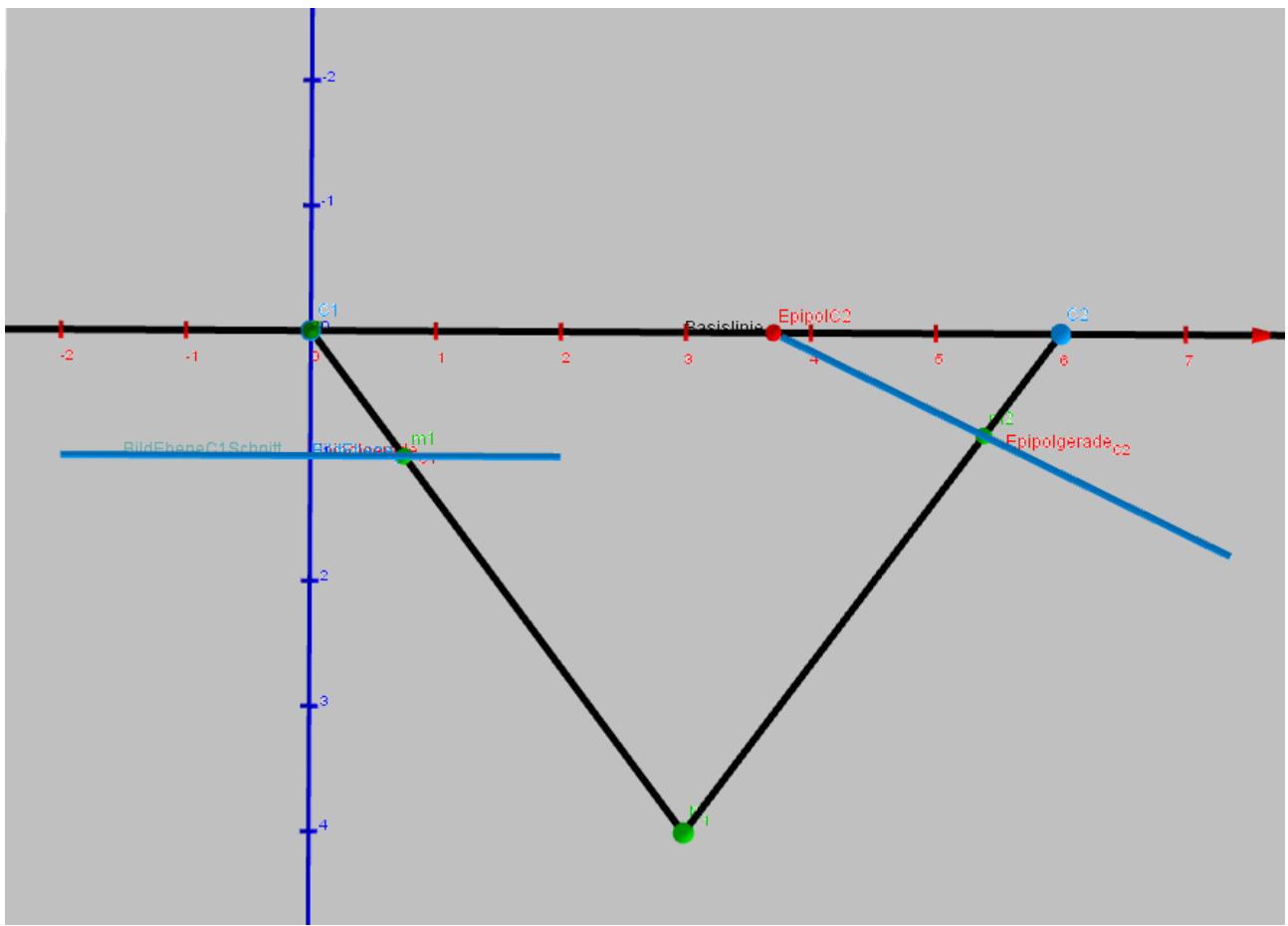


Abbildung 4.1: Top-Down-Ansicht des entstandenen Aufbaus der Kameras

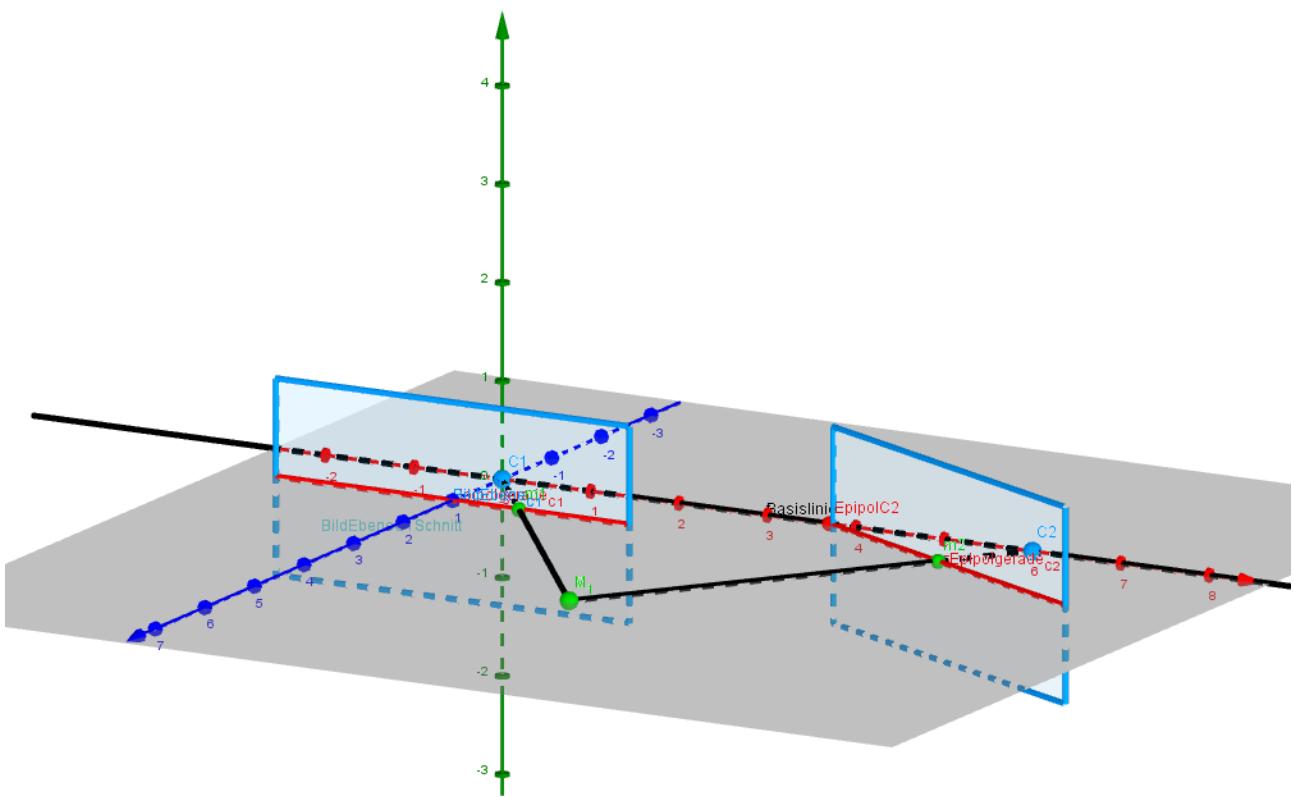


Abbildung 4.2: 3D-Ansicht des entstandenen Aufbaus der Kameras

4.2 Berechnung der Projektionsmatrizen

Die Punkte $a, b, c, d, a', b', c', d', d2$ in Weltkoordinaten sind bekannt. Die ersten acht Punkte bilden gemeinsam einen Quader, der neunte Punkt ist aus diesem Quader ausgelagert. Um die Koordinaten der Punkte in Kamera 1 und Kamera 2 Koordinaten zu Transformieren, muss zusätzlich zur Transformationsmatrix für Kamera2 noch die Abbildungsmatrix $AB1$ und $AB2$ der jeweiligen Kamera berücksichtigt werden.

$$AB1 = \begin{bmatrix} \zeta_{K1} & 0 & 0 & 0 \\ 0 & \zeta_{K1} & 0 & 0 \\ 0 & 0 & \zeta_{K1} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.6)$$

$$AB2 = \begin{bmatrix} \zeta_{K2} & 0 & 0 & 0 \\ 0 & \zeta_{K2} & 0 & 0 \\ 0 & 0 & \zeta_{K2} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.7)$$

Um die Projektionsmatrizen $PM1$ und $PM2$ zu berechnen, bekommt die Matrix M noch eine Homogene Erweiterung und sieht dann folgendermaßen aus:

$$M = \begin{bmatrix} \cos(\alpha) & 0 & -\sin(\alpha) & -v_1 \cos(\alpha) + v_3 \sin(\alpha) \\ 0 & 1 & 0 & -v_2 \\ \sin(\alpha) & 0 & \cos(\alpha) & -v_1 \sin(\alpha) - v_3 \cos(\alpha) \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.8)$$

$$PM1 = AB1 \cdot I_{4x4} \quad (4.9)$$

$$PM1 = \begin{bmatrix} \zeta_{K1} & 0 & 0 & 0 \\ 0 & \zeta_{K1} & 0 & 0 \\ 0 & 0 & \zeta_{K1} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.10)$$

I ist die Einheitsmatrix, wir gehen davon aus das Kamera 1 weder eine Drehung noch eine Translation beinhaltet und ihr Ursprung sich im Weltkoordinatenursprung befindet.

$$PM2 = AB2 \cdot M \quad (4.11)$$

$$PM2 = \begin{bmatrix} \zeta_{K2} \cos(\alpha) & 0 & \zeta_{K2} \sin(\alpha) & -\zeta_{K2}(v_1 \cos(\alpha) + v_3 \sin(\alpha)) & 0 \\ 0 & 1 & 0 & \zeta_{K2} - v_2 & 0 \\ \zeta_{K2} \sin(\alpha) & 0 & \zeta_{K2} \cos(\alpha) & -\zeta_{K2}(v_1 \sin(\alpha) + v_3 \cos(\alpha)) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.12)$$

4.3 Transformation der Weltpunkte in Koordinaten der Koordinatensysteme von beiden Kameras

Die 3D-Punkte $a, b, c, d, a', b', c', d', d2$, werden um eine Homogene Komponenten erweitert und mit den Projektionsmatrizen $PM1$ und $PM2$ verrechnet. Die so neu entstehenden Punktematrix $PK1$ mit $K1a, K1b, K1c, K1d, K1a', K1b', K1c', K1d', K1d2$ und $PK2$ mit $K2a, K2b, K2c, K2d, K2a', K2b', K2c', K2d', K2d2$ beschreiben jeweils den selben Quader aus den beiden verschiedenen Kamerakoordinatensystemen.

$$PK1 = \begin{bmatrix} \begin{pmatrix} a \\ 1 \end{pmatrix} & \begin{pmatrix} b \\ 1 \end{pmatrix} & \begin{pmatrix} c \\ 1 \end{pmatrix} & \begin{pmatrix} d \\ 1 \end{pmatrix} & \begin{pmatrix} a' \\ 1 \end{pmatrix} & \begin{pmatrix} b' \\ 1 \end{pmatrix} & \begin{pmatrix} c' \\ 1 \end{pmatrix} & \begin{pmatrix} d' \\ 1 \end{pmatrix} & \begin{pmatrix} d2 \\ 1 \end{pmatrix} \end{bmatrix} \cdot PM1 \quad (4.13)$$

$$PK2 = \begin{bmatrix} \begin{pmatrix} a \\ 1 \end{pmatrix} & \begin{pmatrix} b \\ 1 \end{pmatrix} & \begin{pmatrix} c \\ 1 \end{pmatrix} & \begin{pmatrix} d \\ 1 \end{pmatrix} & \begin{pmatrix} a' \\ 1 \end{pmatrix} & \begin{pmatrix} b' \\ 1 \end{pmatrix} & \begin{pmatrix} c' \\ 1 \end{pmatrix} & \begin{pmatrix} d' \\ 1 \end{pmatrix} & \begin{pmatrix} d2 \\ 1 \end{pmatrix} \end{bmatrix} \cdot PM2 \quad (4.14)$$

Die entstandenen Koordinaten sehen dann entsprechen Gleichung 16 aus und müssen dann wieder auf eine homogene Form gebracht werden.

$$aK2 = \begin{pmatrix} x \\ y \\ \delta z \\ \delta \end{pmatrix} \rightsquigarrow aK2 = \begin{pmatrix} \frac{x}{\delta} \\ \frac{y}{\delta} \\ \frac{\delta z}{\delta} \\ \frac{\delta}{\delta} \end{pmatrix} \rightsquigarrow aK2 = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ z \\ 1 \end{pmatrix} \quad (4.15)$$

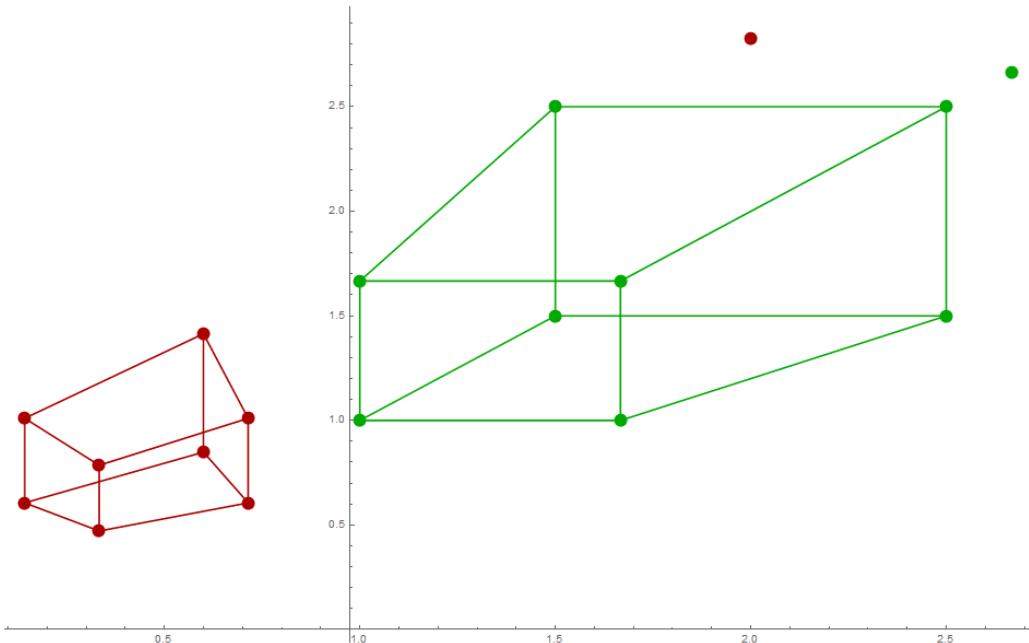


Abbildung 4.3: Grün zeigt den Quader aus sicht von Kamera 1. Das größere Quadrat sind die vorderen Punkte a, b, c, d , das kleinere Quadrat sind die hinteren Punkte a', b', c', d' . Rot zeigt denselben Quader aus Sicht vom Kamera 2, wenn diese in $+e_1$ -Richtung verschoben und um 45° um e_2 gedreht wurde

4.4 Berechnung der projizierten Punkte auf den beiden Bildebenen

Um die 3D-Kamerakoordinaten auf die 2D-Bildecke zu Projizieren muss man lediglich die Tiefeninformation der Punkte $PK1$ und $PK2$ entnehmen und durch die homogene Koordinate ersetzen. Die folgende Gleichung zeigt ein Beispiel dazu

$$aK2 = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ z \\ 1 \end{pmatrix} \rightsquigarrow aBK2 = \begin{pmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{pmatrix} \quad (4.16)$$

4.5 Umrechnung von Bildebenenkoordinaten in Sensorkoordinaten

Für die Umrechnung der Bildebenenkoordinaten $BK1$ und $BK2$ in Sensorkoordinaten, muss der sogenannte Pixelpitch des Sensors bekannt sein. Nehmen wir für unser Beispiel man an wir haben einen PixelPitch von 1. Dann bedeutet dies, dass die Bildebenenkoordinaten in mm 1:1 in die Sensorkoordinaten in Pixel umgesetzt werden können. In der Realität ist dies aber eher selten der Fall.

