
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	5
2 Basis Transformationen	6
2.1 Transformations eines synthetischen Welt- zu einem Kamerakoordinatensystem mit Hilfe von Basistransformationen (Die Transformation von Koordinaten schließt die Transformation der Basis mit ein)	6
2.2 Übersicht über die benötigten Koordinatensysteme und Transformationen für die Stereobildanalyse	10
2.3 Aufbau der Koordinatensysteme	10
3 Homographien in der Ebene	17
3.1 Homographie zwischen der Abbildungen eines Quadrates einer definierten Ausgangskamera und einer um ihr Projektionszentrum rotierten Kamera	20
3.2 Abbildungsunterschiede von Rotationen um ein Projektionszentrum und Rotation um einen beliebigen Drehpunkt von Punkten in der Ebene	27
3.3 Epipolargeometrie als Grundlage der Stereokalibrierung und Szenenrekonstruktion . .	35
3.4 Geometrische Erläuterung der Fundamentalmatrix und der Essentiellen Matrix	37
4 Minimalbeispiel 3D-Stereokalibrierung und Szenenrekonstruktion bei Kameras gleicher Auflösung	40
4.1 Vorgehen: Projektion eines Quaders in zwei verschieden transformierte Kameras . . .	41
4.2 Berechnung der Projektionsmatrizen	41
4.3 Transformation der Weltpunkte in Koordinaten der Koordinatensysteme von beiden Kameras	42
4.4 Umrechnung von Bildebenenkoordinaten in Sensorkoordinaten	44
4.5 Ermitteln der Fundamentalmatrix mit Hilfe des 8-Point-Algorithms	44
4.6 Berechnen der Epipole und Epipolgeraden mit der Fundamentalmatrix	45
4.6.1 Konstruktion der Epipole und der Epipolgeraden auf Grundlage der Epipolargeometrie	46
4.7 Ermitteln der Essentiellen Matrix über die Fundamentalmatrix	47
4.8 Ermitteln der externen Kameraparameter mit Hilfe der Essentiellen Matrix	48
4.9 Szenenrekonstruktion durch Triangulation	50
4.10 Rektifizierung	54
4.10.1 Projektive Transformation	58
4.10.2 Ähnlichkeitstransformation	63
4.10.3 Scherungstransformation	65
5 Minimalbeispiel 3D-Stereokalibrierung und Szenenrekonstruktion bei Kameras unterschiedlicher Auflösung	67
5.1 Abbildungsunterschiede bei unterschiedlichen Kameraauflösungen	67
5.2 Auswirkung unterschiedlicher Auflösungen auf die Epipolargeometrie	69
5.3 Minimalbeispiel mit unterschiedlichen Auflösungen	70
6 3D-Stereokalibrierung und Szenenrekonstruktion mit reellen Daten und Kameras gleicher Auflösung	75
6.1 Vorgehen	75
6.2 Aufbau der Set-Ups	79

6.3	Unterschiede im Algorithmus im Vergleich zum Minimalbeispiel	81
6.3.1	normalisierung der eingehenden Daten und Berechnung der Fundamentalmatrix über den normalized 8-Point-Algorithm	81
6.3.2	Ermitteln der Essentiellen Matrix und Einführung des singularity-constraints .	86
6.4	Rekonstruktion der exterenen Kameraparameter	87
6.5	Szenenrekonstruktion mit Hilfe der Sampson-approximation	89
6.5.1	andere Ansätze für Triangulationsverfahren	89
7	3D-Stereokalibrierung und Szenenrekonstruktion mit reellen Daten und Kameras unterschiedlicher Auflösung	90
7.1	Vorgehen	90
7.2	Aufbau der Set-Ups	90
7.3	Was bedeuten andere Auflösungen für die Belichtung auf dem Sensor	90
7.4	Rekonstruktion der Szene ohne Rektifizierung	90
7.5	Rekonstruktion der Szenen mit Rektifizierung	90
8	Aufbauprojekt- Algorithmus zur Punktesortierung in verzeichneten Schachbrettbildern	91
8.1	Algorithmus zur Punktesortierung in verzeichneten Schachbrettbildern	91
8.1.1	Vorläufiges Klassendiagramm	92
8.1.2	Beispiele	94
8.1.3	Weiteres Vorgehen/ Was fehlt noch	99
9	C3 - ExampleHeader2	102
10	C3 - ExampleHeader3	103
11	Fazit - Conclusion	104
12	Nächste Schritte - next steps	105
13	Protocol - 10.11.2015	106
14	Abkürzungsverzeichnis - List of Abbreviations	107

Over the last decades computer vision scientist have taken a new approach to vision. They build different computational models of what shoud be computed, what can really be computed, and how these computations can be realized by computer programs, and they use computers to test their models are correct. The result is a better understandung of vision from a different point of view, and at the same time some working artificial vision systems are built that can be used in idustry, medicine, etc. The knowledge obtained on neirophysiology and psychophysics have given hints to and influenced computer vision scientists, helping find solutions to the design of specific algorithms and implementation of vision systems. On the other Hand coputational vision has also given nerophysologists and psychophysicsist a mathematical framework for modeling vision processes.[1]

1 Einleitung, Motivation - motivation

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

Generelle kleine einleitung über kamerakalibrierung und was das genau ist (NICHT VIEL!!)
Kameramodellbeschreibung geometrisch in [2]

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 Kamera 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 einem Weltkoordinatensystem (O, δ) mit $\delta = (d_1, d_2, d_3)$ in ein Kamerakoordinatensystem (C, β) mit $\beta = (b_1, b_2, b_3)$ aufgezeigt. Noch anzumerken ist, dass in dieser Arbeit grundlegend von orthogonalen Koordinatensystemen ausgegangen wird, andernfalls wird explizit darauf hingewiesen.

2.1 Transformations eines synthetischen Welt- zu einem Kamerakoordinatensystem mit Hilfe von Basistransformationen (Die Transformation von Koordinaten schließt die Transformation der Basis mit ein)

Anhand eines Beispiels wird die Transformation der Koordinaten noch genauer veranschaulicht. Es soll kompakt in einer Symbolischen Schreibweise Punkte aus einem Weltkoordinatensystem (O, δ) mit $\delta = (d_1, d_2, d_3)$ in ein Kamerakoordinatensystem (C, β) mit $\beta = (b_1, b_2, b_3)$ überführt werden, welches eine Verschiebung und Rotation zum Weltkoordinatesystem aufweist. Beispielhaft wird dies in Abbildung 2.1. aufgezeigt.

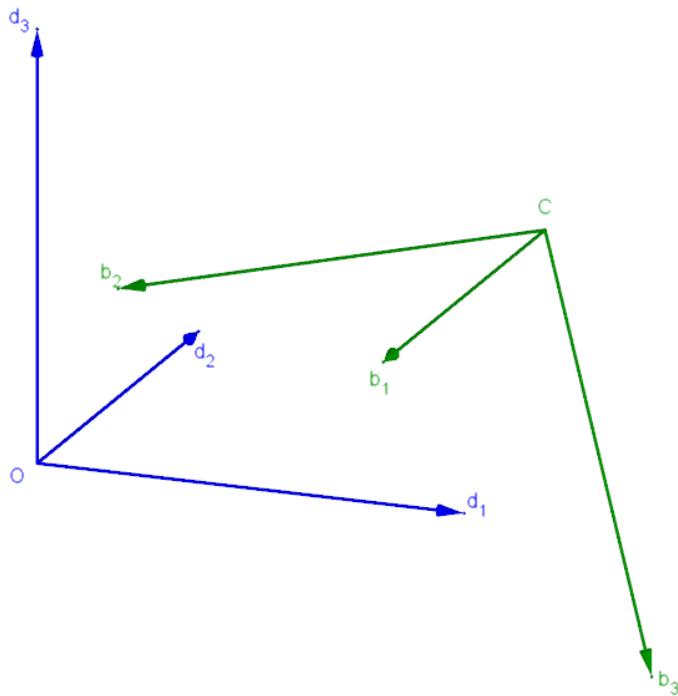


Abbildung 2.1: Weltkoordinatensystem (O, δ) mit $\delta = (d_1, d_2, d_3)$ und Kamerakoordinatensystem (C, β) mit $\beta = (b_1, b_2, b_3)$

Zunächst wird eine sogenannte Koordinatisierung von Punkten im Weltkoordinatensystem vorgenommen. Ein Punkt P_δ bezüglich des Weltkoordinatensystems wird dann wie folgt beschrieben.

$$P_\delta = O + p_{1\delta}d_1 + p_{2\delta}d_2 + p_{3\delta}d_3 \quad (2.1)$$

$$\rightsquigarrow P_\delta = (p_{1\delta}, p_{2\delta}, p_{3\delta})^T = \begin{pmatrix} p_{1\delta} \\ p_{2\delta} \\ p_{3\delta} \end{pmatrix} \quad (2.2)$$

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

$$P_\delta = \begin{bmatrix} p_{1\delta} \\ p_{2\delta} \\ p_{3\delta} \\ 1 \end{bmatrix} = \left\{ k \begin{bmatrix} p_{1\delta} \\ p_{2\delta} \\ p_{3\delta} \\ 1 \end{bmatrix} \in \mathbb{R}^4 \mid k \in \mathbb{R} \right\} \quad (2.3)$$

$$\begin{bmatrix} \lambda p_{1\delta} \\ \lambda p_{2\delta} \\ \lambda p_{3\delta} \\ 1 \end{bmatrix} = \begin{bmatrix} p_{1\delta} \\ p_{2\delta} \\ p_{3\delta} \\ 1 \end{bmatrix} \text{ für } \lambda \neq 0 \quad (2.4)$$

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

$$P_\beta = C + p_{1\beta}b_1 + p_{2\beta}b_2 + p_{3\beta}b_3 \quad (2.5)$$

$$P_\beta = \begin{pmatrix} p_{1\beta} \\ p_{2\beta} \\ p_{3\beta} \end{pmatrix} \quad (2.6)$$

Zwischen den beiden Koordinatensystemen (O, δ) und (C, β) werden die folgenden Beziehungsgleichungen aufgestellt.

$$C_\beta = O_\delta + C_{\beta,1}d_1 + C_{\beta,2}d_2 + C_{\beta,3}d_3 \quad (2.7)$$

$$b_1 = (b_1)_1d_1 + (b_1)_2d_2 + (b_1)_3d_3 \quad (2.8)$$

$$b_2 = (b_2)_1d_1 + (b_2)_2d_2 + (b_2)_3d_3 \quad (2.9)$$

$$b_3 = (b_3)_1d_1 + (b_3)_2d_2 + (b_3)_3d_3 \quad (2.10)$$

Diese Beziehungsgleichungen können nun in Gleichung 2.1 eingesetzt werden.

$$\begin{aligned} P_\delta &= O + (C_{\beta,1} + p_{1\beta}(b_1)_1 + p_{2\beta}(b_2)_1 + p_{3\beta}(b_3)_1) \cdot d_1 \\ &\quad + (C_{\beta,2} + p_{1\beta}(b_1)_2 + p_{2\beta}(b_2)_2 + p_{3\beta}(b_3)_2) \cdot d_2 \\ &\quad + (C_{\beta,3} + p_{1\beta}(b_1)_3 + p_{2\beta}(b_2)_3 + p_{3\beta}(b_3)_3) \cdot d_3 \end{aligned} \quad (2.11)$$

Aus Gleichung 2.11 wird ein Gleichungssystem aufgestellt und gelöst.

$$\begin{aligned} p_{1\delta} &= C_{\beta,1} + (C_{\beta,1} + p_{1\beta}(b_1)_1 + p_{2\beta}(b_2)_1 + p_{3\beta}(b_3)_1) \\ \rightsquigarrow p_{1\delta} - C_{\beta,1} &= (C_{\beta,1} + p_{1\beta}(b_1)_1 + p_{2\beta}(b_2)_1 + p_{3\beta}(b_3)_1) \end{aligned} \quad (2.12)$$

Wenn P_β gegeben ist, erhält man auf diese Weise direkt P_δ . Wenn jedoch P_δ gegeben ist, so muss das LGS aufgestellt und gelöst werden.

$$\begin{bmatrix} (b_1)_1 & (b_2)_1 & (b_3)_1 \\ (b_1)_2 & (b_2)_2 & (b_3)_2 \\ (b_1)_3 & (b_2)_3 & (b_3)_3 \end{bmatrix} \begin{pmatrix} p_{1\beta} \\ p_{2\beta} \\ p_{3\beta} \end{pmatrix} = \begin{pmatrix} p_{1\delta} - C_{\beta,1} \\ p_{2\delta} - C_{\beta,2} \\ p_{3\delta} - C_{\beta,3} \end{pmatrix} \quad (2.13)$$

Wenn (C, β) ein kartesisches Koordinatensystem ist, so ist die entstehende koeffizientenmatrix M_β orthogonal und es gilt $M_\beta^{-1} = M_\beta^T$.

$$M_\beta^T = \begin{bmatrix} (b_1)_1 & (b_1)_2 & (b_1)_3 \\ (b_2)_1 & (b_2)_2 & (b_2)_3 \\ (b_3)_1 & (b_3)_2 & (b_3)_3 \end{bmatrix} \quad (2.14)$$

$$\rightsquigarrow \begin{pmatrix} p_{1\beta} \\ p_{2\beta} \\ p_{3\beta} \end{pmatrix} = M_\beta^T \begin{pmatrix} p_{1\delta} - C_{\beta,1} \\ p_{2\delta} - C_{\beta,2} \\ p_{3\delta} - C_{\beta,3} \end{pmatrix} \quad (2.15)$$

Handelt es sich um kein kartesisches Koordinatensystem, so muss lediglich die Inverse M_β^{-1} anstatt M_β^T gebildet und wie gehabt verfahren werden. Im Folgenden wird noch einmal kompakt und in einer symbolischen Schreibweise die Transformation von Welt- in Kamerakoordinaten festgehalten. Einmal wird wie bereits angefangen mit Spaltenvektoren gearbeitet, das selbe Verfahren wird dann noch einmal mit Zeilenvektoren dargestellt. Beide Ansätze funktionieren nach dem selben Prinzip, nur dass sich die Darstellung der Matrizen ändert. Es ist jedoch gerade wenn man mit Programmen wie Beispielsweise *Matlab* arbeitet wichtig zu wissen welche Darstellung benutzt wird und worin sie sich unterscheiden. Auf diese weise passieren keine Fehler beim Interpretieren der späteren Resultate. *MatLab* arbeitet beispielsweise mit Spaltenvektoren, während im entstandenen Algorithmus dieser Arbeit mit Zeilenvektoren gearbeitet wurde. Zur Erinnerung, es gilt, dass $(\beta = (b_1, b_2, b_3))$ durch Rotation R aus $\delta = (d_1, d_2, d_3)$ 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 gekennzeichnet 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.

$$\begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ C_\beta \end{pmatrix} = \begin{bmatrix} c_{11} & c_{21} & c_{31} & 0 \\ c_{12} & c_{22} & c_{32} & 0 \\ c_{13} & c_{23} & c_{33} & 0 \\ C_{\beta,1} & C_{\beta,2} & C_{\beta,3} & 1 \end{bmatrix} \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ O_\delta \end{pmatrix} \quad (2.16)$$

$$C^T = C^{-1} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \quad (2.17)$$

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

Dementsprechend muss für die Rücktransformation von Kamera in Weltkoordinaten nur wieder die transformierte beziehungsweise die Inverse C^{-1} gebildet werden. Des Weiteren muss der Verschiebungsvektor ebenfalls mit dieser Inversen Multipliziert werden, um diesen ebenfalls in das Zielkoordinatensystem zu überführen.

$$\rightsquigarrow \begin{pmatrix} d_1 \\ d_2 \\ d_3 \\ O_\delta \end{pmatrix} = \begin{bmatrix} c_{11} & c_{21} & c_{31} & 0 \\ c_{12} & c_{22} & c_{32} & 0 \\ c_{13} & c_{23} & c_{33} & 0 \\ -(C_{\beta,1}, C_{\beta,2}, C_{\beta,3})C^{-1} & 1 \end{bmatrix} \begin{pmatrix} b_1 \\ b_2 \\ b_3 \\ C_\beta \end{pmatrix} \quad (2.19)$$

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

$$(b_1, b_2, b_3, C_\beta) = (d_1, d_2, d_3, O) \cdot \begin{bmatrix} c_{11} & c_{21} & c_{31} & C_{\beta,1} \\ c_{12} & c_{22} & c_{32} & C_{\beta,2} \\ c_{13} & c_{23} & c_{33} & C_{\beta,3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.20)$$

Daraus folgt, dass für den Fall der Rücktransformation gilt:

$$\rightsquigarrow (d_1, d_2, d_3, O) = \begin{bmatrix} c_{11} & c_{12} & c_{13} & \\ c_{21} & c_{22} & c_{23} & - \begin{pmatrix} C_{\beta,1} \\ C_{\beta,2} \\ C_{\beta,3} \end{pmatrix} C^{-1} \\ c_{31} & c_{32} & c_{33} & \\ 0 & 0 & 0 & 1 \end{bmatrix} (b_1, b_2, b_3, C_\beta) \quad (2.21)$$

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

$$\begin{bmatrix} c_{11} & & c_{21} & c_{31} & 0 \\ c_{12} & & c_{22} & c_{32} & 0 \\ c_{13} & & c_{23} & c_{33} & 0 \\ -(C_{\beta,1}, C_{\beta,2}, C_{\beta,3})C^{-1} & & & & 1 \end{bmatrix} \quad (2.22)$$

$$\begin{bmatrix} c_{11} & c_{21} & c_{31} & \\ c_{12} & c_{22} & c_{32} & - \begin{pmatrix} C_{\beta,1} \\ C_{\beta,1} \\ C_{\beta,1} \end{pmatrix} C^{-1} \\ c_{13} & c_{23} & c_{33} & \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.23)$$

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

$$(O_\delta) \begin{bmatrix} \begin{pmatrix} c_1 \\ 0 \end{pmatrix}_\delta & \begin{pmatrix} c_2 \\ 0 \end{pmatrix}_\delta & \begin{pmatrix} c_3 \\ 0 \end{pmatrix}_\delta & \begin{pmatrix} C \\ 1 \end{pmatrix}_\delta \end{bmatrix} = (C_\beta) \quad (2.24)$$

$$\rightsquigarrow (d_1, d_2, d_3, O_\delta) = \begin{bmatrix} C & \vec{v} \\ 0 & 0 & 0 & 1 \end{bmatrix} (b_1, b_2, b_3, C_\beta) \quad (2.25)$$

Sei im Weltkoordinatensystem ein Punkt $P_\delta = (d_1, d_2, d_3, O_\delta) \begin{pmatrix} p_{1\delta} \\ p_{2\delta} \\ p_{3\delta} \\ 1 \end{pmatrix} = (b_1, b_2, b_3, C_\beta) \begin{pmatrix} p_{1\beta} \\ p_{2\beta} \\ p_{3\beta} \\ 1 \end{pmatrix}$ im Kamerakoordinatensystem, so liefert die Transformation von Welt- in Kamerakoordinatensystem folgendes:

$$(b_1, b_2, b_3, C_\beta) \begin{bmatrix} C & \vec{v} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} p_{1\beta} \\ p_{2\beta} \\ p_{3\beta} \\ 1 \end{pmatrix} = (d_1, d_2, d_3, O_\delta) \begin{pmatrix} p_{1\delta} \\ p_{2\delta} \\ p_{3\delta} \\ 1 \end{pmatrix} \quad (2.26)$$

Aus der Eindeutigkeit der Koordinatendarstellung folgt:

$$\begin{bmatrix} C & \vec{v} \\ 0 & 0 & 0 & 1 \end{bmatrix} \left(\begin{pmatrix} P_\beta \\ 1 \end{pmatrix}_\beta \right) = \left(\begin{pmatrix} P_\delta \\ 1 \end{pmatrix}_\delta \right) \quad (2.27)$$

Zur Probe, ob die Transformationsmatrix C eine gültige ist, kann sie mit ihrer Inversen multipliziert werden und auf ihren Wahrheitswert geprüft werden. Ergibt sich aus dem Produkt die Einheitsmatrix, sind C und C^{-1} gültige Transformationsmatrizen und C^{-1} auch gleichzeitig die richtige Inverse zu C .

$$\begin{bmatrix} C^T & - \begin{pmatrix} C_{\beta,1} \\ C_{\beta,2} \\ C_{\beta,3} \end{pmatrix} C^{-1} \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} C & C_{\beta,1} \\ C_{\beta,2} & C_{\beta,3} \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.28)$$

2.2 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 späteren Einsatz in 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.