Das Sensorkoordinatensystem wird folgendermaßen beschrieben:

$$K_s = (\vec{u}, \vec{v}, O_s) \quad (4.17)$$

$$\vec{u} = u_1 b_1 + u_2 b_2 \quad (4.18)$$

$$\vec{v} = v_1 b_1 + v_2 b_2 \quad (4.19)$$

$$O_s = O_B + p_1 b_1 + p_2 b_2 \quad (4.20)$$

$$(\vec{u}, \vec{v}, O_s) = (b_1, b_2, O_B) \cdot \begin{bmatrix} u_1 & u_2 & p_1 \\ v_1 & v_2 & p_2 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.21)$$

$$M_S = \begin{bmatrix} u_1 & u_2 \\ v_1 & v_2 \end{bmatrix} \quad (4.22)$$

Stellen wir nun ${}_{K_s} [\pi]_K$ dar

$$\begin{bmatrix} M^{-1} & -M \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}^{-1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -\zeta & 0 & 0 & 0 \\ 0 & -\zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -\zeta M^{-1} & -M \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}^{-1} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad (4.23)$$

$$aSK2 = \begin{bmatrix} -\zeta M^{-1} & -M \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}^{-1} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \cdot aBK2 \quad (4.24)$$

Als Beispiel nehmen wir an wir haben einen Pixelpitch p und es gilt $\vec{u} = 1pb_1$ und $\vec{v} = 2pb_2$. Des Weiteren sei $p_1 = 15$ und $p_2 = 20$.

$$O_s = O_B - \vec{u} - \vec{v} \rightsquigarrow O_s = O_B - 15b_1 - 20b_2 \quad (4.25)$$

$$M_S = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \rightsquigarrow M^{-1} = \begin{bmatrix} \frac{1}{p} & 0 \\ 0 & \frac{1}{2p} \end{bmatrix} \quad (4.26)$$

$$\rightsquigarrow [\pi] = \begin{bmatrix} \frac{\zeta}{p} & 0 & 15 & 0 \\ 0 & \frac{\zeta}{2p} & 20 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.27)$$

$$(4.28)$$

4.6 Ermitteln der Fundamentalmatrix mit Hilfe des 8-Point-Algorithms

Sind die Sensorkoordinaten berechnet, kann nun die Fundamentalmatrix mit Hilfe des 8-Point-Algorithms ermittelt werden. Für die Fundamentalmatrix benötigen wir mindestens sieben Punktekorrespondenzen. Mit sieben Punkten bekommen wir als Lösung des 8-Point-Algorithmus zwei linear unabhängige

Kerne als Lösung der Koeffizientenmatrix mit welchen weiter verfahren werden muss (Mehr dazu später). Hartley und Zisserman schlagen vor, dass sich zur Sicherheit am besten neun Punkte eignen. Zu beachten ist nämlich, dass wenn die Punkte in sofern voneinander abhängen, dass immer 2 Punkte auf der selben Epipolarlinie liegen, verliert unsere Koeffizientenmatrix an Rang und wir bekommen wie bei den sieben Punkten zwei Lösungen für den Kern. Aufgrund dessen haben wir einen neunten Punktd2 zu unserem Quader hinzugefügt, welcher nicht Gefahr läuft auf der selben Epipolarlinien wie ein anderer Punkt zu liegen.

Wir haben also nun unsere 9 Punkte auf dem Sensor der Kamera 1 SK_1 und dem Sensor der Kamera 2 SK_2 . Die Fundamentalmatrix ist definiert als

$$x'^T F x = 0 \quad (4.29)$$

Die Fundamentalmatrix ist eine 3×3 -Matrix mit Rang 2. Um sie zu ermitteln muss zunächst eine Koeffizientenmatrix nach folgendem Schema erstellt werden:

$$F = \begin{bmatrix} f_{11} & f_{122} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \quad (4.30)$$

$$(x'_n \ y'_n \ 1) \cdot \begin{bmatrix} f_{11} & f_{122} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \cdot \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = 0 \quad (4.31)$$

$$f_{11}x_nx'_n + f_{12}y_nx'_n + f_{13}x'_n + f_{21}x_ny'_n + f_{22}y_ny'_n + f_{23}y'_n + f_{31}x_n + f_{32}y_n + f_{33} = 0 \quad (4.32)$$

$$(x_nx'_n, y_nx'_n, x'_n, x_ny'_n, y_ny'_n, y'_n, x_n, y_n, 1) \cdot f = 0 \quad (4.33)$$

$$\begin{bmatrix} x_1x'_1 & y_1x'_1 & x'_1 & x_1y'_1 & y_1y'_1 & y'_1 & x_1 & y_1 & 1 \\ x_2x'_2 & y_2x'_2 & x'_2 & x_2y'_2 & y_2y'_2 & y'_2 & x_2 & y_2 & 1 \\ \vdots & \vdots \\ x_nx'_n & y_nx'_n & x'_n & x_ny'_n & y_ny'_n & y'_n & x_n & y_n & 1 \end{bmatrix} \cdot \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0 \quad (4.34)$$

Der Kern und alle seine Vielfache sind Lösungen für die Fundamentalmatrix. Um den Kern aus der Koeffizientenmatrix zu berechnen wird der null-space oder auch Nullraum ermittelt. Der null-space beschreibt denjenigen Vektor, welcher durch multiplizieren mit der Koeffizientenmatrix gleich den Nullvektor ergibt.

$$A \cdot f \quad (4.35)$$

Also Kern bekommt man in diesem Fall eine Liste mit Einträgen. Diese neun Einträge ergeben dann die Werte der 3×3 -Fundamentalmatrix.

4.7 Berechnen der Epipole und Epipolargeraden mit der Fundamentalmatrix

Mit Hilfe der Fundamentalmatrix und dem Wissen über die Epipolargerometrie, kann man die Epipole e und e' , sowie die Epipolargeraden l und l' ermitteln. Im ersten Abschnitt wird die Ermittlung mit

Hilfe der Fundamentalmatrix aufgezeigt und in Kapitel 1.2.1 wird nochmal drauf eingegangen und aufgezeigt, wie sich die Epipolarlinien und Epipole geometrisch Konstruieren lassen.

Mit der Fundamentalmatrix lassen sich die Epipolgerade folgendermaßen berechnen

$$l' = F \cdot x \quad (4.36)$$

$$l = F^T \cdot x' \quad (4.37)$$

l' ist die zu x korrespondierende Epipolgerade. l ist die zu x' korrespondierende Epipolgerade. Zu Berechnung des Epipols e muss der Rechte Null-Space von F ermitteln werden und für den Epipol e' brauchen wir den linken Null-Space. Um diesen zu bekommen ermitteln wir wie bekannt den Kern aber diesmal von F^T statt F . Die Abbildung 4 zeigt, dass Ergebnis.

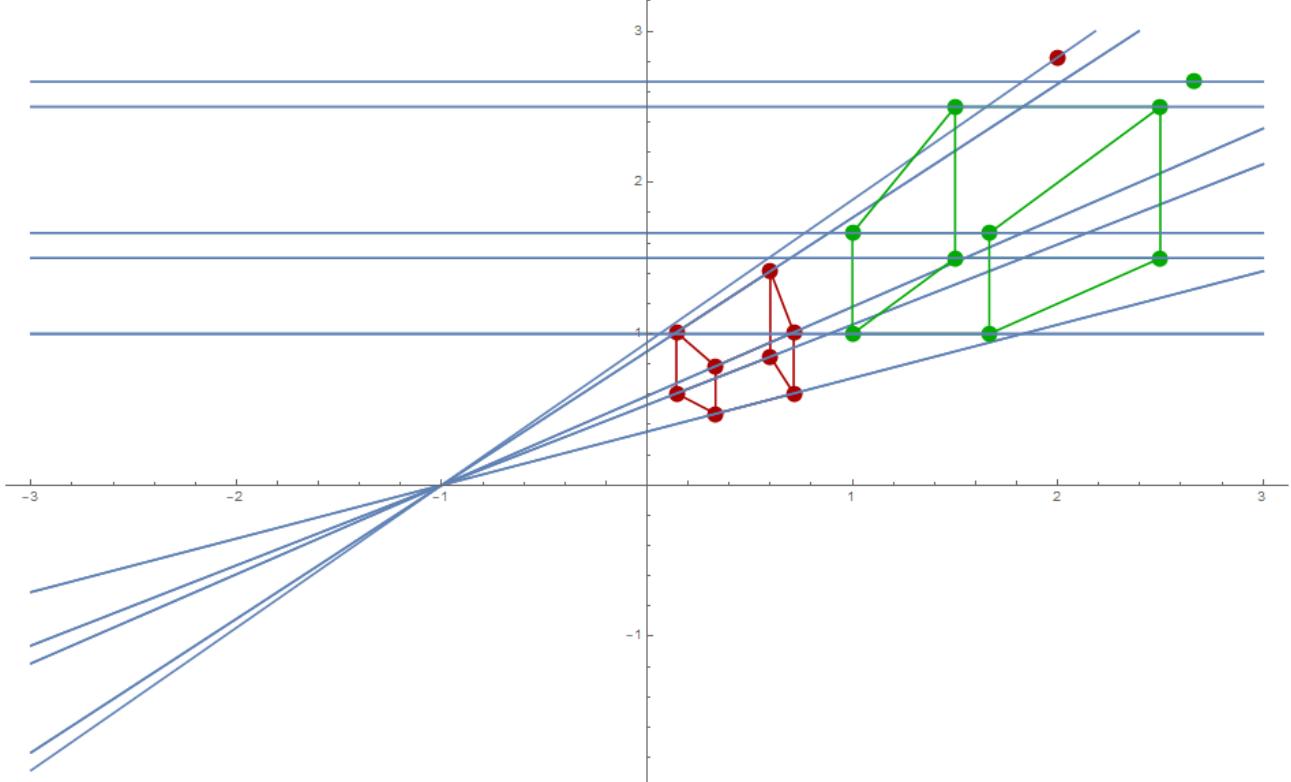


Abbildung 4.4: Die blauen Geraden zeigen die jeweiligen Epipolgeraden. Die vom roten Quader schneiden sich bei -1 im Epipol. Die Epipolgeraden vom grünen Quader schneiden sich im Epipol im Unendlichen

4.7.1 Konstruktion der Epipole und der Epipolgeraden auf Grundlage der Epipolargeometrie

Die Vorgehensweise zur Ermittlung der Epipole und der Epipolgeraden ist zwar schnell aber zur Verdeutlichung der geometrischen Beziehungen untereinander wird hier nochmal eine rein geometrische Konstruktion der Epipole und der Epipolgeraden aufgezeigt. Im nachfolgenden Beispiel wird der Epipol e' geometrisch konstruiert.

Vorbedingungen: Wir berechnen alles in Weltkoordinaten, hierzu muss sichergestellt sein dass alle Werte im Weltkoordinaten angegeben sind oder gegebenenfalls noch umgerechnet werden müssen.

Für die Konstruktion der Epipole brauchen wir folgendes:

- Die Basisgerade zwischen den Kamerazentren, welche sich un unserem Beispiel mit dem Ursprung des Kamerakoordinatensystems decken
- einen Punkt auf der Bildebne in Bildebenenkoordinaten (mm), nicht in Sensorkoordinaten, der Kamera 2 ,welcher Weltkoordinaten umgerechnet werden muss
- Die Bildebene der Kamera 2

Für unser Beispiel nennen wir das Kamerazentrum der Kamera 1 O_c und das Kamerazentrum der Kamera 2 O_{c2} (vgl mit vorherigen Dokumentationen). Wir erstellen auf Grundlage dieser zwei bekannten Punkte eine Gerade, welche durch die beiden Kamerazentren geht

$$BaseLine = O_{c2} + t \cdot (O_{c2} - O_c) \quad (4.38)$$

Um den Epipol zu ermitteln müssen wir nun den Schnittpunkt der Gerade $BaseLine$ mit der Bildebenen $ImagePlane'$ der Kamera 2 finden. Also müssen wir jetzt die Ebenengleichung der Bildebenen aufstellen. Am einfachsten ist es wenn man zunächst die Normalenform aufstellt.

$$\vec{n}_0 \cdot [\vec{x} - \vec{a}] \quad (4.39)$$

Nehmen wir das bereits bekannte Beispiel von vorhin, in welchem die Kamera 2 entlang der positiven e_1 verschoben und um 45° um die e_2 -Achse gedreht wurde, so ist der normalen Vektor der Bildebene aus

sicht der Weltkoordinaten $\vec{n}_0 = \begin{pmatrix} -1 \\ 0 \\ -1 \end{pmatrix}$. Für die Ebene brauchen wir nun noch einen Punkt welcher auf

ihr liegt. In unserem Beispiel nehmen wir als Aufpunkt \vec{a} den Punkt a in Bildkoordinaten $aBK2$ der 2. Kamera und rechnen diesen in Weltkoordinaten zurück. Wir brauchen also die nicht-transponierte Rotation R welche lautet:

$$R = \begin{pmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix} \quad (4.40)$$

Des Weiteren brauchen wir den verschiebe Vektor $\vec{v} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$, welcher mit R verrechnet wird.

$$\begin{pmatrix} \cos \alpha & 0 & \sin \alpha \\ 0 & 1 & 0 \\ -\sin \alpha & 0 & \cos \alpha \end{pmatrix} \cdot \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} = \begin{pmatrix} v_1 \cos \alpha + v_3 \sin \alpha \\ v_2 \\ -v_1 \sin \alpha + v_3 \cos \alpha \end{pmatrix} \quad (4.41)$$

$$\rightsquigarrow D = \begin{bmatrix} \cos \alpha & 0 & \sin \alpha & v_1 \cos \alpha + v_3 \sin \alpha \\ 0 & 1 & 0 & v_2 \\ -\sin \alpha & 0 & \cos \alpha & -v_1 \sin \alpha + v_3 \cos \alpha \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.42)$$

Matrix D wird dann mit Punkt $aBK2$ multipliziert und wir bekommen den Punkt $aBK2$ in Weltkoordinaten $aWK2$. Die Normalengleichung ist somit komplett aufgestellt. um auf die Koordinatengleichung zu kommen werden die Skalare einfach ausmultipliziert.

$$\vec{n}_0 \cdot \vec{x} - \vec{n}_0 \cdot \vec{a} \quad (4.43)$$

$$ImagePlane : ax + by + cz + d = 0 \quad (4.44)$$

Danach wird die Geradengleichung *BaseLine* in die Ebenengleichung *ImagePlane* eingesetzt und es wird ein Wert für t ermittelt. Dieser wird wiederum in die Geradengleichung eingesetzt und das Ergebnis ist der Epipol e' , jedoch noch in Weltkoordinaten. Dieser muss dann nach dem selben Schema wie am Anfang gezeigt von Welt in Sensorkoordinaten umgerechnet werden.

$$e'_{Sensor} = PM2 \cdot e'_{Welt} \quad (4.45)$$

Die dritte Zeile kann dadurch dass wir eine 1 zu 1 umsetzung von Bildebenenkoordinaten (mm) auf Sensorkoordinaten (Pixel) haben einfach rausgestrichen und durch die vierte Zeile ersetzt werden.

4.8 Ermitteln der Essentiellen Matrix über die Fundamentalmatrix

Nachdem nun die Fundamentalmatrix haben und Epipole und Epipolarlinien bestimmt sind, wollen wir mit Hilfe der Fundamentalmatrix die Essentielle Matrix bestimmen, aus welcher wir die externen Kameraparameter extrahieren wollen.

$$\hat{x}'^T \cdot E \cdot \hat{x} = 0 \quad (4.46)$$

In unserem Minimalbeispiel sind innere und äußere Kameraparameter ja bereits bekannt, somit können wir leichter erkennen ob unsere Ergebnisse mit den vordefinierten Werten übereinstimmen. Wir gehen also jetzt davon aus, dass wir die externen Kameraparameter noch nicht kennen. Wir kennen die Fundamentalmatrix F und wir kennen die inneren Kameraparameter $AB1$ und $AB2$ (vgl. Kapitel 1.2). Anzumerken ist noch, dass wir für die Essentielle Matrix nicht die Sensorkoordinaten, sondern die Bildebenenkoordinaten betrachten, welche zuvor auch noch normiert werden müssen. In Gleichung 46 ist dies durch \hat{x}' und \hat{x} gekennzeichnet. Der Normierungsvorgang sieht folgendermaßen aus.

$$aK2 = PM2 \cdot a \quad (4.47)$$

$$aK2 = AB2[R|t] \quad (4.48)$$

$$AB2^{-1} \cdot aK2 = AB2^{-1} \cdot AB2[R|t]a \quad (4.49)$$

$$a\hat{K}2 = [R|t] \quad (4.50)$$

$$[R|t] \hat{M} \quad (4.51)$$

$a\hat{K}2$ beschreibt die normierte Koordinate von $aK2$. Um die Essentielle Matrix aus der Fundamentalmatrix herzuleiten benutzen wir folgende Formel.

$$E = AK2^T \cdot F \cdot AK1 \quad (4.52)$$

Die Lösung dieser Gleichung gibt uns eine Mögliche Lösung der Essentiellen Matrix. Vergleichen wir das Ergebnis mit dem Ergebnis welches wir bekommen, wenn die Essentielle Matrix über den 8-Point-Algorithm berechnet wurde, so kann man erkennen dass sie Vielfache voneinander sind.

4.8.1 Ermitteln der Essentiellen Matrix mit normierten Bildkoordinaten und dem 8-Point-Algorithm

Im vorherigen Abschnitt haben wir bereits die normierten Bildebenenkoordinaten berechnet. Um die essentielle Matrix mit dem 8-Point-Algorithm zu bestimmen, verfährt genau so wie bei der Bestimmung der Fundamentalmatrix nur verwendet man statt den Sensorkoordinaten die normierten Bildebenenkoordinaten.

(vgl Hartley and Zisserman, O. Schreer)

Um herauszufinden, ob die errechnete Matrix den Kriterien einer Essentiellen Matrix entspricht gilt folgendes zu überprüfen.

- Zwei der Singulärwerte und der Eigenwerte der Essentiellen Matrix müssen gleich und von null verschieden sein.
- Die Quadratwurzel der Eigenwerte muss die Singulärwerte ergeben
- Der Rang der Matrix muss zwei sein

4.9 Ermitteln der externen Kameraparameter mit Hilfe der Essentiellen Matrix

Um nun die äußeren Kameraparameter zu bestimmen, muss zunächst die Essentielle Matirx E mit Hilfe der Singulärwertszerlegung zerlegt werden. Das Ergebnis sind drei Matrizen der Form

$$E = U\Sigma V^T \quad (4.53)$$

Zu Beachten ist dass die Essentielle Matrix so angepasst werden muss dass am besten für Σ gilt:

$$\Sigma = \text{diag}(1, 1, 0) \quad (4.54)$$

Mit angepasst ist damit gemeint, dass man das Vielfache des Kerns der Essentiellen Matrix nimmt, so dass die Singulärwerte $\text{diag}(1, 1, 0)$ werden.