2.3 Aufbau der Koordinatensysteme

Für die geplante Stereoanalyse müssen fünf verschiedene Koordinatensysteme definiert. Das Weltkoordinatensystem (O, δ), die Kamerakoordinatensysteme (C, β), Das Bildebenenkoordinatensystem (I, τ) und das Sensorkoordinatensystem (S, σ). Grundlegend muss erst einmal festgelegt werden, welche Orientierungen die Koordinatensysteme haben sollen. Die Arbeit und auch die entstandenen Algorithmen basieren auf rechtsdrehenden Systemen. Die Möglichkeit linksdrehende Systeme zu benutzen, ist 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 (O, δ) ist mit $\delta = (d_1, d_2, d_3)$ definiert, um die Notation des Beispiels im vorherigen Abschnitts einzuhalten und Verwirrung zu vermeiden. Des Weiteren wird festgehalten, dass das Weltkoordinatensystem deckungsgleich mit dem Koordinatensystem von einer der beiden Kameras ist. Die Bildebene ist gleich einer Ebene im 3D-Raum und wird mit I

bezeichnet. Das Projektionszentrum auf der Bildebene bekommt als Notation Z . Die Kamerakoordinatensysteme (C, β) und (C', β') 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 aus dem vorherigen Kapitel zu Transformationen bekannt, wird das Kamerakoordinatensystem (C, β) mit $\beta = (b_1, b_2, b_3)$ definiert. Des Weiteren soll gelten, dass $C_\beta = Z$, sprich der Ursprung des Kamerakoordinatensystems Deckungsgleich mit dem Projektionszentrum auf der Bildebene I ist und somit auch $\langle d_1, d_2 \rangle + P = I$. 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: $[d_1 \ d_2 \ d_3 \ 1]^T \cdot R = [b_1 \ b_2 \ b_3 \ 1]^T$, wobei R eine 3x3-Rotationsmatrix darstellt. Es kann also wie bereits bekannt eine Matrix R aufgestellt werden, welche das Weltkoordinatensystem in das Kamerakoordinatensystem überführt. Die Matrix komplette Transformationsmatrix wird mit R bezeichnet, um mit den Notationen der Literatur übereinzustimmen[3, 4, 5]. Der Translationsvektor \vec{v} besteht aus den Koordinaten des Projektionszentrums Z . Es gilt also $\vec{v} = Z \rightsquigarrow \vec{v} = \begin{pmatrix} z_1 \\ z_2 \\ z_3 \end{pmatrix}$. Da zuvor bestimmt

wurde dass $C_\beta = Z$ wäre $\vec{v} = \begin{pmatrix} C_{\beta,1} \\ C_{\beta,2} \\ C_{\beta,3} \end{pmatrix}$ als Translationsvektor zu nehmen zu definieren das selbe.

$$(d_1, d_2, d_3)[C] = (b_1, b_2, b_3) \rightsquigarrow b_1 = C_{11}d_1 + C_{21}d_2 + C_{31}d_3 \quad (2.29)$$

$$[C] = \left[\begin{pmatrix} c_1 \\ c_2 \\ c_3 \end{pmatrix}_\delta \right] \quad (2.30)$$

$$(b_1, b_2, b_3, C_\beta) = (d_1, d_2, d_3, O_\delta) \cdot \begin{bmatrix} [C] & z_1 \\ & z_2 \\ & z_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.31)$$

Im nächsten Schritt müssen die Transformierten Kamerakoordinaten noch mit einer entsprechenden Kameramatrix K , welche die Punkte aus dem Kamerakoordinatensystem (C, β) auf das Koordinatensystem der Bildebene (I, τ) mit $\tau = (t_1, t_2)$ projiziert, verrechnet werden. Hierzu muss eine entsprechende Kameramatrix K aufgestellt werden. ζ erhält als Wert den Abstand des Projektionszentrums Z zur Bildebene I .

$$K_{K_{I_\beta}} [\pi]_{K_{C_\beta}} = \begin{bmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & \zeta & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.32)$$

$$\begin{bmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & \zeta & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} \zeta X \\ \zeta Y \\ \zeta Z \\ Z \end{bmatrix} = \begin{bmatrix} \zeta \frac{X}{Z} \\ \zeta \frac{Y}{Z} \\ \zeta \\ 1 \end{bmatrix} \quad (2.33)$$

Für die Koordinaten der Bildebene I_β ergeben sich dann aus den Kamerakoordinaten $[\zeta \frac{X}{Z}, \zeta \frac{Y}{Z}, \zeta, 1]^T$.

Da die Koordinaten momentan noch in homogenen 3D-Koordinaten angegeben sind, aber das Bildebenenkoordinatensystem ein 2D-Koordinatensystem muss der Tiefenwert noch eliminiert werden. In der Literatur wird ζ meist mit f für *focal length* dargestellt und die Bildebene auch *Focal plane* genannt [4].

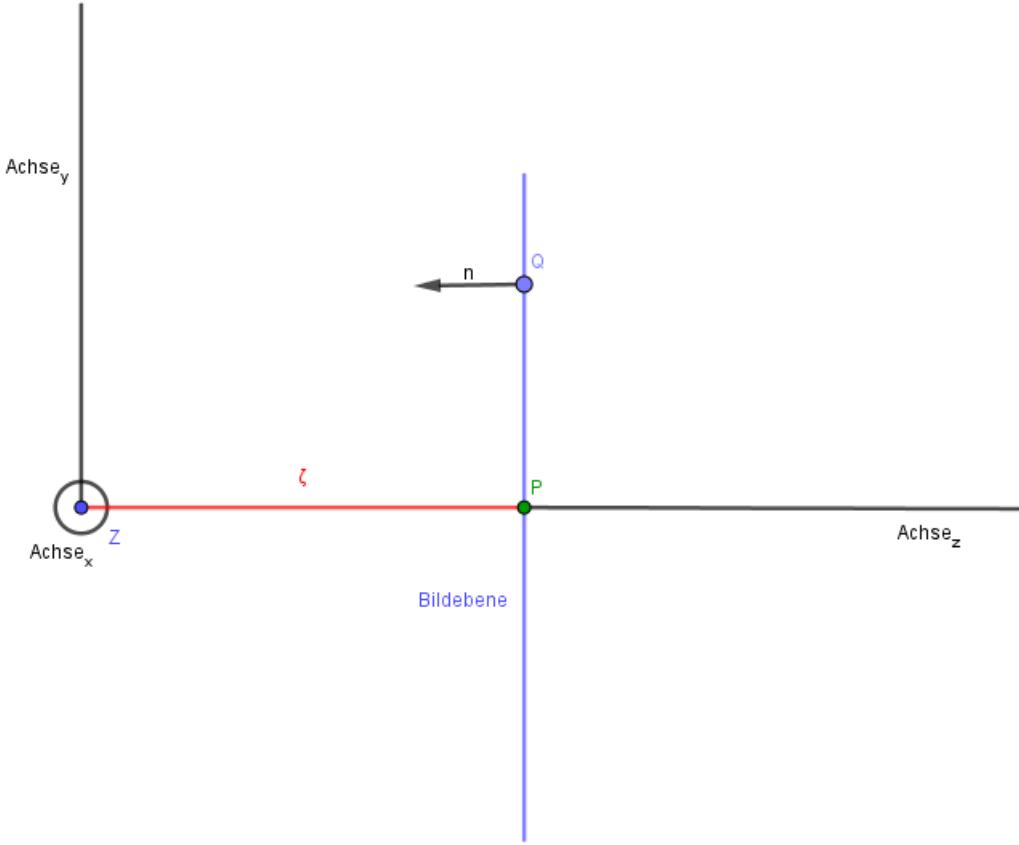


Abbildung 2.2: In blau ist die Bildebene dargestellt auf ihr befinden sich die Punkte Q und P . Z liegt nicht auf der Ebene. Das Projektionszentrum liegt hinter der Bildebene und somit auch hinter dem Sensor. n ist die Normale der Bildebene

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

$$\vec{p} + |\vec{QZ} \cdot \vec{n}| \vec{n} = \vec{Z} \quad (2.34)$$

Q ist ein beliebiger Punkt auf der Ebene. Setze

$$\vec{QZ} \cdot \hat{n} = \zeta \quad (2.35)$$

$$\vec{p} = \vec{Z}\zeta\hat{n} \quad (2.36)$$

Wählt man nun $b_3 = \hat{n}$, 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 Kameramatrix der Form

$$K = {}_{K_I} [\pi] {}_{K_C} = \begin{pmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (2.37)$$

aufgestellt werden. Es gilt allgemein dass $I = (Z - \zeta\hat{n} = \vec{p}, t_1 = b_1, t_2 = b_2)$. Zur Probe ob die Kameramatrix stimmt kann geprüft werden, ob es nach der Matrixmultiplikation mit der Abbildungsmatrix

und den Kamerakoordinaten zu den gewünschten 2D-Bilbenenkoordinaten mit $\tau = (t_1, t_2)$ kommt.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{pmatrix} \zeta X \\ \zeta Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{pmatrix} \zeta X \\ \zeta Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \zeta \frac{X}{Z} \\ \zeta \frac{Y}{Z} \\ 1 \end{pmatrix} \quad (2.38)$$

Um die Kameramatrix allgemeiner zu formulieren und die Bedeutung und hinter ζ und dessen Zusammenhang mit dem in der Literatur benutzten f genauer zu erläutern, wird die Kameramatrix der Photogrammetrie in Bezug auf ein Lochkameramodell hier noch einmal genauer betrachtet und mit dem hergeleiteten Modell verglichen[4, 6]. Zur Beschreibung optischer Kameras greift man üblicherweise auf das Lochkameramodell zurück. Das Modell beruht ausschließlich auf der geometrischen Optik und vernachlässigt physikalische Effekte wie Beugung oder die Auswirkungen der Linse [6]. 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 .

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{pmatrix} \zeta X \\ \zeta Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{pmatrix} \zeta \frac{X}{Z} \\ \zeta \frac{Y}{Z} \\ 1 \end{pmatrix} \quad (2.39)$$

$$\rightsquigarrow \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{pmatrix} f \frac{X}{Z} \\ f \frac{Y}{Z} \\ 1 \end{pmatrix} \quad (2.40)$$

Zum Vergleich dient die Definition im Buch von *Hartley & Zisserman*[4], welche der selbst hergeleiteten entspricht.

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \mapsto \begin{pmatrix} fX \\ fY \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \quad (2.41)$$

Die Kameramatrix K lautet in der Literatur[4]:

$$K = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.42)$$

Anstelle von f steht in der Kameramatrix aus Gleichung 2.42 α_x und α_y . Die Werte welche α_x und α_y annehmen können, hängen von der geometrischen Form der Pixel des in der Kamera verbauten Sensors ab[4, 7]. α_x und α_y sind genau dann einheitlich, wenn die Form der verbauten Pixel, ebenfalls einheitlich quadratisch ist. Die Einheiten der x- und y- Achsen des Sensorkoordinatensystems sind einheitlich. α_x und α_y sind also genau dann ungleich, wenn die auf dem Sensor verbauten Pixel beispielsweise rechteckig oder die geometrische Form eines Parallelogramms besitzen[4]. Im Stereoaufbau dieser Arbeit, wurden Kameras mit CMOS-Chips mit quadratischen Pixeln verwendet, weshalb α_x und α_y einheitlich sein werden. α_x und α_y entstehen aus der Multiplikation der Werte für ζ_x und ζ_y beziehungsweise f_x und f_y mit einem jeweiligen Skalierungswert m_x und m_y . 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[4]. Die Matrixeinträge x_0 und y_0 , bilden einen Translationsvektor. Sie beinhalten die Position des Hauptpunkts auf der Bildebene. x_0 und y_0 sind definiert als $x_0 = p_x \cdot m_x$ und $y_0 = p_y \cdot m_y$. Der Matrixeintrag s wird dem sogenannten *skew-Faktor* zugeordnet, welcher nur dann zum Einsatz kommt, sollte die optische Achse nicht orthogonal auf den Bildsensor

auftreffen. Sprich wenn der Chip geneigt in der Kamera montiert wurde[4]. Die komplette Projektionsmatrix $P = KR = K[C|v]$ beziehungsweise wie sie in der Literatur zitiert wird $P = K[R|t]$ [4] besteht aus der Matrixmultiplikation der hergeleiteten Transformationsmatrix R welche die externen Kameraparameter repräsentiert und der Kameramatrix K , welche die internen Kameraparameter repräsentiert. P beinhaltet also sowohl die internen als auch die externen Kameraparameter. Zuletzt folgt in der Transformationskette noch die Transformation der Bildebenenkoordinaten in die Sensorkoordinaten mit dem Sensorkoordinatensystem (S, σ) mit $\sigma = (\vec{u}, \vec{v})$. \vec{u} und \vec{v} definieren die geometrische Beschaffung der Pixel . Es muss sich dem entsprechend nicht zwangsläufig ein kartesisches Koordinatensystem handeln, obwohl das Bildebenenkoordinatensystem ein kartesisches Koordinatensystem ist. Die Werte z_1 und z_2 stehen für die Verschiebung des Koordinatensystemursprungs von der Mitte der Bildebene in eine der Ecken der Bildebene.