Wir wissen dass:

$$E = [t]_x R \quad (4.55)$$

$$S = [t]_x \quad (4.56)$$

$$E = SR \quad (4.57)$$

Zur Unterstützung der Berechnung der externen Kameraparameter nehmen wir die Matrizen W und Z zu Hilfe

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (4.58)$$

Mit dem Ergebnis der SVD von E lassen sich nun jeweils zwei Ergebnisse für S und R finden:

$$S_1 = -UZU^T \quad R_1 = UW^TV^T \quad (4.59)$$

$$S_2 = UZU^T \quad R_2 = UWV^T \quad (4.60)$$

Um sicher zu gehen, dass es sich bei R_1 und R_2 auch um Rotationen handelt, kann man folgende Proben durchführen. Zum einen muss $R \cdot R^T = I_{3x3}$ sein. I_{3x3} bezeichnet in diesem Fall eine 3x3-Einheitsmatrix. Des Weiteren kann man überprüfen, ob die Determinante $\det(R_1) = 1$ ist. Nun müssen wir aus den 3x3-Matrizen S_1 und S_2 noch das fehlende t rausfiltern. Wir wissen:

$$St = [t]_x \cdot t = t \times t \quad (4.61)$$

Daraus kann man schließen, dass $t =$ dem Null-Space von S_1 und S_2 ist. Der Null-Space beider Matrizen S_1 und S_2 ist der selbe. Die externen Kameraparameter lassen sich wie gesagt nur bis zu

einem Skalierungsfaktor genau berechnen. Dass soll heißen, ist der t der Null-Space von S_1 und S_2 , so ist auch das Vielfache λt eine gültige Lösung für t . Letzten Endes haben wir für die externen Kameraparameter vier verschiedene Lösungen.

$$PM2 = [UWV^T] + \lambda t \quad \text{or} \quad [UW^T V^T] + \lambda t \quad (4.62)$$

$$\text{or} \quad [UWV^T] - \lambda t \quad \text{or} \quad [UW^T V^T] - \lambda t \quad (4.63)$$

4.10 Punkterekonstruktion durch Triangulation

4.10.1 Rektifizierung

5 Minimalbeispiel 3D-Stereokalibrierung und Szenenrekonstruktion bei Kameras unterschiedlicher Auflösung

5.1 Vorgehen: Projektion eines Quaders in zwei verschiedenen transformierte Kameras mit unterschiedlichen Auflösungen

5.2 Berechnung der Projektionsmatrizen

5.3 Transformation der Weltpunkte in Koordinaten der Koordinatensysteme von beiden Kameras

5.4 Berechnung der projizierten Punkte auf den beiden Bildebenen

was ist anders als zuvor Im Bezug auf die Darstellung davor und danach

5.5 Behauptung 1: Kameras unterschiedlicher Auflösung haben keine Auswirkung auf die Ermittlung der externen Kameraparameter

5.6 Ermitteln der Fundamentalmatrix mit Hilfe des 8-Point-Algorithms

5.7 Ermitteln der Essentiellen Matrix über die Fundamentalmatrix

5.8 Ermitteln der externen Kameraparameter mit Hilfe der Essentiellen Matrix

5.9 Behauptung 2: Durch Rektifizierung der Bilder kann eine reelle Triangulation erfolgen

5.10 Rektifizierung

5.11 Erstellen einer Tiefenkarte aus zwei rektifizierten Bildern

5.12 Punkterekonstruktion durch Triangulation

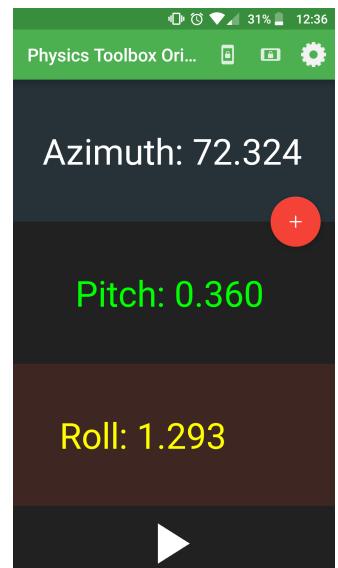
5.13 Vergleich der rekonstruierten Szenen

6 3D-Stereokalibrierung und Szenenrekonstruktion mit reellen Daten und Kameras gleicher Auflösung

Bis jetzt haben wir eine Stereoanalyse und Szenenrekonstruktion in einem wie wir es nennen Minimalbeispiel durchgeführt. Hierzu wurde zum Beispiel ein Quaderobjekt definiert und dessen Eckpunkte mit Hilfe von zwei selbst definierten Kameramatrizen verrechnet so, dass ein Stereobildpaar dieses Quaders entstand. Vorteil war, dass man die korrespondierenden Punkte somit bereits besaß und die Stereoanalyse und Szenenrekonstruktion ohne Fehlerquellen wie Beispielsweise Rauschen durchführen konnte. In unserem Beispiel unter Realbedingungen, müssen zunächst die korrespondierenden Punkte des Stereobildpaars ausgemacht werden, was bis jetzt, nicht besonders präzise, von Hand funktioniert. Zu dieser Ungenauigkeit kommen dann auch noch Fehler wie Rauschen hinzu. Wie hiermit bis jetzt verfahren wird, wird folgend beschrieben.

6.1 Vorgehen

Es wurde ein Stereosetup (Abbildungen 2 - 6) aufgebaut und die beiden Kameras Canon 6D und Canon 60D wurden so im Raum platziert, dass sie leicht zueinander hin gedreht waren. Die 6D wird als Ausgangskamera oder auch primär Kamera angesehen, die Position und Orientierung der 60D wird dann dementsprechend ausgehend von der 6D gemessen. Dem Voran gilt es unter anderem ein Koordinatensystem für die Canon 6D zu definieren. Danach werden die Abstände der 6D zur 60D entlang der Koordinatenachsen gemessen. Hier kann es zu deutlichen Abweichungen kommen, da von Hand mit einem Metermaß die Maße genommen wurde. Nachdem die Translation der 60D ausgehend von der 6D gemessen wurde, muss nun noch die Rotation der 60D gegenüber der 6D ermittelt werden. Da kein Gimbal zur Verfügung stand, wurde mit Hilfe einer App namens *Physics Toolbox Orientation* zunächst die Orientierung der 6D ungefähr abgemessen und danach die Orientierung der 60D und die Differenz der beiden Gierwinkel der Kameras zueinander genommen um abzuschätzen wie weit die 60D zur 6D eingedreht ist. In der Toolbox gibt es den sogenannten *Azimuth*-Winkel mit dessen Hilfe der Gierwinkel der beiden Kameras zueinander abgeschätzt wurde. Zur Überprüfung und weiteren Orientierung wurden zwei Metermaße auf den Boden gelegt und zwar so, dass sie quasi die Richtung der Kameramitte durch die Objektivmitte bilden. Der Schnittpunkt dieser beiden Metermaße sollte dann dazu dienen den Winkel abzumessen. Da die Metermaße nicht ganz gereicht haben, wurde ein Bild der beiden Enden so parallel wie möglich zum Boden aufgenommen und in Geogebra die Linien der Metermaße vervollständigt, so dass der Winkel zwischen dem Schnittpunkt ausgegeben werden konnte (vgl Abbildung 1). Wie man im Protokoll sehen kann stimmen die gemessenen Winkel größtenteils überein. Rotationen um Roll- und Nickwinkel konnten leider nicht genau gemessen werden, weshalb diese auf Null gesetzt wurden. Die Daten wurden in das folgende Protokoll eingetragen.



6D links

Pixelpitch = 6.5 µm

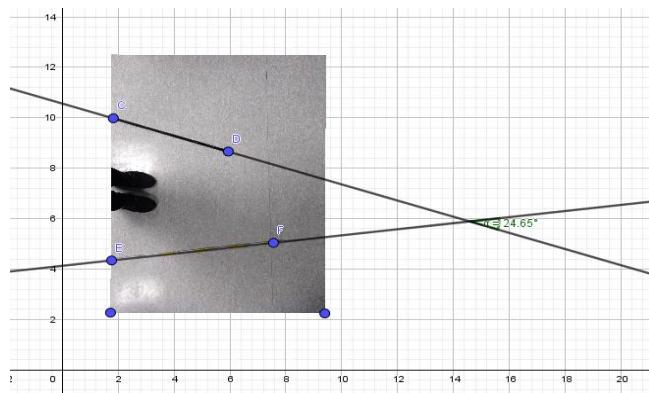
K in px	2663,969662	0	0
	0	2664,587487	0
	945,6292097	707,7870042	1
K in mm (transponiert)	17,3158028	0	6,146589863
	0	17,31981867	4,600615527
	0	0	1

Rotation

Rotation um y von 6D aus

Gemessen aus Bild

ca .24,65°



Orientation	1 Messung	2.Messung	3.Messung
6D			
Azimuth (facing north)	72,324	74.224	72.822
Pitch	0		
Roll	0		
60D			
Azimuth (facing north)	100,736	101,06	100.086
Pitch	0		
Roll	0		

Aus VibraTilt	1 Messung	2.Messung	3.Messung
	rad/s	rad/s	rad/s
VT x ist unser x	0.2	0.17	0.2
VT z ist unser y	0.5	0.57	0.57
VT Y ist unser z	0.075	0.3	0.2

60D rechts PixelPitch = 4.29 µm

K in px	4337,371	0	0
	0	4332,858039	0
	966,3522	752,2296946	1
K in mm (tr)	18,60732	0	4,145650968
	0	18,58796099	3,22706539
	0	0	1

Koordinatensystem

von hinter der Kamera x -> rechts

Y -> unten

z -> vorne

Abstände 60D aus Koordinatensystem von 6D

x = 150-160 cm (+)
y = ca 0,7 - 1 cm (-)
z = ca 50 - 55 cm(-)

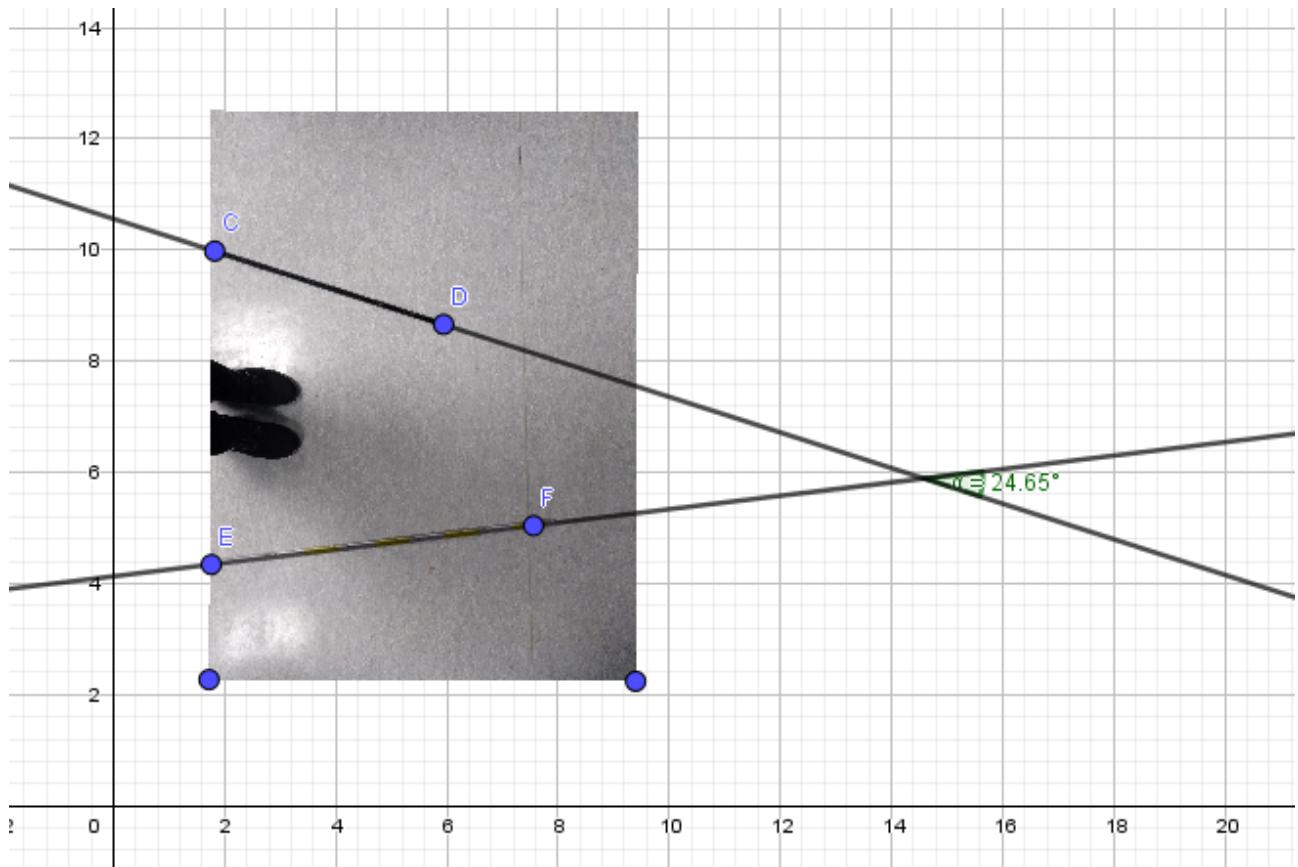


Abbildung 6.1: Rekonstruktion des Gier-Winkels der Kameras zueinander

Mit Hilfe von Matlab wurden dann für beide Kameras Kamerakalibrierungen durchgeführt um die intrinsischen Kameramatrizen beider Kameras zu bekommen. Diese benötigen wir für die spätere Ermittlung der Essentiellen Matrix und können die Kalibrierung bis dato noch nicht selbst durchführen. Jedoch besteht hier auch die dringende Überlegung die Ermittlung der kompletten Kameramatrix allein mit der Fundamentalmatrix durchzuführen, um eine mögliche Fehlerquelle auszuschließen, weiteres zu dieser Überlegung im Unterkapitel Probleme und mögliche Ansätze für Lösungen Mit den beiden Kameras wurde jeweils zeitgleich Bilder einer Szene aufgenommen (Abbildungen 5 und 6). Aus diesem Stereobildpaar wurden dann von Hand mögliche Korrespondierende Punkte rausgesucht und in Mathematica als zwei separate Punktelisten gespeichert. Nachdem alle Daten gesammelt wurden, wurde der bereits bestehende Algorithmus zur Stereokalibrierung und Szenenrekonstruktion unseres Minimalbeispiels auf die Bildpunkte angewandt. Daraufhin mussten auf Grund von fehlerhaften Resultaten bestimmte Modifizierungen vorgenommen werden, welche im KapitelAlgorithmus genauer beschrieben werden.

6.2 Aufbau der Set-Ups



Abbildung 6.2: Kamera 6D



Abbildung 6.3: Kamera 60D

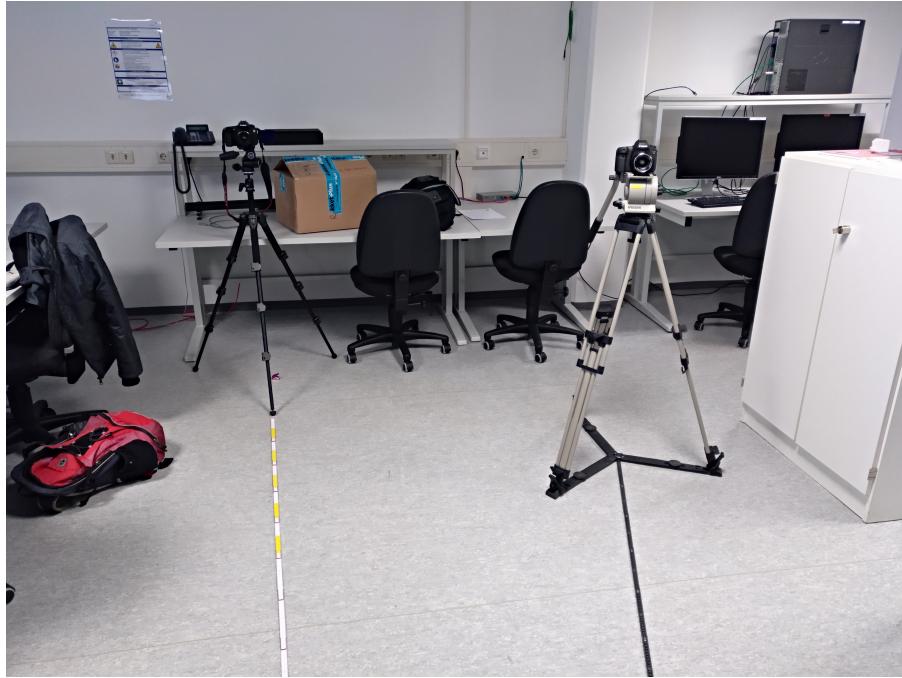


Abbildung 6.4: Stereosetup

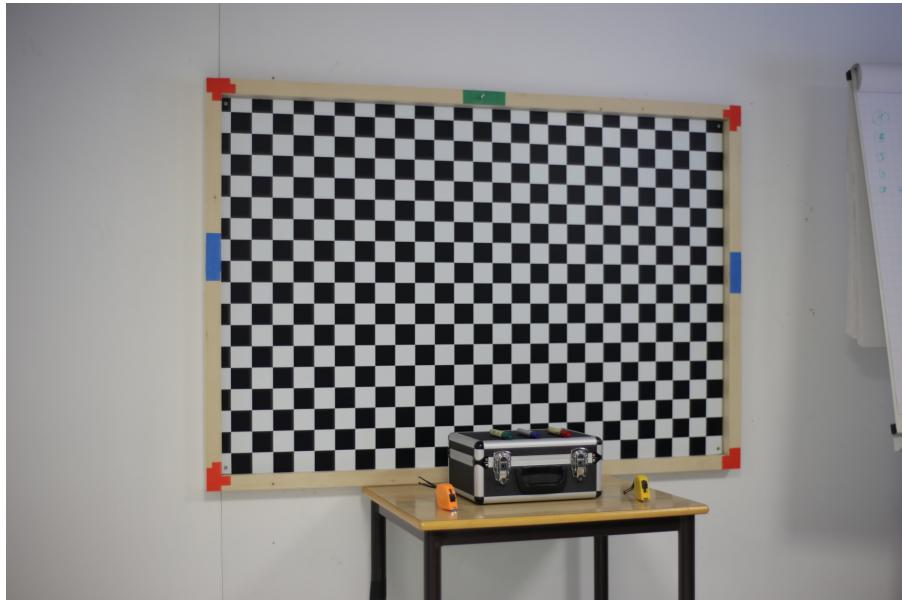


Abbildung 6.5: Szenenbild der primären Kamera 6D



Abbildung 6.6: Szenenbild der sekundären 60D

6.3 Unterschiede im Algorithmus im Vergleich zum Minimalbeispiel

StereoAnalyse.nb beinhaltet den kompletten Code von der Ermittlung der Fundamentalmatrix über die Umrechnung in die essentiellen Matrix und letztendlich die Rekonstruktion der externen Kameraparameter. Die Datei *Points_SetUp.nb* beinhaltet die ausgelesenen korrespondierenden Punkte und die Ergebnisse der Berechnungen werden in der Datei *results.nb* angezeigt.