$$\vec{u} = u_1 t_1 + u_2 t_2 \quad (2.43)$$

$$\vec{v} = v_1 t_1 + v_2 t_2 \quad (2.44)$$

$$S_\sigma = I_\tau + z_1 t_1 + z_2 t_2 \quad (2.45)$$

$$(\vec{u}, \vec{v}, S) = (t_1, t_2, I_\tau) \cdot \begin{bmatrix} u_1 & v_1 & z_1 \\ u_2 & v_2 & z_2 \\ 0 & 0 & 1 \end{bmatrix} \quad (2.46)$$

Der Koordinatenwechsel von Bildebenenkoordinaten in Sensorkoordinaten wird noch einmal Symbolisch veranschaulicht.

$$\text{Es sei ein Punkt } X_\sigma = \begin{pmatrix} a \\ b \\ 1 \end{pmatrix}$$

$$\rightsquigarrow x = a\vec{u} + b\vec{v} + S_\sigma \quad (2.47)$$

$$= a(u_1 t_1 + u_2 t_2) + b(v_1 t_1 + v_2 t_2) + I_\tau(z_1 t_1 + z_2 t_2) \quad (2.48)$$

$$\mapsto X_\tau = \begin{bmatrix} z_1 + av_1 + bv_1 \\ z_2 + au_2 + bu_2 \end{bmatrix} \quad (2.49)$$

$$(2.50)$$

Die Transfomationsmatrix $M = \begin{bmatrix} u_1 & v_1 \\ u_2 & v_2 \end{bmatrix}$ beinhaltet statt einer Rotation eine Skalierung der Koordinatenwerte auf die Koordinaten des Sensorkoordinatensystems. Mit M lässt sich wieder eine Projektionsmatrix P aus einer Abbildungsmatrix K mit einer Transformationsmatrix M aufstellen. Ein Punkt X_τ wird in einen Punkt X_σ transformiert.

$$X_\sigma = \begin{bmatrix} [u_1 & v_1]^{-1} & -M^{-1} \begin{pmatrix} z_1 \\ z_2 \end{pmatrix} \\ 0 & 0 & 1 \end{bmatrix} \cdot X_\tau \quad (2.51)$$

Die Projektionsmatrix P , welche die Kamerakoordinaten in Sensorkoordinaten überführt, wird dann wie folgt aufgebaut:

$$P = \begin{bmatrix} M^{-1} & -M \begin{pmatrix} z_1 \\ z_2 \end{pmatrix}^{-1} \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} -\zeta & 0 & 0 & 0 \\ 0 & -\zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} -\zeta M^{-1} & -M \begin{pmatrix} p_1 \\ p_2 \end{pmatrix}^{-1} & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \quad (2.52)$$

Zur Verdeutlichung folgen nun noch zwei Beispiele. Es werden \vec{u} und \vec{v} , sowie z_1 und z_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.

Beispiel 1:

$$\vec{u} = 1pt_1$$

$$\vec{v} = 2pt_2$$

$$p_1 = 15, p_2 = 20$$

Für die Projektionsmatrix P ergibt sich somit:

$$S_\sigma = I_\tau - \vec{u} - \vec{v} \rightsquigarrow S_\sigma = I_\tau - 15t_1 - 20t_2 \quad (2.53)$$

$$M = \begin{bmatrix} 1 & 0 \\ 0 & 2 \end{bmatrix} \rightsquigarrow M^{-1} = \begin{bmatrix} \frac{1}{p} & 0 \\ 0 & \frac{1}{2p} \end{bmatrix} \quad (2.54)$$

$$K = {}_{S_\sigma} [\pi]_{I_\tau} = \begin{bmatrix} \frac{-\zeta}{p} & 0 & 15 & 0 \\ 0 & \frac{-\zeta}{p} & 20 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.55)$$

Beispiel 2:

$$\vec{v} = 1pt_1 + 2pt_2$$

$$\vec{u} = 1pt_1$$

$$z_1 = 10, z_2 = 5$$

$$M = \begin{bmatrix} 1 & 1 \\ 0 & 2 \end{bmatrix} \rightsquigarrow M^{-1} = \begin{bmatrix} 1 & -\frac{1}{2} \\ 0 & \frac{1}{2} \end{bmatrix} \quad (2.56)$$

$$K = {}_{S_\sigma} [\pi]_{C_\beta} = \begin{bmatrix} \frac{-\zeta}{p} & \frac{-\zeta}{2p} & 10 & 0 \\ 0 & \frac{-\zeta}{p} & 5 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (2.57)$$

$$= [K|0] \quad (2.58)$$

Die symbolische Darstellung der Projektionsmatrix von Welt in Sensorkoordinaten $P = {}_{S_\sigma} [\pi]_{O_\delta}$ fügt sich somit aus dem vorhergehenden Abbildungsmatrix $K = {}_{S_\sigma} [\pi]_{C_\beta}$ mit der Tranformationsmatrix R zusammen.

$${}_{S_\sigma} [\pi]_{O_\delta} = {}_{S_\sigma} [\pi]_{C_\beta} \cdot \begin{bmatrix} [C]^{-1} & -[C]^{-1}Z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.59)$$

Somit ist die Transformations-Pipeline eines Punktes im Raum auf den Sensor einer Kamera am Ende. Zum Schluss hier noch einmal zum Vergleich die Darstellung der Projektionsmatrix P aus *Hartley&Zisserman*[4]. Zu beachten ist hier, dass im *Hartley&Zisserman* R für $[R]^{-1}$ steht[4].

$$\begin{aligned}
P &= [K|0] \begin{bmatrix} R & -RZ \\ 0 & 0 & 0 & 1 \end{bmatrix} \\
&= [KR| -KRZ] = KR[I_{3x3}| -Z]
\end{aligned} \tag{2.60}$$

3 Homographien in der Ebene

Im vorherigen Kapitel wurde ausführlich dargelegt, wie Koordinatensysteme ineinander überführt werden können. Die folgenden Kapitel sollen anhand eines Beispiels zeigen, dass sich die in Kapitel (Link Kapitel) gezeigten Transformationen in Matrizen zusammenfassen lassen. Es soll eine Matrix H aufgestellt werden, welche die Bildpunkte zweier Kameras ineinander überführen kann, ohne, dass die intrinsischen und extrinsischen Parameter bekannt sind. Für den Aufbau des Minimalbeispiels, werden die entsprechenden Kameramatrizen aufgestellt, jedoch wird gezeigt, dass sie für die Herleitung von H keine Rolle spielen. Im ersten Beispiel wird sowohl davon ausgegangen, dass die intrinsischen und extrinsischen Kameraparameter unbekannt sind und nur die Bildpunkte von beiden Kamerakoordinatensystemen bekannt sind, als auch, dass die beiden Kameras sich ein Projektionszentrum teilen. Des Weiteren wird festgelegt, dass sich alle Objektpunkte im Raum \mathbb{R}^3 auf einer Ebene befinden. Nun wird behauptet, dass es möglich ist eine sogenannte 3x3-Homographiematrix H zu ermitteln, welche die Punkte von Kamera eins in die Punkte von Kamera zwei und umgekehrt überführen kann. Als Homographie wird eine projektive Transformation zwischen zwei Ebenen bezeichnet. Dabei bleiben Kollinearitäten und die Reihenfolge von Punkten auf Geraden, wie zum Beispiel Schnittpunkte mit anderen Geraden, erhalten. Aufgrund dieser Ebenenannahme, kann solch eine projektive Transformation durch eine 3x3-Homographiematrix ausgedrückt werden[8]. Sprich die entstehende Homographiematrix H , projiziert jede Figur in eine Figur gleicher projektiver Entsprechung[4, 3]. Sind die Punkte A', B', C' und D' die projektiven Bilder eines Systems von vier kollinearen Punkten, so ist $(A', B', C', D') = (A, B, C, D)$ [9]. Die Homographie fasst die Transformationen wie Rotation und Translation, sowie die jeweiligen Abbildungsmatrizen der Kameras in einer 3x3-Matrix zusammen.[3, 9].

Es seien $m = \begin{pmatrix} m_1 \\ m_2 \\ m_3 \end{pmatrix}$ die homogenen Koordinaten eines Punktes der projektiven Ebene und $m' = \begin{pmatrix} m'_1 \\ m'_2 \\ m'_3 \end{pmatrix}$ die Punkte des projektiv transformierten Punktes. Dann gilt

$$m' = Hm \quad (3.1)$$

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

$$\rightsquigarrow m' = Hm = \begin{bmatrix} h_{11}m_1 + h_{12}m_2 + h_{13}m_3 \\ h_{21}m_1 + h_{22}m_2 + h_{23}m_3 \\ h_{31}m_1 + h_{32}m_2 + h_{33}m_3 \end{bmatrix} \quad (3.3)$$

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

Dabei müssen die Koeffizienten so geartet sein, dass die zugehörige Transformation umkehrbar ist. [4][9]. Sprich es muss gelten dass wenn

$$m' = Hm \quad (3.5)$$

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

Die Einträge der Homographiematrix soll die Transformationen der Kameras zueinander und deren jeweilige Abbildungsmatrizen in einer 3×3 -Matrix zusammenfassen. Im folgenden wird die Homographiematrix H in ihre Elemente zerlegt, um die Herleitung von H für zwei Kameras mit identischem Projektionszentrum aufzuzeigen[4, 9, 3]. zunächst wird festgelegt, dass für Punkte im Weltkoordinatensystem die Darstellung zur Basis (O, δ) gilt. Für Punkte in den jeweiligen Bildebenekoordinaten gelten die Basen (C, β) und (C', β') . Für die beiden Kameras soll des Weiteren gelten, dass $C_{1\delta} = C_{2\delta}$. Diese Kameras projizieren nun einen Punkt M_δ im Raum auf die jeweiligen Bildebene $I_{1\delta}$ und $I_{2\delta}$ der beiden Kameras. Auf den Bildebene entstehen die projizierten Bildpunkte von M_δ als m_β und $m'_{\beta'}$.