Der erste Schritt des Algorithmus beinhaltet die Ermittlung der Fundamentalmatrix mit dem normalized-8-Point-Algorithm. Die Entscheidung den normalized-8-Point-Algorithm zu benutzen fiel als festgestellt wurde, dass ohne vorherige Normalisierung der ausgelesenen Punkte es zu größeren Fehlern in den weiteren Berechnungen kam. Zur Erklärung dieser Fehler kann zum einen die *Condition Number* betrachten [7], welche in der *SVD* der entstehenden Matrix aus der Koeffizientenmatrix mit ihrer transponierten ausgelesen werden kann. Diese sollte möglichst klein sein, jedoch ist sie bei nicht normalisierten Koordinaten relativ hoch. Der Grund für den nicht optimalen Wert der *Condition Number* ist das Fehlen der Homogenität in den Bildkoordinaten.[7]. Die weitere Erklärung findet sich im Paper von Richard I. Hartley unter Kapitel 4. Hinzuzufügen ist noch, dass je größer die *Condition Number* ausfällt, desto größer wirken sich kleinste Abweichungen der Daten wie beispielsweise Rauschen in den Bildern auf die Resultate aus.

6.3.1 normalisierung der eingehenden Daten und Berechnung der Fundamentalmatrix über den normalized 8-Point-Algorithm

Nach diesem Exkurs zurück zum Algorithmus. Die Bildpunkte werden also zu aller erst Normalisiert. Hierzu wurde ein Modul geschrieben, welches für die Bildpunktpaare beider Bilder zwei Matrizen aufstellt, die eine Skalierung und Translation der Punkte des jeweiligen Bildes bewirkt. Normalisieren bedeutet in diesem Falle, dass die Schwerpunkt der Punktwolken pro Bild ermittelt wurden und diese dann in den Ursprung des jeweiligen Bildes verschoben wurden, danach wurden die Punkte ebenfalls mit Beibehaltung ihres Abstandsverhältnisses zum Schwerpunkt mit verschoben. Als nächstes wurde der Durchschnittsabstand der Punkte zum neu gelegenen Schwerpunkt auf $\sqrt{2}$ gebracht. Die entstandenen Punkte befinden sich nun in einem deutlich kleineren Zahlenbereich als zuvor meist zwischen ca -1 und 1. danach werden diese normierten Punkte an ein Modul namens *FundamentalMtxBestimmung* übergeben zusammen mit den beiden Normalisierungsmatrizen T und T' , welche später für

die Denormalisierung der Fundamentalmatrix erneut benötigt werden.

Im Modul *FundamentalMtxBestimmung* wird eine Koeffizientenmatrix nach den Vorgaben aus *Hartley and Zisserman* erstellt[2].

$$x'^T F x = 0 \quad (6.1)$$

$$F = \begin{bmatrix} f_{11} & f_{122} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \quad (6.2)$$

$$(x'_n \ y'_n \ 1) \cdot \begin{bmatrix} f_{11} & f_{122} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \cdot \begin{pmatrix} x_n \\ y_n \\ 1 \end{pmatrix} = 0 \quad (6.3)$$

$$f_{11}x_nx'_n + f_{12}y_nx'_n + f_{13}x'_n + f_{21}x_ny'_n + f_{22}y_ny'_n + f_{23}y'_n + f_{31}x_n + f_{32}y_n + f_{33} = 0 \quad (6.4)$$

$$(x_nx'_n, y_nx'_n, x'_n, x_ny'_n, y_ny'_n, y'_n, x_n, y_n, 1) \cdot f = 0 \quad (6.5)$$

$$\begin{bmatrix} x_1x'_1 & y_1x'_1 & x'_1 & x_1y'_1 & y_1y'_1 & y'_1 & x_1 & y_1 & 1 \\ x_2x'_2 & y_2x'_2 & x'_2 & x_2y'_2 & y_2y'_2 & y'_2 & x_2 & y_2 & 1 \\ \vdots & \vdots \\ x_nx'_n & y_nx'_n & x'_n & x_ny'_n & y_ny'_n & y'_n & x_n & y_n & 1 \end{bmatrix} \cdot \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0 \quad (6.6)$$

Im Idealfall und auch in unserem Fallbeispiel ist der Rang dieser Koeffizientenmatrix bei 8, jedoch durch Ungenauigkeiten der Bildpunkte durch Rauschen und da es sich um einen überbestimmten Fall mit mehr also nur acht korrespondierenden Punkten handelt ist der Rang bei 9, weshalb wenn man sich den Kern dieser Matrix ausgeben lässt, man für F keine Eindeutige Lösungen bekommt.

```
CoefficientMtx = 
1.87631  3.05337 -1.41978  2.48713  4.04738 -1.88199 -1.32154 -2.15059  1.
1.75575  2.48496 -1.37334  2.02505  2.86611 -1.58399 -1.27845 -1.80943  1.
2.09688  2.06386 -1.49719  1.67642  1.65002 -1.19698 -1.40055 -1.37849  1.
1.57686 -0.460803 -1.33464 -0.41022  0.119877  0.347206 -1.18149  0.345264  1.
2.07001 -0.821263 -1.49332 -0.754891  0.299498  0.544582 -1.38618  0.549959  1.
0.735452 -0.485137 -0.943758 -0.376127  0.24811  0.48266 -0.77928  0.514048  1.
0.0154866 0.0227874  0.0431243  0.163602  0.240728  0.455569  0.359115  0.528412  1.
-0.0874237 0.238019  0.283072 -0.243489  0.66292  0.7884 -0.308839  0.840843  1.
0.481839  0.180028  0.793929  0.112071  0.0418728  0.18466  0.606905  0.226756  1.
0.476137  0.544971  0.793929  0.352129  0.403036  0.587154  0.599722  0.686423  1.
0.853263  0.33676   1.00679  0.241781  0.0954246  0.285284  0.847512  0.33449  1.
3.03148   0.396832  1.72276  0.277269  0.0362956  0.157569  1.75966  0.230347  1.
2.93569   0.660029  1.69954  0.506153  0.113798  0.293024  1.72734  0.388357  1.
3.0185    1.19223   1.71889  0.942734  0.372356  0.536842  1.75607  0.693605  1.
```

Rank CoefficientMtx = 9

Abbildung 6.7: Beispiel einer resultierenden Koeffizientenmatrix

Aus diesem Grund beschaffen wir uns die Fundamentalmatrix etwas anders. Es wird eine *SVD* der Koeffizientenmatrix durchgeführt, um eine *least-square-solution* zu finden. Die *least-square-solution* für f ist der Singulärvektor entsprechend dem kleinsten Singulärwert der Koeffizientenmatrix, welcher der letzten Spalte von V entspricht. Die Fundamentalmatrix ist hiermit aber noch nicht vollständig bestimmt. Das Resultierende F hat nämlich den Rang 3 und ist somit keine gültige Fundamentalmatrix. Diese muss singulär sein und den Rang 2 besitzen. Es muss also ein *singularity-constraint* erzwungen werden, welcher uns eine Fundamentalmatrix \hat{F} mit Rang 2 aus F ermittelt. Das Problem ist nämlich, dass wenn wir mit F uns Beispielsweise die Epipolarlinien ausrechnen wir folgende Plots der beiden

Bilder erhalten.

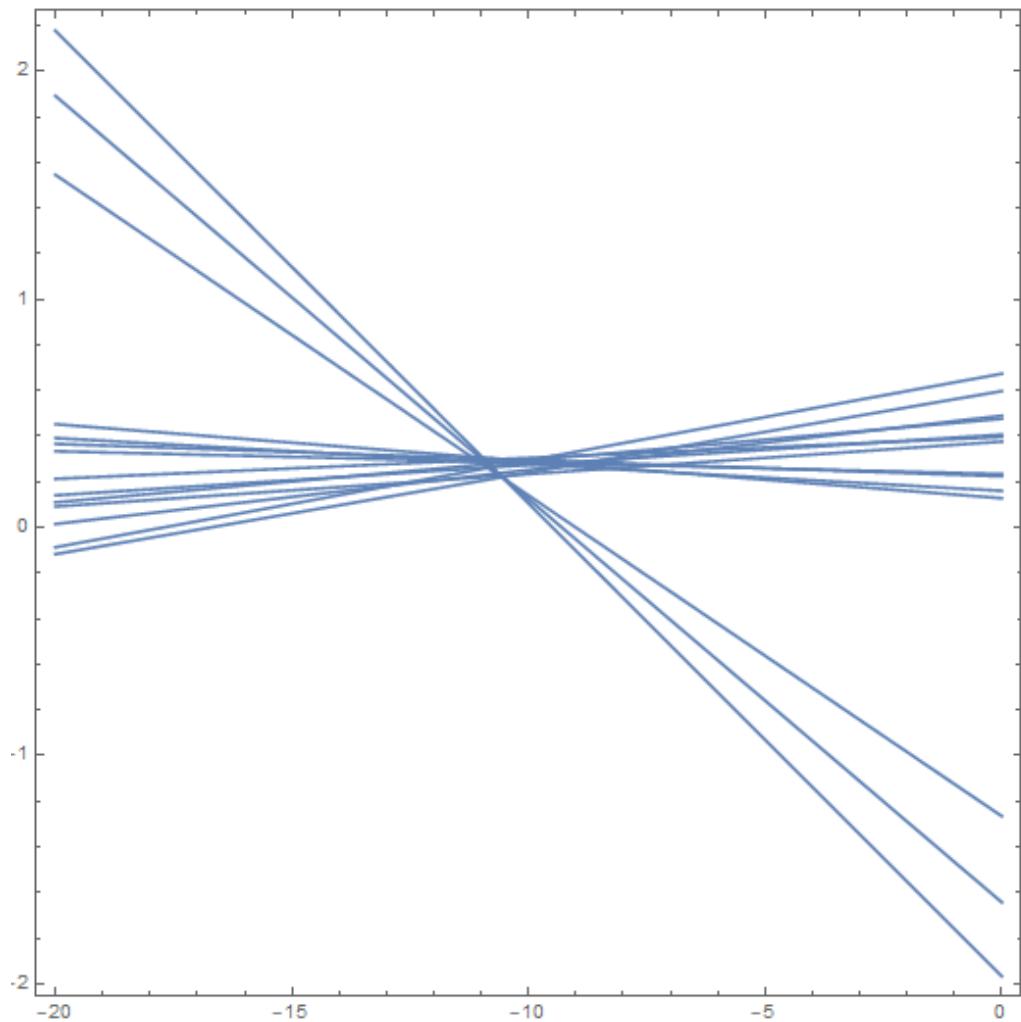


Abbildung 6.8: Resultierende Epipolarlinien des Bildes der Canon 6D

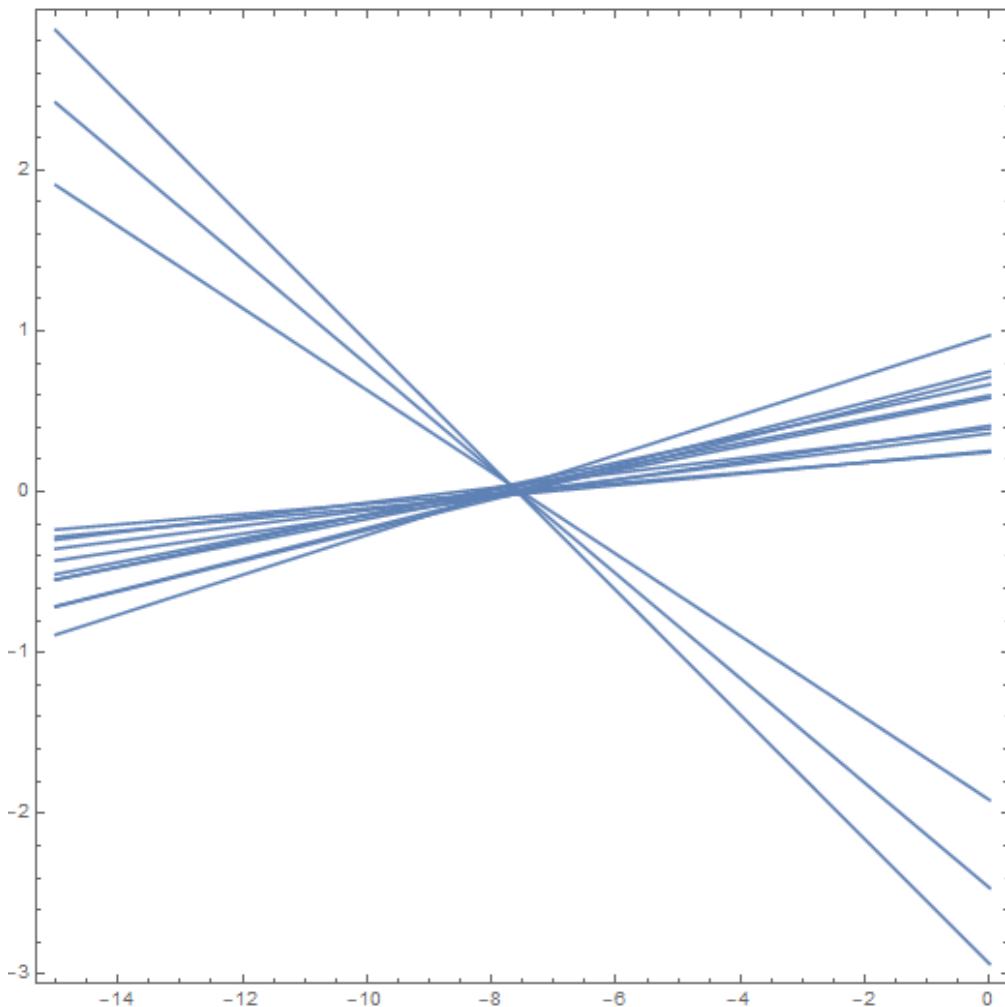


Abbildung 6.9: Resultierende Epipolarlinien des Bildes der Canon 60D

Wie man auf den Bildern erkennen kann, treffen sich die Epipolarlinien nicht in einem Epipol. Die Erzwingung einer Singularität von F soll dieses Problem beheben. Auch hier halten wir uns an das Verfahren welches im *Hartley und Zisserman* [2] beschrieben wird. Wir führen an F eine *SVD* durch und bekommen F in UDV^T zerlegt. D beinhaltet in einer diagonalen Matrix die Singulärwerte $D = \text{diag}(r, s, t)$, welche die Bedingung $r \leq s \leq t$ erfüllen. Um den *singularity-constraint* zu erzwingen, setzen wir den letzten Singulärwert $t = 0$ und erhalten somit $D = \text{diag}(r, s, 0)$. Danach wird \hat{F} wieder zusammengesetzt mit dem modifizierten D und bekommen $\hat{F} = UDV^T$ welches die Frobenius-Norm von $F - \hat{F}$ minimiert [2]. Das neue \hat{F} führt, wenn wir uns wieder die Epipolarlinien ausgeben lassen zu folgendem Ergebnis.

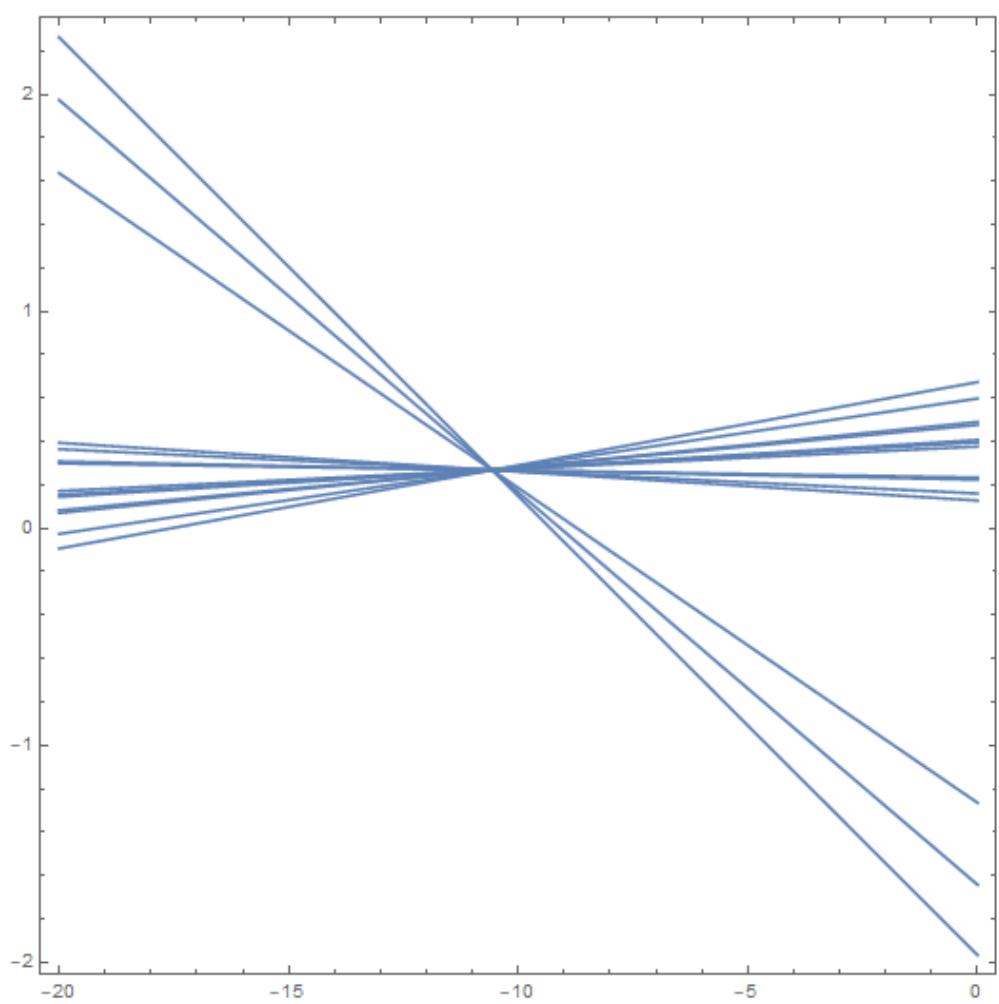


Abbildung 6.10: Resultierende Epipolarlinien des Bildes der Canon 6D nach singularity-constraint

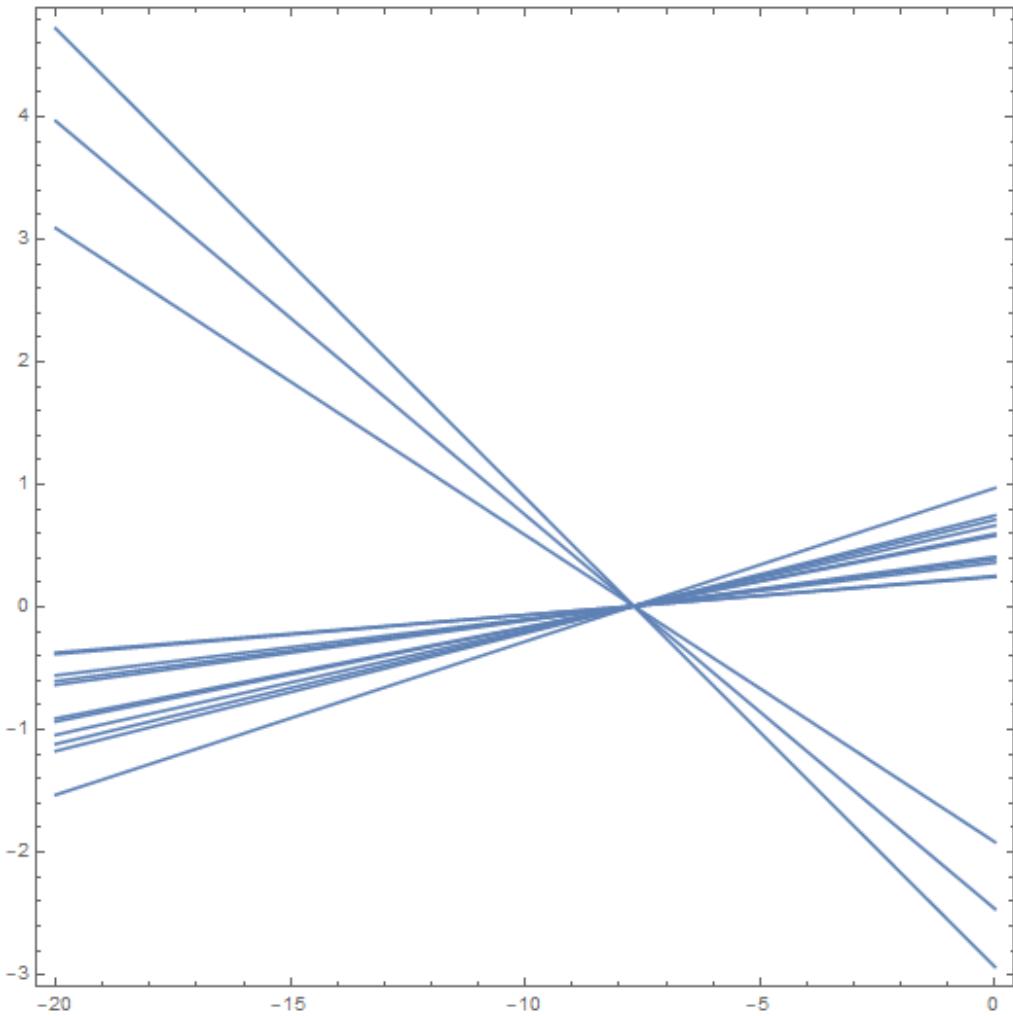


Abbildung 6.11: Resultierende Epipolarlinien des Bildes der Canon 60D nach singularity-constraint
Letztendlich wird dann \hat{F} noch mit Hilfe der beiden Matrizen T und T' denormalisiert

$$F = T'^T \hat{F} T \quad (6.7)$$

6.3.2 Ermitteln der Essentiellen Matrix und Einführung des singularity-constraints

Mit dieser Fundamentalmatrix und den aus Matlab erlangten internen Kameraparametern K_1 der Canon 6D und K_2 der Canon 60D kann nun auf zwei Verschiedene Weisen die essentielle Matrix berechnet werden. Hier werden beide Vorgänge kurz erläutert aber mit dem Vorbehalt, dass die essentielle Matrix in unserem Realbeispiel wohl nicht gänzlich korrekt ist, da diese uns nicht die gewünschten Resultate liefert. Mehr dazu später.

In der Theorie kann die Essentielle Matrix durch Verrechnung der beiden Kameramatrizen K_1 und K_2 mit der denormalisierten Fundamentalmatrix F erschlossen werden.

$$E = K_2^T F K_1 \quad (6.8)$$

Zum anderen kann aber auch so vorgehen wie bei der Berechnung der Fundamentalmatrix nur müssen hierzu die Koordinaten zunächst von Pixel in Millimeter umgerechnet werden und danach noch mit den Kameramatrizen normalisiert werden. Ist dies erfolgt kann das selbe Verfahren wie bei der Fundamentalmatrix mit dem *8-Point-Algorithm* angewandt werden.

Schon nach Berechnung der essentiellen Matrix zeigt sich in unserem Beispiel eine Ungenauigkeit. Denn bei der Überprüfung ob gilt dass:

$$\hat{x}'E\hat{x} = 0 \quad (6.9)$$

Kommt es doch zu einer nicht ignorierbaren Ungenauigkeit. Die Werte befinden sich in einem Bereich zwischen -0.3 und -0.7. Betrachtet man von E die *SVD*, so wird ersichtlich, dass sie ersten beiden Singulärwerte nicht identisch sind, was aber bei einer gültigen Essentiellen Matrix der Fall sein muss [2]. Auch hier wird einem nahegelegt diesen *constraint* zu erzwingen. E wird durch die *SVD* in $E = UDV^T$ zerlegt. Auch hier sind die Singulärwerte in der Diagonalmatrix $D = \text{diag}(a, b, c)$ mit $a \leq b \leq c$ gegeben. Um nun den gewünschten *singularity-constraint* zu erwirken wird D folgendermaßen modifiziert. $\hat{D} = \text{diag}(\frac{a+b}{2}, \frac{a+b}{2}, 0)$. Danach wird \hat{E} wieder zusammengesetzt mit $\hat{E} = UDV^T$. Doch trotzdem haben sich die Resultate kaum gebessert wie man in Abbildung 12 sehen kann.

```
-0.397993
-0.401145
-0.404757
-0.430126
-0.431955
-0.436821
-0.448127
-0.450752
-0.446406
-0.455935
-0.451189
-0.458417
-0.461546
-0.46867
```

Abbildung 6.12: Ergebnis der Prüfung der Punktekorrespondenzen mit der essentiellen Matrix

Auch den Versuch das Verfahren mit den bereits normalisierten Koordinaten für die Berechnung der Fundamentalmatrix zu verwenden oder die essentielle Matrix aus der nicht denormalisierten Fundamentalmatrix zu berechnen, hat leider keine Besserung des Ergebnisses erzielt. Vermutet werden kann, dass die ausgelesenen korrespondierenden Punkte vielleicht doch viel zu ungenau sind oder die Bilder gar leicht Verzeichnet sind, was zu einer Anhäufung der Fehler führen könnte. Nichtsdestotrotz wurde der Algorithmus weitergeführt und versucht die Rotation und Translation der Canon 60D zur 6D zu ermitteln. Hierzu wurde sich ebenfalls an dem Verfahren aus dem Buch von *Hartley and Zisserman* orientiert.

6.4 Rekonstruktion der externen Kameraparameter

Dem Modul *ExtractRotationAndTranslation* wird die Essentielle Matrix übergeben. Um nun die äußeren Kameraparameter zu bestimmen, muss zunächst die essentielle Matirx E mit Hilfe der *SVD* zerlegt werden. Das Ergebnis sind drei Matrizen der Form

$$E = U\Sigma V^T \quad (6.10)$$

Zu Beachten ist dass die Essentielle Matrix so angepasst werden muss dass am besten für Σ gilt:

$$\Sigma = \text{diag}(1, 1, 0) \quad (6.11)$$

Was zuvor durch den *constraint* erwirkt wurde. Wir wissen dass:

$$E = [t]_x R \quad (6.12)$$

$$S = [t]_x \quad (6.13)$$

$$E = SR \quad (6.14)$$

Zur Unterstützung der Berechnung der externen Kameraparameter nehmen wir die Matrizen W und Z zu Hilfe. In Hartley and Zisserman, kann man im Appendix auf Seite 541 den genaueren Ursprung dieser beiden Hilfsmatrizen erfahren[2].

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (6.15)$$

Mit dem Ergebnis der *SVD* von E lassen sich nun jeweils zwei Ergebnisse für S und R finden:

$$S_1 = -UZU^T \quad R_1 = UW^TV^T \quad (6.16)$$

$$S_2 = UZU^T \quad R_2 = UWV^T \quad (6.17)$$

Um sicher zu gehen, dass es sich bei R_1 und R_2 auch um Rotationen handelt, kann man folgende Proben durchführen. Zum einen muss $R \cdot R^T = I_{3 \times 3}$ sein. $I_{3 \times 3}$ bezeichnet in diesem Fall eine 3×3 -Einheitsmatrix. Des Weiteren kann man überprüfen, ob die Determinante $\det(R_1) = 1$ ist. Nun müssen wir aus den 3×3 -Matrizen S_1 und S_2 noch das fehlende t rausfiltern. Wir wissen:

$$St = [t]_x \cdot t = t \times t \quad (6.18)$$

Daraus kann man schließen, dass $t =$ dem Nullspace von S_1 und S_2 ist. Der Nullspace beider Matrizen S_1 und S_2 ist der selbe. Die externen Kameraparameter lassen sich wie gesagt nur bis zu einem Skalierungsfaktor genau berechnen. Dass soll heißen, ist t der Nullspace von S_1 und S_2 , so ist auch das Vielfache λt eine gültige Lösung für t . Letzten Endes haben wir für die externen Kameraparameter vier verschiedene Lösungen.

$$PM2 = [UWV^T] + \lambda t \quad \text{oder} \quad [UW^TV^T] + \lambda t \quad (6.19)$$

$$\text{or} \quad [UWV^T] - \lambda t \quad \text{oder} \quad [UW^TV^T] - \lambda t \quad (6.20)$$

Bei den Bestimmungen der externen Kameraparameter kommt es dann zu weiteren Unstimmigkeiten. So sollte zum Beispiel der linke Nullspace von E und S das selbe Ergebnis aufzeigen. Jedoch unterscheiden sich beide Ergebnisse leicht voneinander

`Left S und E = {{-0.902776, 0.430005, 0.00958548}}, {{-0.893112, 0.44975, 0.00864971}}`

Abbildung 6.13: Linker Nullspace von E und S

Zudem sollte aus den Ergebnissen resultieren, dass $E = [t]_x R$ ist und somit das Ergebnis mit unserem E , welches wir dem Modul übergeben übereinstimmt. Auch hier kommt es zu Unstimmigkeiten.

$$E = \begin{pmatrix} -0.0764775 & -0.160882 & 0.0144127 \\ -0.151482 & -0.318837 & 0.0362243 \\ -0.0201221 & -0.0334324 & -0.39536 \end{pmatrix}$$

Abbildung 6.14: Übergebene Essentielle Matrix

$$\begin{pmatrix} -0.0105752 & -0.0130195 & 0.00940217 \\ -0.441974 & -0.891923 & 0.094008 \\ 0.344877 & -0.265938 & -0.900146 \end{pmatrix}$$

Abbildung 6.15: Test $E = [t]_x R$

Und letztendlich kommen für Rotation und Translation nicht die gewünschten Ergebnisse welche im Protokoll notiert waren raus. Außerdem stimmen sie nicht hundertprozentig mit der zur Probe in Matlab durchgeführten Stereokalibrierung überein.

6.5 Szenenrekonstruktion mit Hilfe der Sampson-approximation

6.5.1 andere Ansätze für Triangulationsverfahren

7 3D-Stereokalibrierung und Szenenrekonstruktion mit reellen Daten und Kameras unterschiedlicher Auflösung

7.1 Vorgehen

7.2 Aufbau der Set-Ups

7.3 Was bedeuten andere Auflösungen für die Belichtung auf dem Sensor

7.4 Rekonstruktion der Szene ohne Rektifizierung

7.5 Rekonstruktion der Szenen mit Rektifizierung

8 Aufbauprojekt- Algorithmus zur Punktesortierung in verzeichneten Schachbrettbildern

8.1 Algorithmus zur Punktesortierung in verzeichneten Schachbrettbildern

In diesem Teil der Masterthesis soll am Ende ein Algorithmus entstehen, welcher durch einen bereits bestehenden Algorithmus zur Detektion von Eckpunkten eines Schachbretts, eine Liste an Eckpunkten bekommt und diese auf deren Nachbarschaftsverhältnisse prüft. Die Schachbretter können dabei sowohl Kissen- als auch Tonnenverzeichnungen aufweisen und oder perspektivisch verzerrt sein. Mit den Algorithmus sollen Punkte wissen in welchen Reihen sie sich sowohl in x- als auch y-Richtung befinden. Jeder Punkt bekommt also eine Indexnummer in x-, sowie y-Richtung beziehungsweise in unserem Beispiel wird die y-Koordinate als j bezeichnet und die x-Koordinate als i , zugewiesen. Jeder Punkt bekommt mit Hilfe von den Mathematica eigenen *Associations* einen *Key* mit *NeighbourJ* und *NeighbourI* zugeteilt. Mit Hilfe dieser *Keys* kann dann später bei einem Stereobildpaar zum Beispiel die Korrespondierenden Eckpunkte der Schachbretter rausgesucht werden, was vielleicht genauere Ergebnisse liefert also die Suche von Hand. Des weiteren kann dieser Algorithmus in späteren Projekten vielleicht bei der Rausrechnung von Verzeichnungen hilfreich sein.