(GRAFIK NOCH ANPASSEN MIT m_β etc)

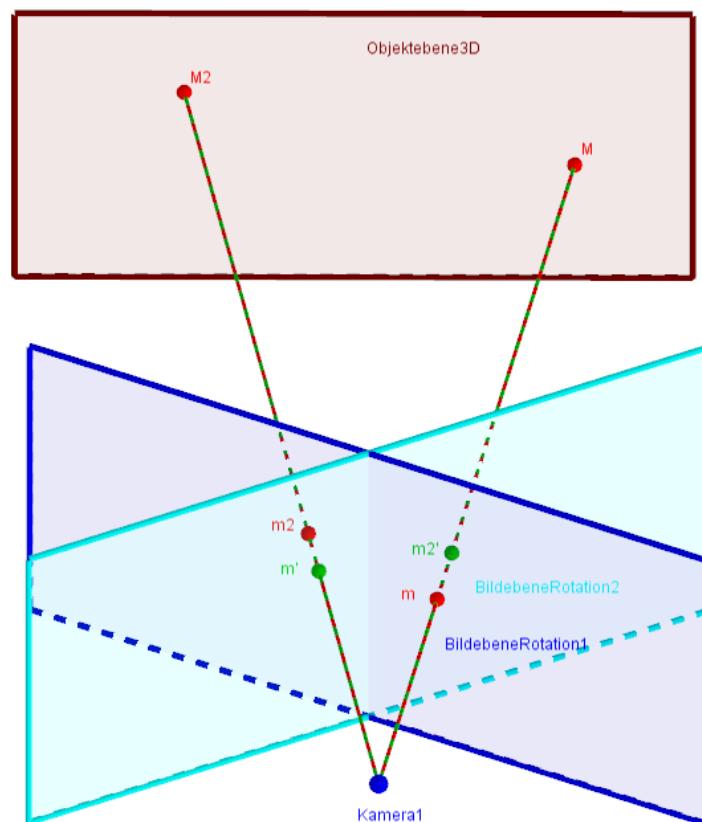


Abbildung 3.1: Veranschaulichung von Homographie mit nur einer rotierten Kamera.

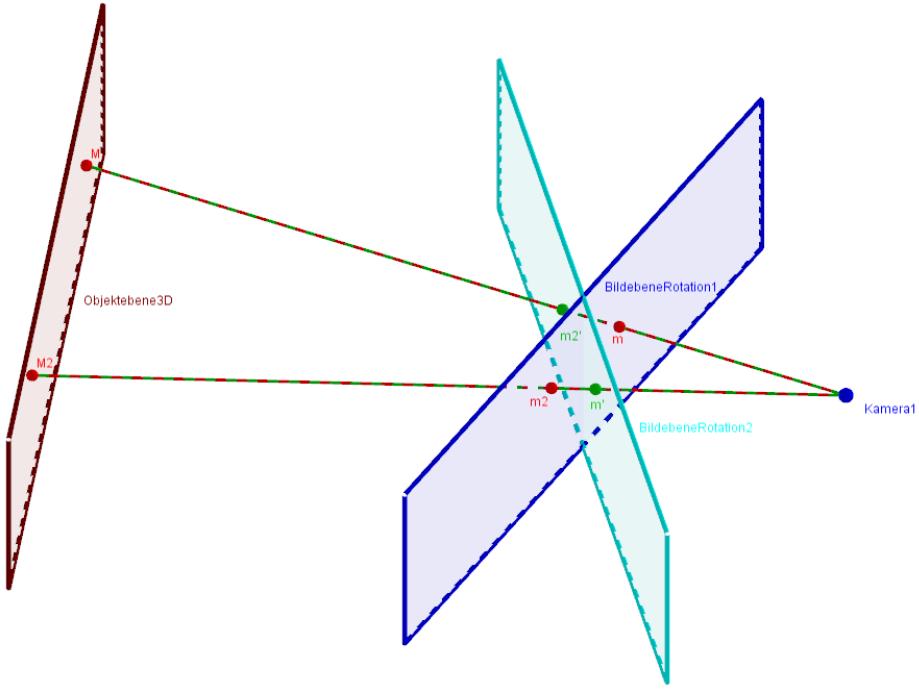


Abbildung 3.2: Veranschaulichung von Homographie mit nur einer rotierten Kamera.

Die Umrechnung des Objektpunktes M_δ zu den abgebildeten Punkten m_β und m'_β mit Basis zu ihren jeweiligen Bildebenenkoordinatensystemem (o, β) und (o', β') funktioniert wie bereits von den Transformationen in den vorherigen Kapitel (KAPITELLINK) bekannt. Hier wird der Vorgang nochmal symbolisch in den Gleichungen 3.7 und 3.8 aufgeführt. Die Projektionsmatrizen, welche die Rotationen und Abbildungsmatrizen der einzelnen Kameras zusammenfasst wird hier mit P für Kamer eins und P' für Kamera zwei abgekürzt.[3]

$$\gamma \vec{m}_\beta = P \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = [KR| - KR\vec{C}_\delta] \cdot \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = KR(\vec{M}_\delta - \vec{C}_\delta) \quad (3.7)$$

$$\gamma' \vec{m}'_\beta = P' \begin{bmatrix} \vec{M}'_\delta \\ 1 \end{bmatrix} = [K'R'| - K'R'\vec{C}_\delta] \cdot \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = K'R'(\vec{M}_\delta - \vec{C}_\delta) \quad (3.8)$$

Der Vektor $(\vec{M}_\delta - \vec{C}_\delta)$, beschreibt die Verbindung zwischen dem Objektpunkt M_δ der Objektebene im Raum und dem Projektionszentrum C_δ im Raum. Die Gleichung sind bereits aus Kapitel(HIER KAPITELLINK),Gleichungen 2.73 oder 2.74, bekannt. Es handelt sich hier lediglich wieder um eine Transformation von einem Punkt in 3D Weltkoordinaten über einen Punkt eines 3D- Kamerakoordinatensystems, in 2D-Bildkoordinaten. Jetzt soll aber gezeigt werden, dass die Bildpunkte nur mit einer Homographiematrix, ohne Kenntniss der Kameraparameter ineinander überführt werden können und das alles nur mit den beiden Bildkoordinaten m_β und m'_β . Dazu wird im folgenden erst einmal aufgeführt, wie man aus Gleichung 3.7 und 3.8 die Homographiematrix H gewinnen kann. Was bei den zwei Gleichungen auffällt ist, dass die rechte Seite der Gleichungen, die Ergebnisse nur in ihren Kameraparametern KR und RK unterscheiden, während der Vektor $(\vec{M}_\delta - \vec{C}_\delta)$, der selbe ist. Das liegt daran, dass zuvor festgelegt wurde, dass die Kameras identische Projektionszentren besitzen. Also gilt $\vec{C}_{1\delta} = \vec{C}_{2\delta} = \vec{C}_\delta$. Mit Hilfe dieser Bedingung können wir die Gleichungen gleichsetzen und für H das folgende Ergebnis herleiten.

$$\gamma R K^{-1} \vec{m}_\beta = \vec{M}_\delta - \vec{C}_\delta \quad (3.9)$$

$$\gamma' R' K'^{-1} \vec{m}_{\beta'} = \vec{M}_\delta - \vec{C}_\delta \quad (3.10)$$

$$\rightsquigarrow \gamma' R' K'^{-1} \vec{m}_{\beta'} = \gamma R K^{-1} \vec{m}_\beta \quad (3.11)$$

$$\frac{\gamma'}{\gamma} \vec{m}_{\beta'} = K' R' R^T K^{-1} \vec{m}_\beta \quad (3.12)$$

$$\lambda \vec{m}'_{\beta'} = H \vec{m}_\beta \quad (3.13)$$

Es Resultiert also dass wenn $\lambda = \frac{\gamma'}{\gamma}$, dann gilt dass $H = K' R' R^T K^{-1}$ gleicht. Gilt nun noch der Fall, dass $K = K'$ ist, dann reduziert sich H zu $K^{-1} H K = R' R$, welches einer schlichten Rotation gleicht.[3]

3.1 Homographie zwischen der Abbildungen eines Quadrates einer definierten Ausgangskamera und einer um ihr Projektionszentrum rotierten Kamera

In diesem Beispiel werden die abgebildeten Punkte eines Quadrats einer Kamera (C_β) mit $\beta = [\vec{b}_1, \vec{b}_2, \vec{b}_3, C]$ in die einer anderen Kamera (C', β') mit $\beta' = [\vec{b}'_1, \vec{b}'_2, \vec{b}'_3, C']$ mit Hilfe einer selbst aufgestellten Homographiematrix überführt. Hierzu wird zunächst eine Szene mit den Eckpunkten des Quadrates $A_\delta, B_\delta, C_\delta, D_\delta$ und dessen Mittelpunkt mit E_δ und zwei Kameras C_δ und C'_δ definiert. Für Die Projektionszentren der Kameras gilt in hier, dass $C'_\delta = C_\delta$. Die Eckpunkte so wie die Kameras werden in Weltkoordinaten mit den Basen (O, δ) mit $\delta = [\vec{d}_1, \vec{d}_1, \vec{d}_1, O]$ angegeben. Mit Hilfe der gelernten Transformationsmethoden, werden die Eckpunkte in ihre jeweiligen 2D-Bildebenenkoordinaten (I, τ) und (I', τ') umgerechnet. Nachdem die Szene vollständig definiert und die auf den Kameras abgebildeten Punkte berechnet sind, werden zwei Methoden gezeigt, mit welchem die Homographiematrix ermittelt werden kann. Da es sich in diesem Beispiel um ein selbst aufgestelltes Minimalbeispiel handelt, kommt es zu keinem überbestimmten Fall bei der der Berechnung der Homographiematrix. Da in Realbeispielen, jedoch immer damit gerechnet werden muss, dass es zu einem überbestimmten Fall kommt, wird das Verfahren, welches diesen Fall abdeckt ebenfalls aufgezeigt. Der Szenenaufbau des Minimalbeispiels soll folgendermaßen aussehen. Die Kamerakoordinatensysteme unterscheiden sich vom Weltkoordinatensystem durch eine Drehung um 180° um die \vec{d}_1 -Achse. Der Ursprung beider Kamerakoordinatensysteme entspricht dem Projektionszentrum C . Des Weiteren ist Kamera zwei noch um 45° um die \vec{b}_2 zu Kamera eins eingedreht. 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. In Abbildung 3.3 ist der Aufbau nochmal grafisch veranschaulicht.