8.1.1 Vorläufiges Klassendiagramm

Module	Parameter	Lokale Variablen	Funktion
FindMinMax	Pointlist	Imin, imax, jmin, jmax, iSplits, jSplits, iDistance, jDistance	<ul style="list-style-type: none"> Die minimas und maximas der i und j-Werte der Koordinaten werden gesucht, um den „Rahmen“ des Gitters um das Schachbrett festzulegen In den ConstantArrays JSplits und ISplits werden die Zellen des Gitters gespeichert. Diese werden über die Distanz der jeweiligen Minimalwerte und Maximalwerte geteilt durch die gewünschte Anzahl an Zellen geteilt.
SortPointList	iSplits, jSplits, Pointlist	pj,pj	<ul style="list-style-type: none"> Die Eckpunkte werden zunächst der Größe nach nach ihren i-Werten Sortiert. Die sortierte Liste wird dann durchgezählt, so dass jeder Punkt seinen Indexwert in I-Richtung bekommt Danach werden die Eckpunkte der Größe nach nach ihren J-Werten sortiert und bekommen hier ebenfalls einen Index zugeordnet (Diese Sortierung ist nach jetzigem Stand des Algorithmus vllt nicht mehr zwingend notwendig)
GoThroughConvex Hulls	iSplits, jSplits, pj	ConvexHull	<ul style="list-style-type: none"> Nun wird herausgefiltert, welcher Punkt in welche Zelle des erstellten Gitters gehört, somit wird eine grobe Vorsortierung der Punkte für den weiteren Verlauf vorgenommen. In einer For-Schleife welche alle iSplits durchzählt wird die Funktion FindPointsInConvexHull bei jedem Durchgang aufgerufen welche eine Liste mit Associations in die Liste ConvexHull hinzufügt. Der Funktion werden die momentanen iSplits der Durchzählung übergeben und alle Jsplits. Des Weiteren wird die nach J sortierte Punktliste übergeben
FindPointsInConvexHull	iSplits[[1,ii]], iSplits[[1,ii+1]], jSplits, pj	ConvexHullCell={}, ConvexHullList, ConvexHullCellKeys = < >	<ul style="list-style-type: none"> Eine Liste namens ConvexHullCell und eine Association nach dem ConvexHullKeys wird angelegt Zwei For-Schleifen werden gestartet. Die erste läuft durch alle Jsplits, die zweite geht alle Punkte von pj durch. Innerhalb der For-Schleife wird dann überprüft, welche Punkte aus pj sich innerhalb der übergebenen iSplits und den dazugehörigen jSplits befinden. Die Koordinaten, die Indizes und die Zellenbezeichnung werden dann in Keys in die Association ConvexHullCellKeys gespeichert und er Liste ConvexHullCell angehängt. Diese Liste wird dann und die Liste ConvexHull angehängt Wiederholung des Vorganges mit neuen iSplits.

Abbildung 8.1: Klassendiagramm

FindStartVectors	ConvexHull	PointCloud={}, StartPointCloudKeys=< >, VecI,VecJ,countI,countJ, Start, nextI,nextJ,	<ul style="list-style-type: none"> Die Punkte der Zellen ($i = 1, j = \text{All}$) und ($i = \text{all}, j = 1$) werden in eine neue Liste namens StartPointCloud gespeichert. Die Liste wird zweimal durchlaufen <ul style="list-style-type: none"> Alle Punkte welche sich in den Zellen $j = 1$ und $i = \text{all}$ aufhalten. Aus ihnen wird der geringste i-Wert ermittelt (VecI) Alle Punkte welche sich in den Zellen $j = \text{all}$ und $i = 1$ aufhalten. Aus ihnen wird der geringste j-Wert ermittelt (VecJ) Die Punkte mit den geringsten Werten werden in VecI und VecJ gespeichert. Jetzt wird die Liste nochmals zweimal durchgegangen. <ul style="list-style-type: none"> Alle Punkte welche sich in den Zellen $j = 1$ und $i = \text{all}$ aufhalten werden durchgegangen. Aus ihnen wird derjenige Wert ermittelt, welcher einen Wert für j besitzt der kleiner ist als der momentane j-Wert von VecI und dessen i-Wert kleiner ist als der i-Wert von VecI plus einem Offset. Dieser Wert ist das neue VecI Alle Punkte welche sich in den Zellen $j = \text{all}$ und $i = 1$ aufhalten. Aus ihnen wird derjenige Wert ermittelt, welcher einen Wert für i besitzt der kleiner ist als der momentane i-Wert von VecJ und dessen j-Wert kleiner ist als der j-Wert von VecI plus einem Offset. Dieser Wert ist das neue VecJ VecI und VecJ ergeben den gleichen Punkt und somit ist der Startwert gesetzt. Nun sollen die ersten Punkte in i- und j-Richtung vom Startpunkt aus gefunden werden. Es wird ein nexti und ein nextj definiert, dessen Koordinaten sehr groß anfangen Es wird wieder die StartPointListe zweimal durchlaufen <ul style="list-style-type: none"> Es werden Punkte gesucht, welche sich in der selben Zelle i wie der Startpunkt befinden und auch die Zellen +1 und -1 drum herum. Sollte es ein Punkt geben, der kleiner ist als das momentane nexti und größer als der Startpunkt, jedoch nicht gleich dem Startpunkt. So nimmt nexti dessen Wert an. Danach muss geprüft werden, ob das potentielle nexti auch wirklich das richtige nexti ist. Hierzu wird eine neue For-Schleife gestartet, welche wieder die StartPointCloud durchgeht und überprüft ob es einen Punkt gibt dessen j-Koordinatenabstand zum Startpunkt kleiner ist also der j-Koordinatenabstand des momentanen nexti zum Startpunkt und ob dessen i-Koordinatenabstand zum Startpunkt kleiner ist als der momentane i-Koordinatenabstand von nexti zum Startpunkt.
------------------	------------	---	---

Abbildung 8.2: Klassendiagramm

			<ul style="list-style-type: none"> Ist dies der Fall so wird dieser Punkt zum neuen nexti. Mit dem potentiellen nextj wird ebenso verfahren.
CreatePossiblePoint - ListsIAndJ	nextI, nextJ, Start, ConvexHull	IList= {}, JList= {}, IDir, JDir, distance, cache, PotNextI, PotNextJ	<ul style="list-style-type: none"> IDir und JDir sind die Richtungsvektoren vom Startpunkt aus in beide Kantenrichtungen des Schachbretts. Danach werden die ersten beiden Spalten in I- und J-Richtung jeweils durchlaufen, und in IList und JList gespeichert. Diese Listen enthalten weitere potentielle Punkte entlang der gesuchten Kante. Die Kanten können natürlich durch die perspektivische Verzerrung mancher Bilder auch noch weiter in die Zellen hineinragen. Hierum kümmert sich dann im späteren Algorithmus die SaftyJList[] und SaftyIList[] Funktionen
FindNeighbours	IList, JList ,Start, nextI, nextJ, ConvexHull	SortedPointsKeys = <>, Sortedpoints = {}, proportionJ, proportionI, Jtemp, itemp, PotNextJDir, distanceNextPotPointJ, PotNextIDir, distanceNextPotPointI, NeighbourNumberJ, NeighbournumberI, distanceJ, distanceI, NextJDir, NextIDir, StartPropJForFirstCompleteGridJ	<ul style="list-style-type: none"> StartPoint und NextPointI und NextPointJ werden die Keys NeighbourI und NeighbourJ gegeben mit startPoint(NeighbourJ → 1, NeighbourI → 1), NextPointI(NeighbourJ → 1, NeighbourI → 2) und NextPointJ(NeighbourJ → 2, NeighbourI → 1). Diese drei bereits bekannten Punkte werden dann auch in eine angelegte CheckPointList gespeichert, diese wird für das spätere Prüfen von weiteren Punkten benötigt. Nun wird zunächst in einer For-Schleife die Punkte von startPoint und NextPointJ aus gesucht. <ul style="list-style-type: none"> Anmerkung: Für die Punkte in I-Richtung des Schachbretts wird das selbe Verfahren angewandt. Benötigt wird die Distanz zwischen dem momentanen startPointJ, welcher nach jedem Durchlauf der Schleife den Wert des momentanen NextPointJ bekommt und einem momentanen NextPointJ, welcher nach jedem Durchlauf der Schleife den Wert des gerade neu gefundenen nächsten Punktes bekommt. Die Schleife selbst durchläuft alle Punkte, welche in der für die Richtung entsprechenden Richtung Liste sind. In diesem Fall die JList Es wird außerdem bei der Suche den nächsten Punktes in j-Richtung eine Distanz namens proportion berechnet, welcher die Distanz i zwischen startPoint und Nextpoint beinhaltet. Innerhalb der durchlaufenden Liste wird derjenige Punkt gesucht welcher zum NextPointJ den geringsten Abstand in J-Richtung hat und dessen Abstand in I-Richtung <= der i-Koordinate des NextPointJ + proportion+noch einen Puffer ist und >= der i-Koordinate des NextPointJ – proportion+noch einen Puffer.

Abbildung 8.3: Klassendiagramm

			<ul style="list-style-type: none"> Ist der nächste Punkt gefunden, so wird dieser der SortedPointsList und der der CheckPointsList übergeben mit den passenden NeighbourI und NeighbourJ associationKey. Des Weiteren bekommt für den nächsten Schleifendurchlauf startPointJ die Werte von NextPointJ und NextPointJ' in wird der neu gefundenen Punkt aus der JList gespeichert. Im Anschluss werden noch in AppendTo[SortedPoints, SaftyListJ[Start, CheckPointJ, proportionY, CheckCellForJ, ConvexHull, distanceJ]], AppendTo[SortedPoints, CompleteJGrid[nextI, ConvexHull, StartDistanceJ, StartProportionJ, Start_Jp, al]] Weitere Punkte zur SortedList in J-Richtung hinzugefügt, bei ersterem nur in bestimmten Fällen. Mehr zu den Funktionen folgt. Nicht zu vergessen: selbiges wie oben wird auch mit den Punkten in I-Richtung vollzogen, bis auf die CompleteGrid Funktion
SaftyList	Start, CheckLastPointJ , proportionJ, LastJPointsCell, ConvexHull, NextJDir	SaftyList = {}, SaftyKeys =<>, SaftyKeysList = {}, propJ, lastDir, lastdistanceJ	<ul style="list-style-type: none"> Der Funktion werden die Parameter CheckLastPointJ und LastJPointCell mitgegeben. Diese stammen aus der Funktion FindNeighbours und es handelt sich um den letzten Punkt der innerhalb der JListe ermittelt wurde und dessen i-Zelle in welcher sich dieser befindet. Da die I- bzw die JListe in jede Richtung nur die Punkte der ersten beiden Zellen beinhaltet, kann es bei einem rotierten Schachbrett sein, dass sich noch weitere Punkte in Zellen weiter oben/unten befinden Die Funktion SaftyList, erstellt eine Liste aus möglichen weiteren Punkten, indem sie die in diesem Falle I-Zelle des letzten Punktes nimmt und diese so wie die unter und oberhalb dieser Zelle und alle deren J-Zellen aufwärts auf einen möglichen nächsten Punkt untersucht. → Dies geschieht nach dem selben Verfahren wie in FindNeighbours. Sollte es noch einen geben wird dieser ebenfalls der CheckPointList und der SortedPointsList zugewiesen, ansonsten passiert nichts.
CompleteJGrid	StartPointI, ConvexHull, StartDistanceJ, proportionJ, Start,	PossiblePointsList = {}, SortedPointsKeys = <>, SaftyPossiblePointsListJ = {}, propJ, StartPointForJGrid, distanceJ,	<ul style="list-style-type: none"> Nachdem die äußersten Punkte der linken und unteren Kante des Schachbretts gefunden wurden, muss nun das restliche Grid des Schachbretts detektiert und mit den richtigen NeighbourI und NeighbourJ Werten versehen werden. Jeder Punkt der in I-Richtung als „Rahmenpunkt“ detektiert wurde, wird einmal als Startpunkt gesetzt, von ihm aus wird dann in einem sehr ähnlichen Verfahren wie schon zuvor der nächste Punkt in J-Richtung gesucht und wenn nötig tritt auch hier

Abbildung 8.4: Klassendiagramm

NeighbourNumberJ, aI	NextNeighbourNumberJ, distanceNextPotGridPointJ, tempJ, NextPointJDir, NextJDir, CheckPointJ, CheckCellForJ	nochmal die SaftyList Funktion in kraft um auch wirklich alle Punkte jeder Reihe ausfindig zu machen
----------------------	--	--

Abbildung 8.5: Klassendiagramm

8.1.2 Beispiele

In den folgenden Beispielen sieht man jeweils das Originalbild und ein Bild welches die durch den Algorithmus sortierten Punkte farbig ausgibt. Die grünen eingefärbten Punkte sind in den Bildern des Algorithmus die Nachbarn, welche sich in i-Richtung an der dritten Stelle befinden. Natürlich können auch andere Reihen oder auch einzelne Punkte abgefragt werden.