$$H = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \quad (3.14)$$

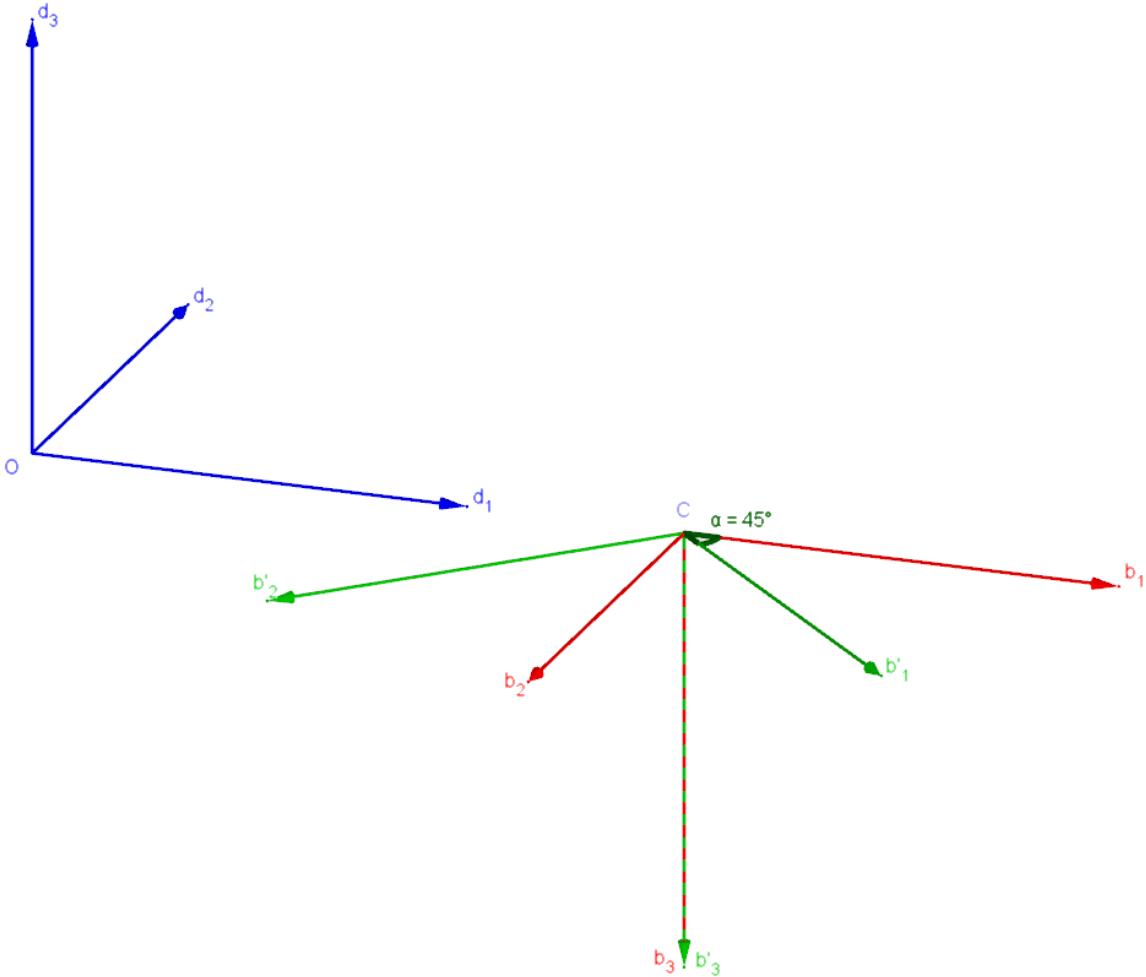


Abbildung 3.3: Weltkoordinatensystem (O, δ) mit $\delta = [\vec{d}_1, \vec{d}_1, \vec{d}_1, O]$ und Kamerakoordinatensysteme $\beta = [\vec{b}_1, \vec{b}_2, \vec{b}_3, C]$ und (C', β') mit $\beta' = [\vec{b}'_1, \vec{b}'_2, \vec{b}'_3, C']$.

Als nächstes werden die intrinsischen Kameraparameter K und K' für beide Kameras festgelegt. Für das Beispiel

$$K = K' = \begin{pmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & \zeta & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.15)$$

$$\rightsquigarrow \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = K \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{pmatrix} \zeta X \\ \zeta Y \\ \zeta Z \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{\zeta}{Z} X \\ \frac{\zeta}{Z} Y \\ Z \\ 1 \end{pmatrix} \quad (3.16)$$

Für das Beispiel gilt die Bedingung, dass $\zeta \neq 0$ sein soll. Des Weiteren soll gelten, dass \vec{b}_1 gleich der Lotgeraden vom Objektpunkt zum Projektionszentrum entspricht und somit folgt, dass \vec{b}_1 in Sensorebene liegt und $\vec{b}_2 = \vec{b}_3 \times \vec{b}_1$ ist. Der Quader besteht aus den Punkten $A_\delta, B_\delta, C_\delta, D_\delta$ und $E\delta$ in homogenen Weltkoordinaten. Die Punkte in Weltkoordinaten bekommen die Koordinatenwerte zugewiesen.

$$A_\delta = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix}, B_\delta = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}, C_\delta = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \end{pmatrix}, D_\delta = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix}, E_\delta = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 2 \\ 1 \end{pmatrix} \quad (3.17)$$

Für die Transformation der Weltkoordinatentupel in die Bildebenen I und I' werden die Projektionsmatrizen $P = [KR| - KRC_\delta]$ und $P' = [K'R'| - K'R'C_\delta]$ aufgestellt werden. R soll eine Drehung der Punkte um 180° um die \vec{d}_1 -Achse durchführen. R' muss zusätzlich im Anschluss noch eine Drehung, um 45° um die neue \vec{b}_3' -Achse von Kamera zwei, an die vorherige Drehung anhängen. Die so erhaltenen Matrizen R und R' können nun dazu verwendet werden, die Objektpunkte bezüglich des Weltkoordinatensystems (O, δ) in Punkte bezüglich der jeweiligen Kamerakoordinatensysteme (C, β) und (C', β') zu transformieren.

$$\begin{bmatrix} \cos(180) & -\sin(180) & 0 & 0 \\ \sin(180) & \cos(180) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (\vec{d}_1, \vec{d}_2, \vec{d}_3, O) = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (\vec{d}_1, \vec{d}_2, \vec{d}_3, O) \quad (3.18)$$

$$\begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = R \quad (3.19)$$

Im nächsten Schritt wird Kamera zwei noch um 45° um ihre \vec{b}_3' -Achse gedreht. Diese Transformation wird mit der vorherigen Transformation multipliziert und es entsteht P' für Kamera zwei.

$$\cos(45) = \sin(45) = \frac{1}{\sqrt{2}} = a \quad (3.20)$$

$$\begin{bmatrix} a & 0 & a & 0 \\ 0 & 1 & 0 & 0 \\ -a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} (\vec{d}_1, \vec{d}_2, \vec{d}_3, O) \quad (3.21)$$

$$= \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}^T = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = R' \quad (3.22)$$

Handelt es sich bei den Kamerakoordinatensystemen nicht um kartesische Koordinatensysteme, so muss von den Rotationsmatrizen jeweils die Inverse genommen werden, um R und R' zu erhalten. Die Inversen Rotationsmatrizen werden in den folgenden Gleichungen nicht mit R^{-1} , sondern R bezeichnet. So entsprechen sie auch den Notationen der Literatur[3, 4]. Sind die Matrizen R und R' ermittelt, können nun die Punkte des Quadrats im Raum in Punkte der jeweiligen Kameras umgerechnet werden. Für Punkte spezifisch Kamera eins ergeben sich die in den Gleichungen 3.23 bis 3.27 aufgezeigten Werte.

$$\begin{pmatrix} A_\beta \\ 1 \end{pmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix} \quad (3.23)$$

$$\begin{pmatrix} B_\beta \\ 1 \end{pmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -1 \\ 0 \\ 2 \\ 1 \end{pmatrix} \quad (3.24)$$

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

$$\begin{pmatrix} D_\beta \\ 1 \end{pmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 0 \\ -1 \\ 2 \\ 1 \end{pmatrix} \quad (3.26)$$

$$\begin{pmatrix} E_\beta \\ 1 \end{pmatrix} = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{1}{2} \\ -\frac{1}{2} \\ 2 \\ 1 \end{pmatrix} \quad (3.27)$$

Für Punkte spezifisch Kamera zwei ergeben sich die Punkte aus den Gleichungen 3.28 bis 3.32.

$$\begin{pmatrix} A'_{\beta'} \\ 1 \end{pmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2a \\ 0 \\ 2a \\ 1 \end{pmatrix} \quad (3.28)$$

$$\begin{pmatrix} B'_{\beta'} \\ 1 \end{pmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ 0 \\ 3a \\ 1 \end{pmatrix} \quad (3.29)$$

$$\begin{pmatrix} C'_{\beta'} \\ 1 \end{pmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} a \\ -1 \\ 3a \\ 1 \end{pmatrix} \quad (3.30)$$

$$\begin{pmatrix} D'_{\beta'} \\ 1 \end{pmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} 2a \\ -1 \\ 2a \\ 1 \end{pmatrix} \quad (3.31)$$

$$\begin{pmatrix} E'_{\beta'} \\ 1 \end{pmatrix} = \begin{bmatrix} -a & 0 & a & 0 \\ 0 & -1 & 0 & 0 \\ a & 0 & a & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 2 \\ 1 \end{pmatrix} = \begin{pmatrix} \frac{3}{2}a \\ -\frac{1}{2} \\ \frac{5}{2}a \\ 1 \end{pmatrix} \quad (3.32)$$

Die entstandenen Punkte sind nun entsprechend der Kamerakoordinatensysteme definiert. Jetzt müssen noch die jeweiligen Kameramatrizen K und K' mit diesen verrechnet werden, um so die 2D-Bildebenenpunkte der entsprechenden Kameras zu erhalten.

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° gedreht auf dem Sensor abgebildet.

$$K = K' = \begin{pmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & \zeta & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.33)$$

Man kann natürlich wie bereits gewohnt die Rotationsmatrizen und die Kameramatrizen bereits im Vorhinein miteinander multiplizieren und dann auf einen 3D-Weltpunkt anwenden. Auf diese Weise bekommt man die fertigen Projektionsmatrizen $P = [KR| - KRC_\delta]$ und $P' = [K'R'| - K'R'C_\delta]$. In diesem Beispiel werden sie nochmal getrennt voneinander auf die Punkte angewandt. Die fünf Punkte werden jeweils in einer Matrix zusammengeschrieben und mit K beziehungsweise K' verrechnet.

$$K \cdot \begin{pmatrix} 0 & -1 & -1 & 0 & -\frac{1}{2} \\ 0 & 0 & -1 & -1 & -\frac{1}{2} \\ 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 0 & -1 & -1 & 0 & -\frac{1}{2} \\ 0 & 0 & -1 & -1 & -\frac{1}{2} \\ 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.34)$$

$$= \begin{pmatrix} 0 & 1 & 1 & 0 & -\frac{1}{2} \\ 0 & 0 & 1 & 1 & -\frac{1}{2} \\ -2 & -2 & -2 & -2 & -2 \\ 2 & 2 & 2 & 2 & 2 \end{pmatrix} \quad (3.35)$$

$$\simeq \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{4} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \simeq \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{4} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.36)$$

$$K' \cdot \begin{pmatrix} 2a & a & a & 2a & \frac{3}{2}a \\ 0 & 0 & -1 & -1 & -\frac{1}{2} \\ 2a & 3a & 3a & 2a & \frac{5}{2}a \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} 2a & a & a & 2a & \frac{3}{2}a \\ 0 & 0 & -1 & -1 & -\frac{1}{2} \\ 2a & 3a & 3a & 2a & \frac{5}{2}a \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.37)$$

$$= \begin{pmatrix} -2a & -a & -a & -2a & -\frac{3}{2}a \\ 0 & 0 & 1 & 1 & \frac{1}{2} \\ -2a & -3a & -3a & -2a & -\frac{5}{2}a \\ 2a & 3a & 3a & 2a & \frac{5}{2}a \end{pmatrix} \quad (3.38)$$

$$\simeq \begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} & -1 & \frac{3}{5}a \\ 0 & 0 & \frac{1}{3a} & \frac{1}{2a} & \frac{1}{5}a \\ -1 & -1 & -1 & -1 & -1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \simeq \begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} & -1 & \frac{3}{5} \\ 0 & 0 & \frac{1}{3a} & \frac{1}{2a} & \frac{1}{5} \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.39)$$

Die entstandenen Punkte beider Kameras ergeben die in Abbildung 3.4 veranschaulichten Abbildungen, des Quadrats mit seinem Mittelpunkt, auf den jeweiligen Kamerabildebenen. Die aus den Objektpunkten M_δ entstandenen 2-D-Bildebenenkoordinaten m_τ und $m'_{\tau'}$ werden in den Koordinatensystemen (I, τ) mit $(\tau = \vec{b}_1, \vec{b}_2, \vec{b}_4)$ und (I', τ') mit $(\tau = \vec{b}'_1, \vec{b}'_2, \vec{b}'_4)$ dargestellt.

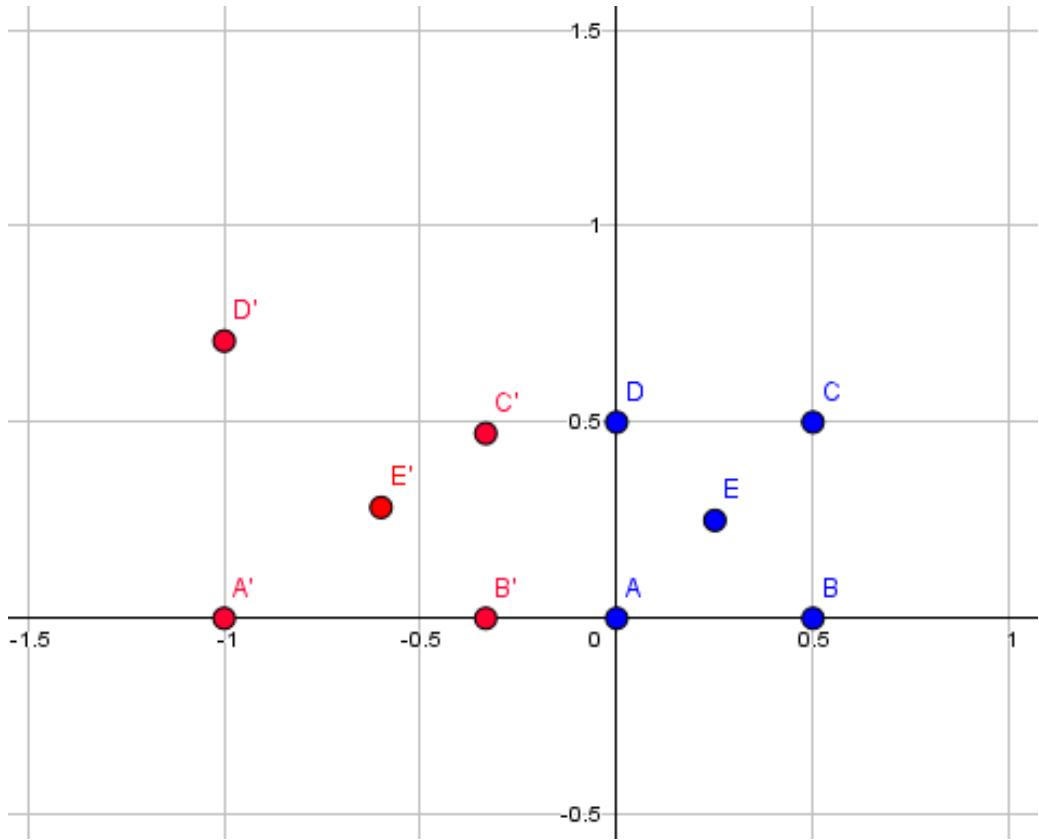


Abbildung 3.4: in blau ist die Abbildung des Quaders von Kamera eins und in rot die Abbildung des selben Quaders in Kamera zwei

Die fertigen Bildebenekoordinaten beider Kameras, sollen nun durch eine eigens aufgestellte Homographiematrix H 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 erhalten werden die Punkte beider Kameras in eine Koeffizientenmatrix eingetragen, welche sich nach dem in den Gleichungen 4.40 bis 3.52 laufenden Schema ergibt. Das Verfahren wird anhand der Bildebenekoordinaten aufgezeigt.

$$H \cdot m_\tau = m'_\tau \quad (3.40)$$

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} m_\tau \end{bmatrix} = \begin{bmatrix} m'_{\tau'} \end{bmatrix} \quad (3.41)$$

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad (3.42)$$

Aus Gleichung 3.42 lassen sich ein Gleichungssystem mit zwölf bekannten und neun unbekannten aufstellen.

$$h_{11}x + h_{12}y + h_{13}z = \lambda x' \quad (3.43)$$

$$h_{21}x + h_{22}y + h_{23}z = \lambda y' \quad (3.44)$$

$$h_{31}x + h_{32}y + h_{33}z = \lambda z' \quad (3.45)$$

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 in den ersten beiden Gleichungen für λ eingesetzt werden. Pro Punktpaar m_τ und $m'_{\tau'}$ ergeben sich somit zwei Gleichungen.

$$h_{11}x + h_{12}y + h_{13}z = (h_{31}x + h_{32}y + h_{33}z) \cdot x' \quad (3.46)$$

$$h_{21}x + h_{22}y + h_{23}z = (h_{31}x + h_{32}y + h_{33}z) \cdot y' \quad (3.47)$$

Für den Aufbau der nötigen Koeffizientenmatrix werden beide Ausdrücke noch nach Null aufgelöst, so dass sich die Gleichungen 3.48 und 3.49 aus 3.46 und 3.47 ergeben.

$$h_{11}x + h_{12}y + h_{13}z - (h_{31}x + h_{32}y + h_{33}z) \cdot x' = 0 \quad (3.48)$$

$$h_{21}x + h_{22}y + h_{23}z - (h_{31}x + h_{32}y + h_{33}z) \cdot y' = 0 \quad (3.49)$$

$$\rightsquigarrow h_{11}x + h_{12}y + h_{13}z - h_{31}x \cdot x' - h_{32}y \cdot x' - h_{33}z \cdot x' = 0 \quad (3.50)$$

$$\rightsquigarrow h_{21}x + h_{22}y + h_{23}z - h_{31}x \cdot y' - h_{32}y \cdot y' - h_{33}z \cdot y' = 0 \quad (3.51)$$

Die entstandenen Gleichungen werden jetzt pro Punktpaar m_τ und $m'_{\tau'}$ in eine Matrix nach folgendem Schema eingetragen.[3, 4, 10, 6]

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 & 1 \cdot x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & x_1y'_1 & y_1y'_1 & 1 \cdot y'_1 \\ & & & \ddots & & & & & \\ & & & \ddots & & & & & \\ & & & \ddots & & & & & \\ x_i & y_i & 1 & 0 & 0 & 0 & x_ix'_i & y_ix'_i & 1 \cdot x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & x_iy'_i & y_iy'_i & 1 \cdot y'_i \end{pmatrix} \cdot \begin{pmatrix} h1 \\ h2 \\ \vdots \\ \vdots \\ hi \end{pmatrix} = 0 \quad (3.52)$$

Wenn ein nicht überbestimmter Fall vorliegt, sprich wenn der Rang der Koeffizientenmatrix genau acht und nicht höher beträgt, kann aus der Koeffizientenmatrix einfach der Nullraum berechnet werden, um so die Einträge für die 3x3-Homographiematrix zu erhalten[4, 3, 10]. Gesucht wird also ein Vector \vec{x} , für den gilt das $H \cdot \vec{x} = 0$. Der gesuchte Vektor \vec{x} entspricht dem Kern der Koeffizientenmatrix und ist ein Spaltenvektor mit 9 Einträgen, welche in die 3x3-Homographiematrix eingetragen werden können[4, 10]. Tritt nun der Fall ein, dass es zu einem überbestimmtes System kommt, **was Beipielweise auftritt wenn mehr als neun Punktpaare durch eine Homographie ineinander überführt werden sollen**, so kann nicht mehr die Ermittlung des Nullraums für die Berechnung der Homographiematrix genutzt werden. Für die Lösung überbestimmter Gleichungssysteme bietet sich die Singulärwertzerlegung an[4][11]. Das bedeutet es wird nicht derjenige Vektor \vec{x} gesucht für den gilt $H \cdot \vec{x} = 0$, sondern es wird derjenige Vektor \vec{x} gesucht, für den $\| H \cdot \vec{x} \|$ minimal wird[4, 10]. Die Singulärwertzerlegung von zum Beispiel der Koeffizientenmatrix ist 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.

$$S = \begin{pmatrix} s_1 & & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & & \ddots & \ddots & & \ddots \\ & & \ddots & \ddots & \ddots & & \ddots \\ & & & \ddots & \ddots & & \ddots \\ 0 & & \dots & s_r & 0 & \dots & 0 \\ 0 & & \dots & 0 & 0 & \dots & 0 \\ & & & \ddots & \ddots & & \ddots \\ & & & \ddots & \ddots & & \ddots \\ 0 & & \dots & 0 & 0 & \dots & 0 \end{pmatrix} \quad (3.53)$$

Dabei soll für die Singulärwerte s_1 bis s_r gelten, dass $s_1 \geq s_2 \geq \dots \geq s_r \geq 0$ [11]. Entsprechend dieser Methode wird eine Singulärwertszerlegung, kurz *SVD* der entstandenen Koeffizientenmatrix

durchgeführt. Wir erhalten 3 Matrizen $U \cdot S \cdot V^T$. Durch die Zerlegung sind die diagonaleinträge von S in einer absteigenden Reihenfolge sortiert. Die Spalte von V^T , welche mit dem kleinsten Singulärwert von S korrespondiert, ergibt den Vektor \vec{x} , für den $\| H \cdot x \|$ minimal wird. Somit gleichen die neun Einträge der Homographiematrix gleich der letzten Spalte von V . Das Ergebnis für H hat dann die folgende Form

$$H = \begin{pmatrix} v_{19} & v_{29} & v_{39} \\ v_{49} & v_{59} & v_{69} \\ v_{79} & v_{89} & v_{99} \end{pmatrix} \quad (3.54)$$

Für das Minimalbeispiel mit 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

$$\begin{pmatrix} 1 & 0 & -1 \\ 0 & \sqrt{2} & 0 \\ 1 & 0 & 1 \end{pmatrix} \quad (3.55)$$

Nun werden die Punkte aus Kamera eins s mit Hilfe von H in die Punkte von Kamera zwei überführt und umgekehrt werden die Punkte aus Kamera zwei mit der Inversen H^{-1} in die Punkte von Kamera eins überführt.