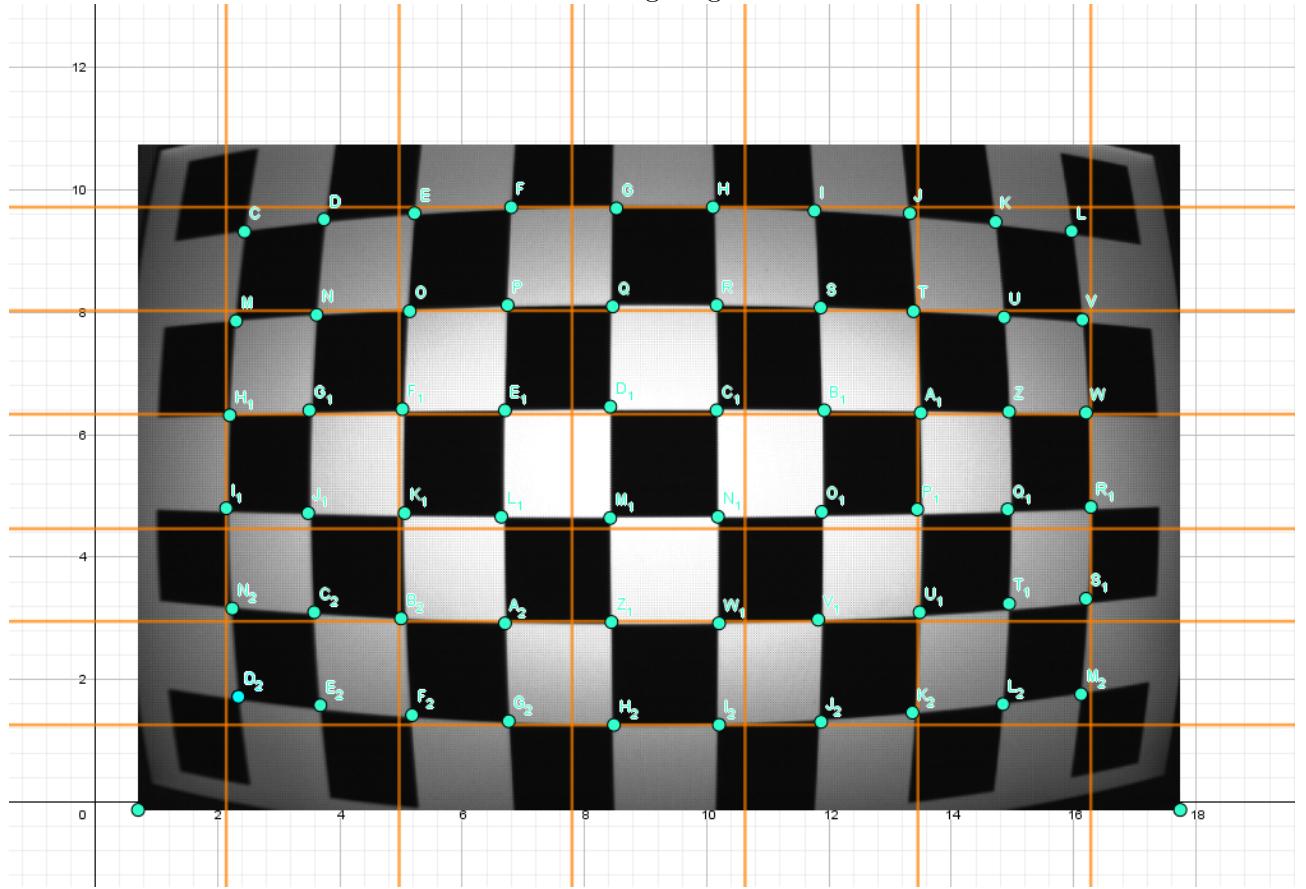


Abbildung 8.6: Bild eines Tonnenförmig verzeichneten Schachbretts

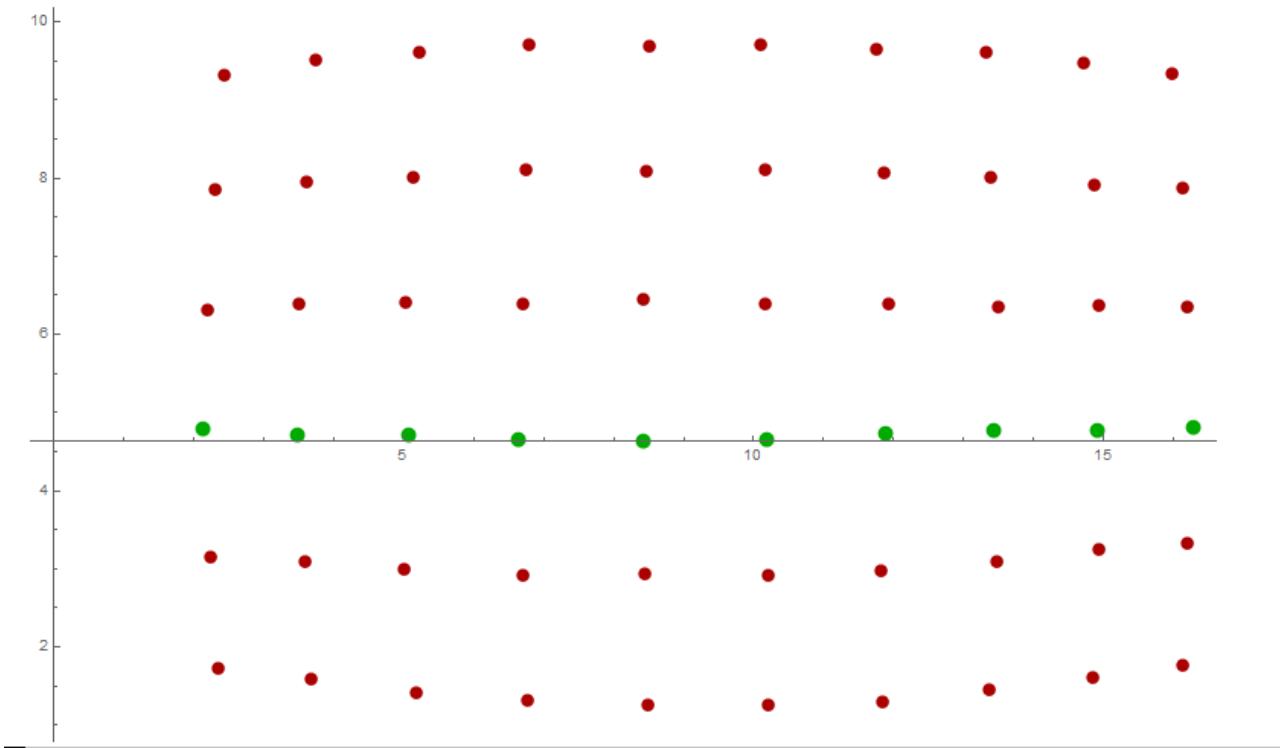


Abbildung 8.7: Algorithmisch detektierte Linie der dritten i-Reihe

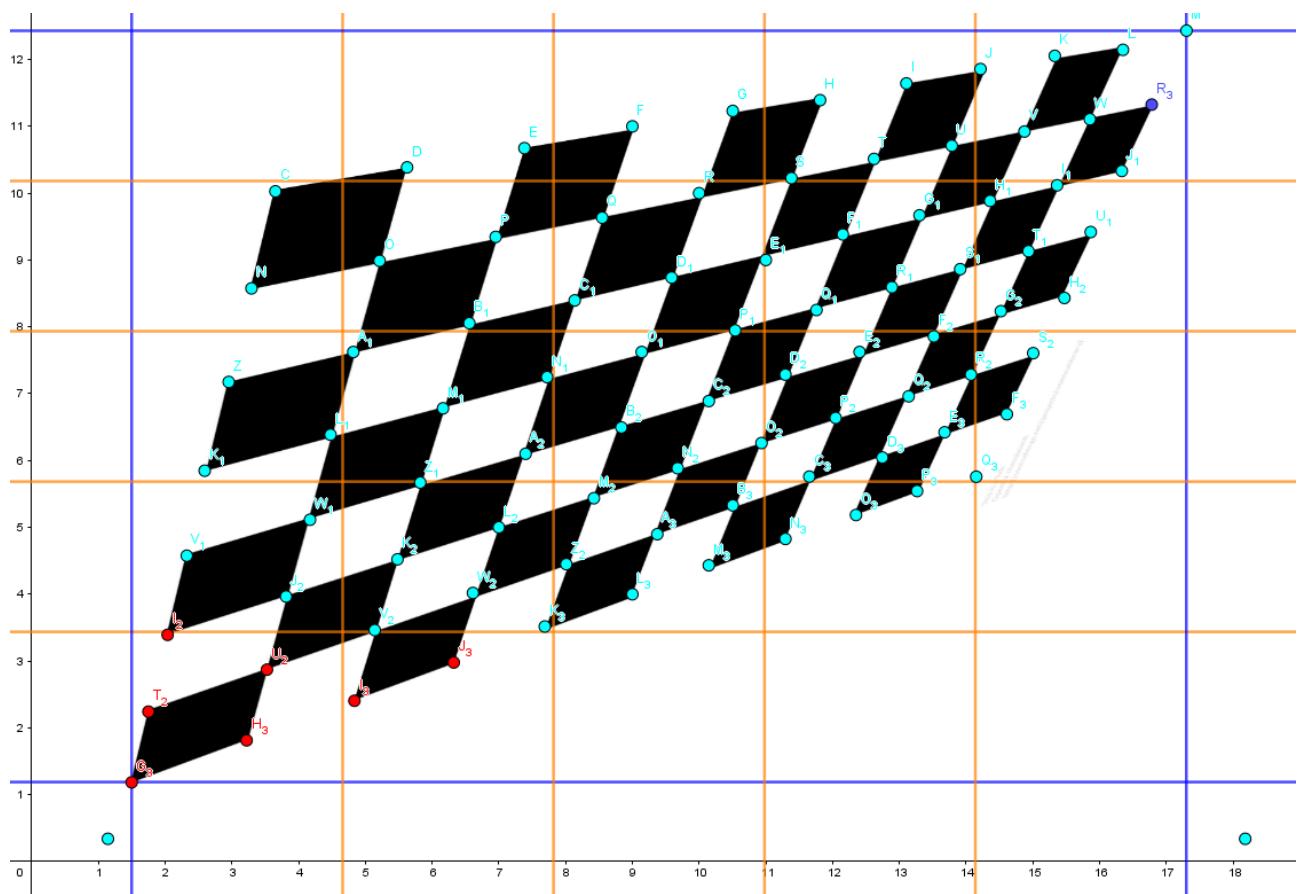


Abbildung 8.8: Bild eines perspektivisch verzerrtem Schachbretts

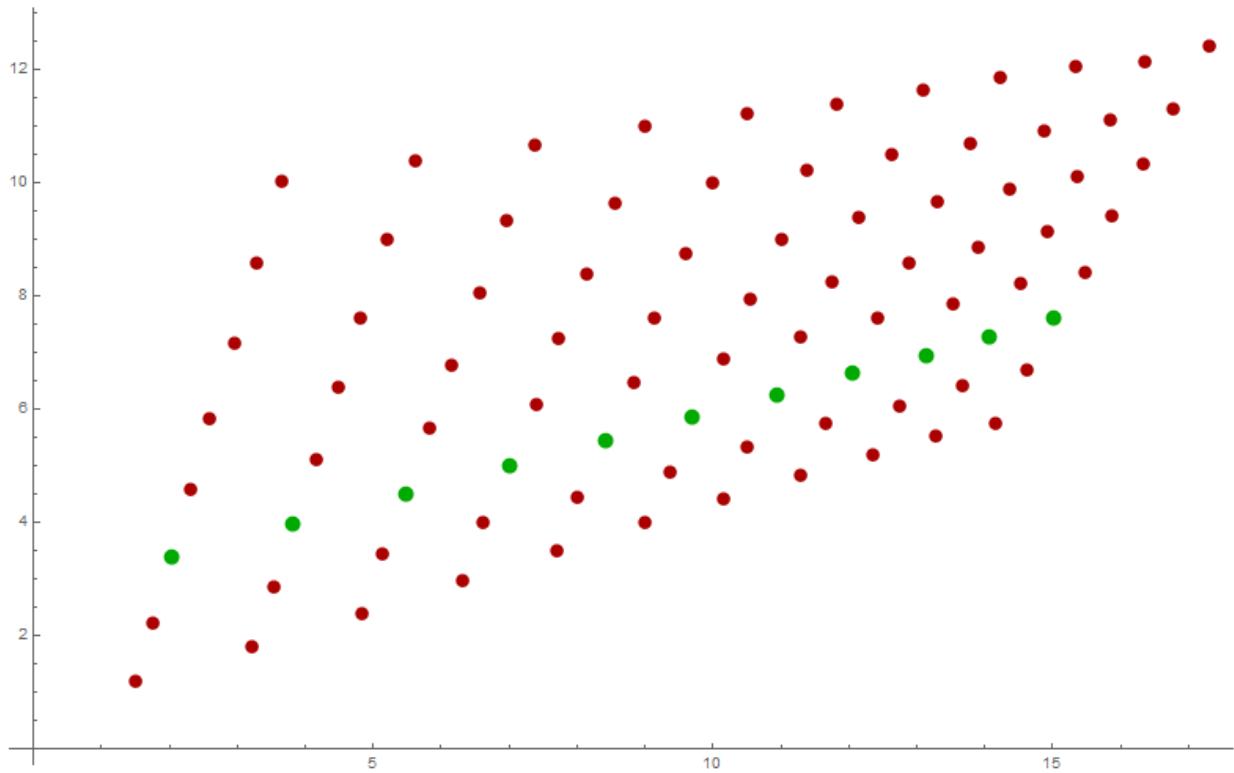


Abbildung 8.9: Algorithmisch detektierte Linie der dritten i-Reihe

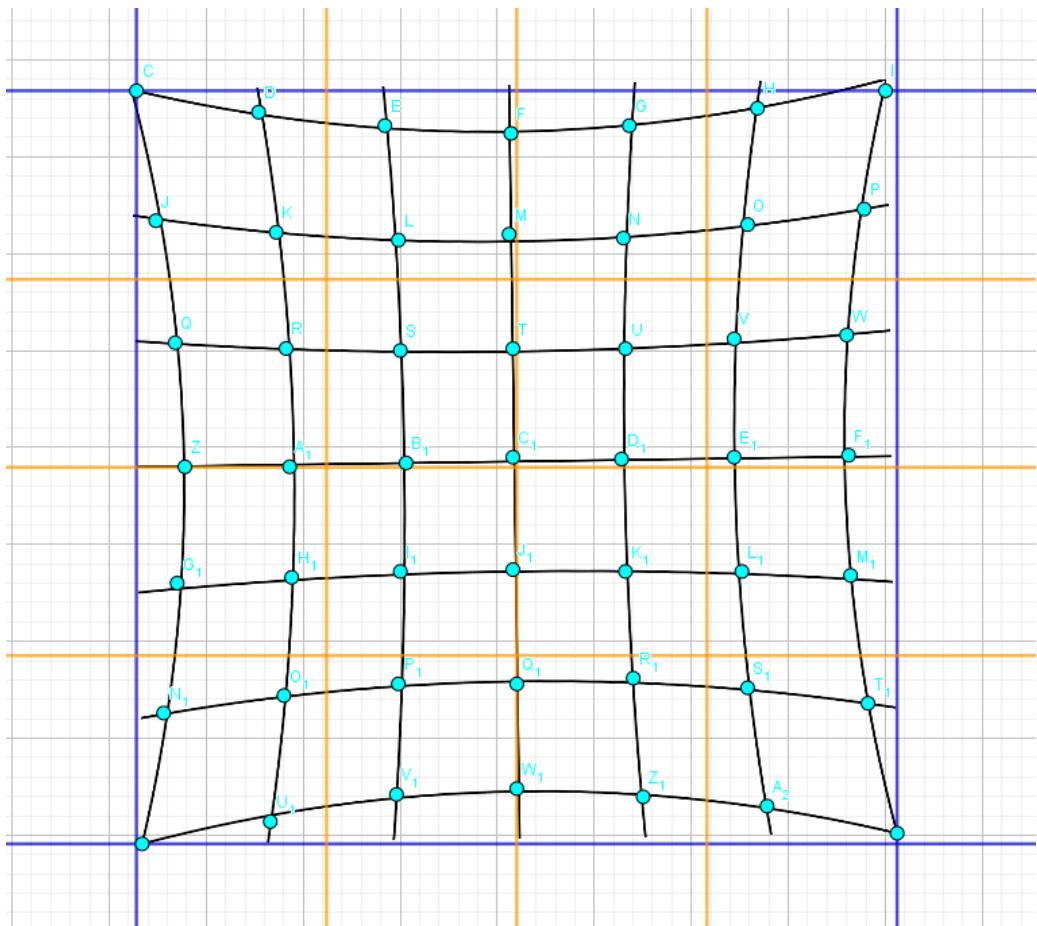


Abbildung 8.10: Bild eines Kissenförmig verzeichnetem Schachbretts

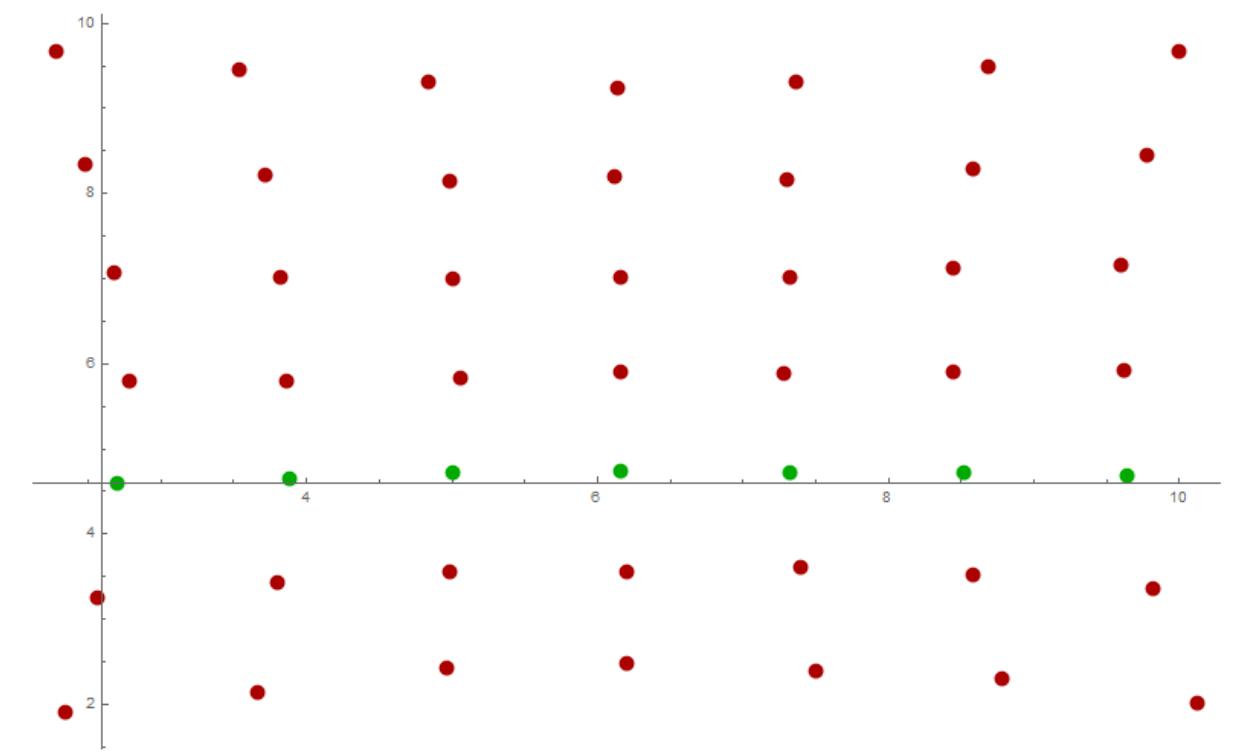


Abbildung 8.11: Algorithmisch detektierte Linie der dritten i-Reihe

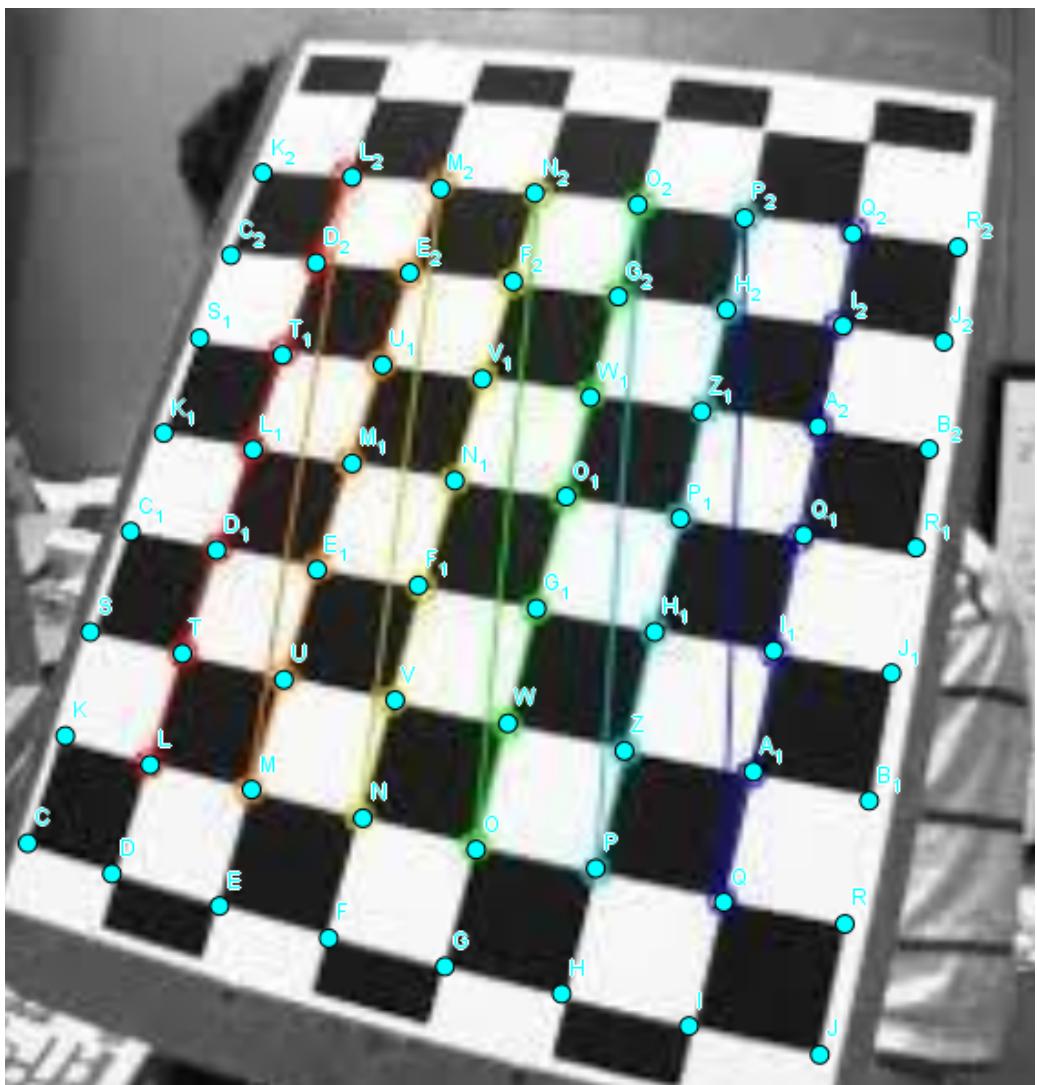


Abbildung 8.12: Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts

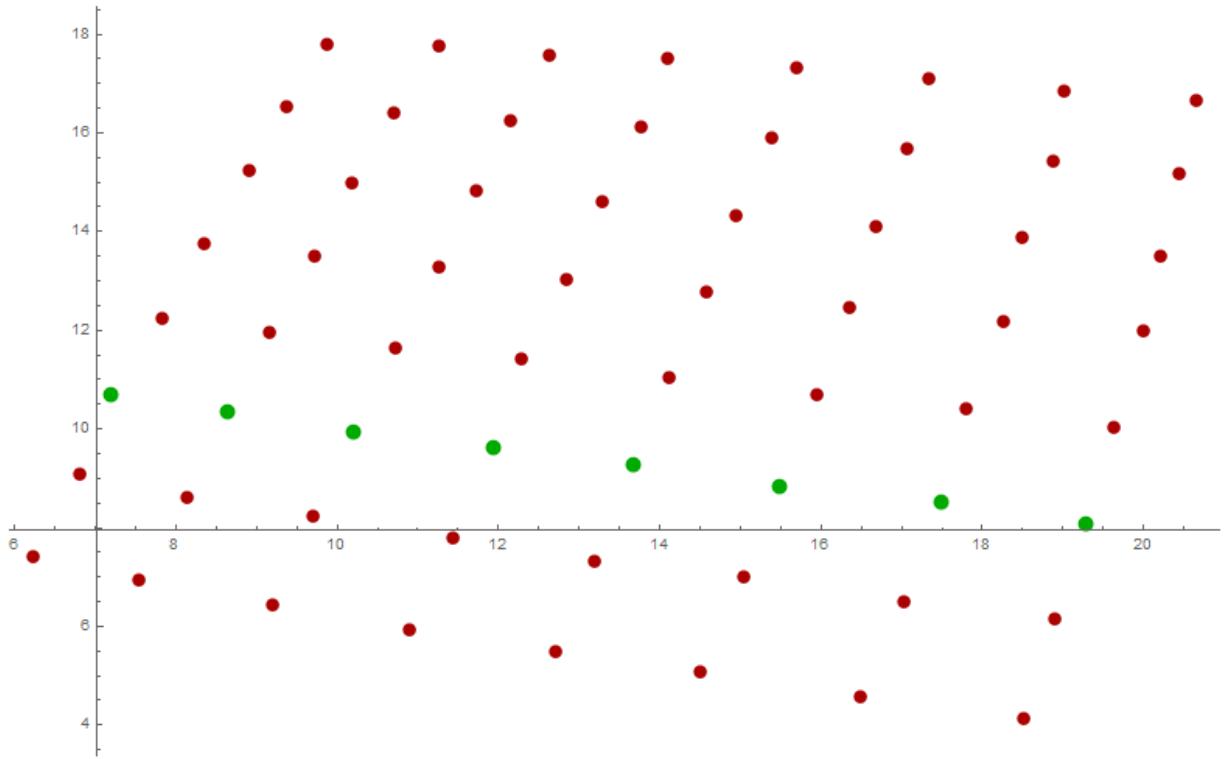


Abbildung 8.13: Algorithmisch detektierte Linie der dritten i-Reihe

8.1.3 Weiteres Vorgehen/ Was fehlt noch

Was nun noch in den Algorithmus eingebaut werden muss ist eine *Saftylist*-Funktion welche auch in i-Richtung wirkt. Des Weiteren muss davon ausgeganegn werden, dass der voranlaufende Algorithmus zu Auffindung der Punkte, nicht immer alle Punkte detektiert. Es muss also noch eine Funktion geschrieben werden, welche zunächst prüft, ob es noch einen nächsten Punkt gibt, wenn dies nicht der Fall ist wird ein vorläufiger Punkt an die Stelle gesetzt, an welcher der Punkt sich befinden sollte und von diesem aus weiter geprüft ob es noch einen gibt. Sollte dies der Fall sein, so wird ab dem nächsten real gefundenen Punkt weiter gesucht und der selbst erzeugte Punkt wird als "fake" Punkt eingesetzt. Sollte nach dem erstellten Punkt weiterhin keiner folgen, kann man überlegen den test nochmals vom selbst erstellten Punkt aus auszuführen, oder es dabei zu belassen.

(VERBESSERN) Des Weiteren muss nochmal genau ermittelt werden warum es bei einem sehr extremen Fall wie in der folgenden Abbildung noch nicht ganz funktioniert. Die Vermutung ist, dass hier noch mit SaftyFunktionen in i-Richtungen hantiert werden muss.

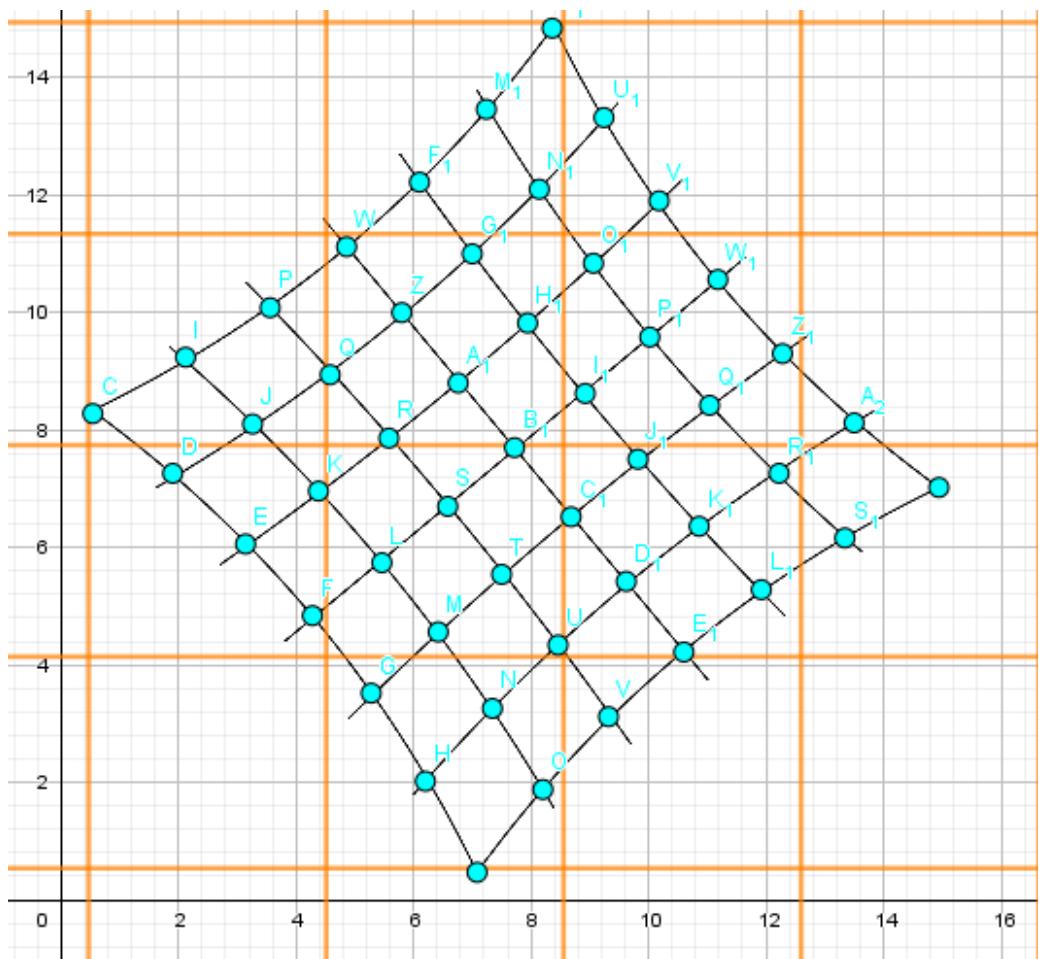


Abbildung 8.14: Kissenverzeichnung start rotiert

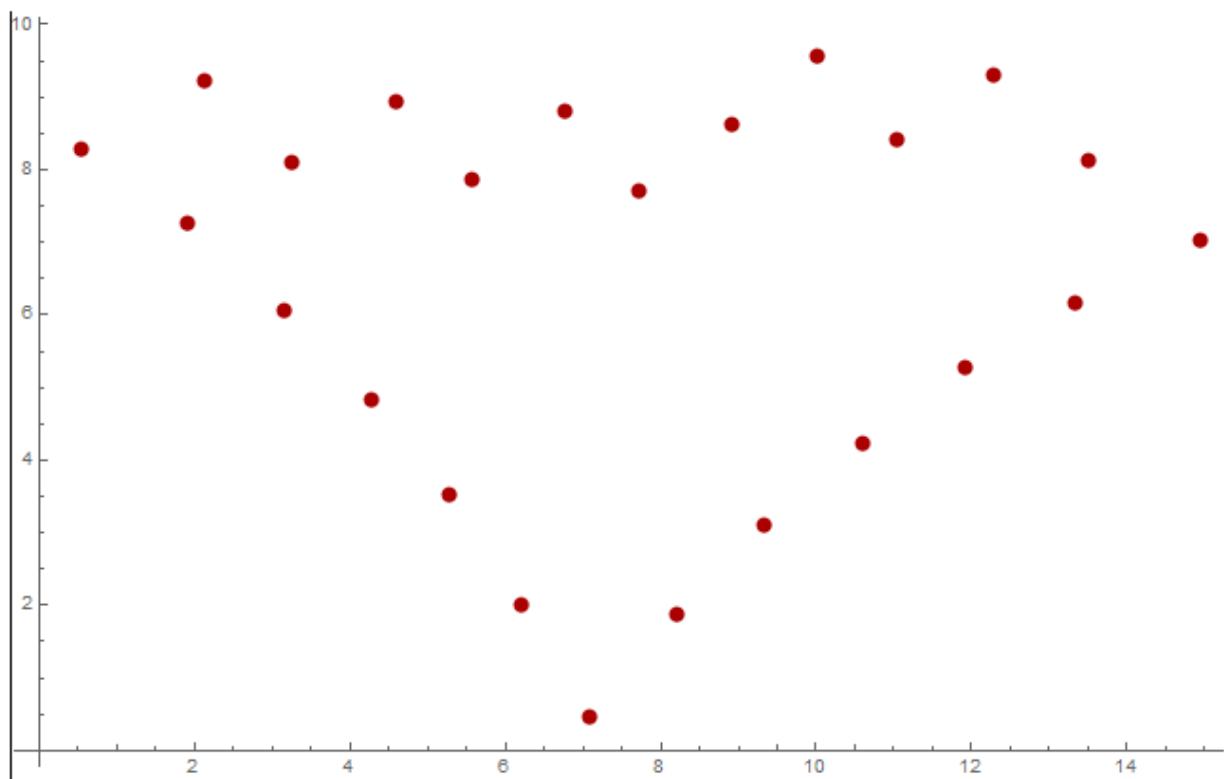


Abbildung 8.15: Kissenverzeichnung start rotiert bisher gefundene Punkte

Als letztes werden die Pufferwerte momentan noch hart gecode, diese sollen in Zukunft aus den

Zahlenbereichen in welchen sich die Punkte aufhalten ausgerechnet werden.

9 C3 - ExampleHeader2

10 C3 - ExampleHeader3

11 Fazit - Conclusion

12 Nächste Schritte - next steps

13 Protocol - 10.11.2015

14 Abkürzungsverzeichnis - List of Abbreviations

Abbildungsverzeichnis

2.1	Weltkoordinatensystem $K_w = (O, \hat{e}_1, \hat{e}_2, \hat{e}_3)$ und Kamerakoordinatensystem $K_c = (O_c, \hat{c}_1, \hat{c}_2, \hat{c}_3)$	6
2.2	In blau ist die Bildebene dargestellt auf ihr befinden sich die Punkte Q und P . Z liegt nicht auf der Ebene, Das Projectionszentrum liegt hinter der Bildebene und somit auch hinter dem Sensor. n ist die Normale der Bildebene	13
3.1	Weltkoordinatensystem $K = (O, \hat{e}_1, \hat{e}_2, \hat{e}_3)$ und Kamerakoordinatensystem $K_c = (O_c, \hat{c}_1, \hat{c}_2, \hat{c}_3)$.	19
3.2	in blau ist die Abbildung des Quaders von Kamera eins und in rot die Abbildung des selben Quaders in Kamera zwei (BILD NOCHMAL ÜBERARBEITEN)	23
3.3	Objekt im Raum	27
3.4	Drehung um das Projektionszentrum	27
3.5	Drehung um einen Drehpunkt. In diesem Beispiel wurde der rote Punkt als Drehpunkt verwendet	28
4.1	Top-Down-Ansicht des entstandenen Aufbaus der Kameras	31
4.2	3D-Ansicht des entstandenen Aufbaus der Kameras	31
4.3	Grün zeigt den Quader aus sicht von Kamera 1. Das größere Quadrat sind die vorderen Punkte a, b, c, d , das kleinere Quadrat sind die hinteren Punkte a', b', c', d' . Rot zeigt denselben Quader aus Sicht vom Kamera 2, wenn diese in $+e_1$ -Richtung verschoben und um 45° um e_2 gedreht wurde	33
4.4	Die blauen Geraden zeigen die jeweiligen Epipolargeraden. Die vom roten Quader schneiden sich bei -1 im Epipol. Die Epipolargeraden vom grünen Quader schneiden sich im Epipol im Unendlichen	36
6.1	Rekonstruktion des Gier-Winkels der Kameras zueinander	45
6.2	Kamera 6D	46
6.3	Kamera 60D	46
6.4	Stereosetup	47
6.5	Szenenbild der primären Kamera 6D	47
6.6	Szenenbild der sekundären 60D	48
6.7	Beispiel einer resultierenden Koeffizientenmatrix	49
6.8	Resultierende Epipolarlinien des Bildes der Canon 6D	50
6.9	Resultierende Epipolarlinien des Bildes der Canon 60D	51
6.10	Resultierende Epipolarlinien des Bildes der Canon 6D nach singularity-constraint	52
6.11	Resultierende Epipolarlinien des Bildes der Canon 60D nach singularity-constraint	53
6.12	Ergebnis der Prüfung der Punktekorrespondenzen mit der essentiellen Matrix	54
6.13	Linker Nullspace von E und S	55
6.14	Übergebene Essentielle Matrix	56
6.15	Test $E = [t]_x R$	56
8.1	Klassendiagramm	59
8.2	Klassendiagramm	59
8.3	Klassendiagramm	60
8.4	Klassendiagramm	60
8.5	Klassendiagramm	60
8.6	Bild eines Tonnenförmig verzeichneten Schachbretts	61
8.7	Algorithmisch detektierte Linie der dritten i-Reihe	62
8.8	Bild eines perspektivisch verzerrtem Schachbretts	62
8.9	Algorithmisch detektierte Linie der dritten i-Reihe	63

8.10 Bild eines Kissenförmig verzeichnetem Schachbretts	63
8.11 Algorithmisch detektierte Linie der dritten i-Reihe	64
8.12 Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts .	65
8.13 Algorithmisch detektierte Linie der dritten i-Reihe	66
8.14 Kissenverzeichnung start rotiert	67
8.15 Kissenverzeichnung start rotiert bisher gefundene Punkte	67

Tabellenverzeichnis

Literaturverzeichnis

- [1] Tomas Pajdla. Elements of geometry for computer vision. 2013.
- [2] Richard Hartley and Andrew Zisserman. Multiple view geometry in computer vision. Second Edition, 2004.
- [3] Dipl.-Ing. Martin Roser. Modellbasierte und positionsgenaue erkennung von regentropfen in bildfolgen zur verbessерung von viedeobasierten fahrerassistenzfunktionen. KIT Scientific Publishing, 1986, 1994 Springer Basel AG.
- [4] Jeanne Peiffer and Amy Dahan-Dalmedico. Wege und irrwege - eine geschichte der mathematik. Editions du Seuil, Springer Basel AG , aus dem französischen von Klaus Volkert, 1986, 1994 Springer Basel AG.
- [5] Norbert Köckler Hans Rudolf Schwarz. Numerische mathematik. 8. Auflage, 2011, Springer Verlag.
- [6] Daniel Scholz. Numerik interaktiv, grundlagen verstehen, modelle erforschen und verfahren anwenden mit taramath. 2016, Sringer Verlag.
- [7] Richard I. Hartley GE-Corporate Research and Development. In defence of the 8-point algorithm. Schenectady, NY, 12309,.