$$x' = H \cdot x \rightsquigarrow \begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} & -1 \\ 0 & 0 & \frac{1}{3a} & \frac{1}{2a} \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} 1 & 0 & -1 \\ 0 & \sqrt{2} & 0 \\ 1 & 0 & 1 \end{pmatrix} \cdot \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.56)$$

$$x = H^{-1} \cdot x' \rightsquigarrow \begin{pmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 1 & 1 & 1 \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 & \frac{1}{2} \\ 0 & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2} & 0 & \frac{1}{2} \end{pmatrix} \cdot \begin{pmatrix} -1 & -\frac{1}{3} & -\frac{1}{3} & -1 \\ 0 & 0 & \frac{1}{3a} & \frac{1}{2a} \\ 1 & 1 & 1 & 1 \end{pmatrix} \quad (3.57)$$

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

Bis jetzt wurden die Kameras jeweils nur um ihr Projektionszentrum gedreht, die Frage die jetzt noch im folgenden beantwortet werden soll ist, ob es ebenfalls möglich ist eine Homographiematrix für korrespondierende Punktpaare in einer Ebene aufzustellen, wenn eine Kamera um einen spezifischen Drehpunkt außerhalb der Kamera gedreht wurde. In diesem Fall wären die beiden Projektionszentren (C, β) und (C', β') der Kameras nicht mehr identisch. Um zu veranschaulichen, was genau sich bei einer Drehung um ein Drehpunkt und der Drehung um das Projektionszentrum ändert, wurde eine Simulation geschrieben, welche die Abbildungsunterschiede beider Drehungen aufzeigt. Des Weiteren soll geklärt werden, ob Punkte die bei der Drehung um das Projektionszentrum verdeckt bleiben auch bei einer Drehung um einen außerhalb der Kamera platzierten Drehpunkt verdeckt bleiben. In der Simulation ist der rote Punkt in der Mitte des Quadrats der Drehpunkt, vergleiche Abbildung 3.3.

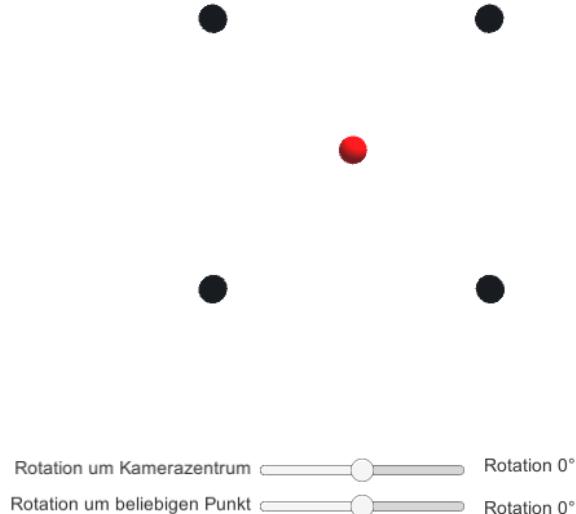


Abbildung 3.5: Objekt im Raum

Für die Simulation der Drehung wurden zwei Schieberegler implementiert mit dem sich die Kamera einmal um das Projektionszentrum, und einmal um den Drehpunkt, welcher der rote Mittelpunkt ist, drehen lässt.

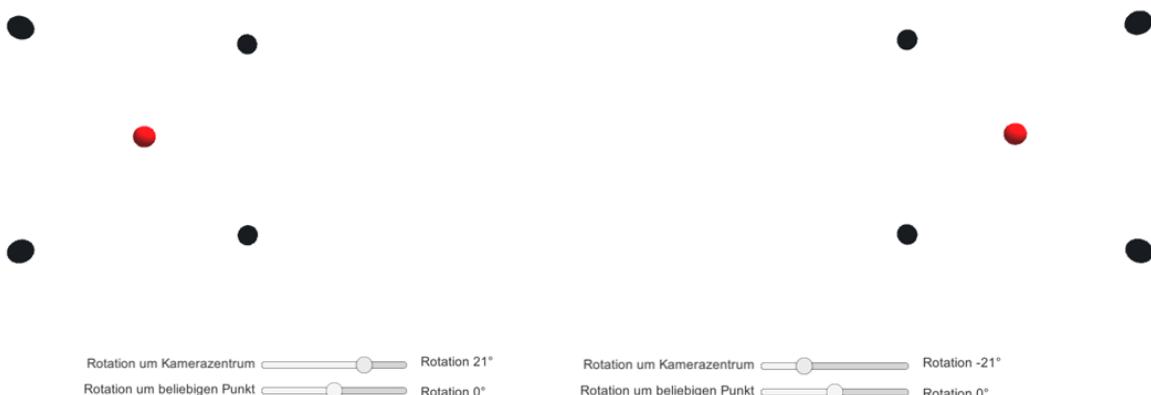


Abbildung 3.6: Drehung um das Projektionszentrum

Abbildung 3.4 zeigt jeweils die entstehenden Bilder, wenn die Kamera um 20° beziehungsweise -20° um das Projektionszentrum gedreht wurde.



Abbildung 3.7: Drehung um einen Drehpunkt. In diesem Beispiel wurde der rote Punkt als Drehpunkt verwendet

Abbildung 3.5 zeigt die entstehenden Bilder, wenn die Kamera um 45° beziehungsweise -45° um den Drehpunkt gedreht wurde. Wie sich zeigt ist hier ein weiterer grüner Punkt zu sehen welcher zu Testzwecken hinter dem roten Punkt platziert wurde sichtbar. Punkte die bei einer Drehung um das Projektionszentrum verdeckt bleiben, werden also bei einer Drehung um einen externen Drehpunkt sichtbar. Abbildung 3.8 veranschaulicht was geometrisch bei beiden Fällen passiert..

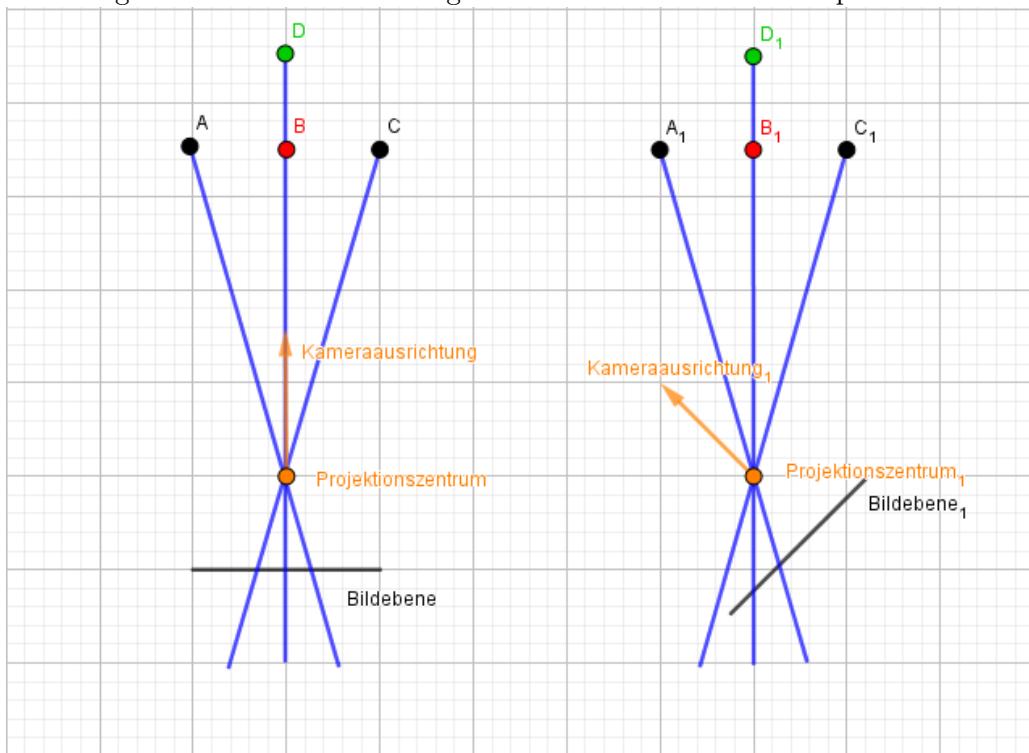


Abbildung 3.8: Strahlengang durch das Projektionszentrum. Auf der Grafik ist erkennbar, dass der grüne Punkt auch nach der Drehung der Kamera um das Projektionszentrum vom roten Punkt verdeckt bleibt

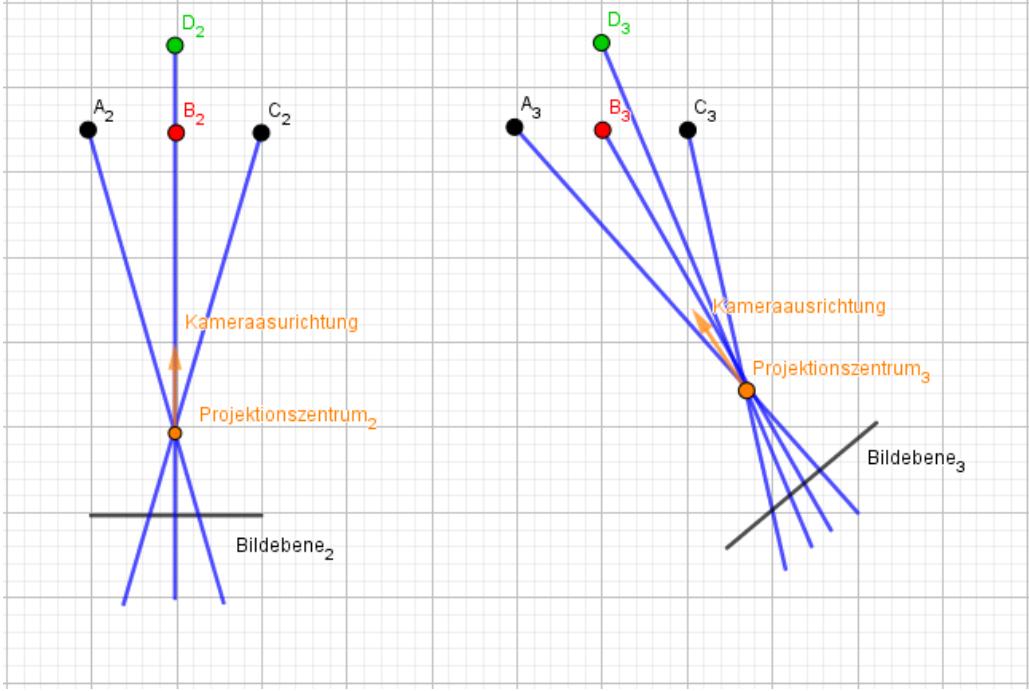


Abbildung 3.9: Strahlengang durch das Projektionszentrum. Auf der Grafik ist erkennbar, dass der grüne Punkt nach der Drehung der Kamera um einen Drehpunkt, welcher in diesem Fall der rote Punkt darstellt, sichtbar wird.

Die entstehenden Abbildungen sind also voneinander verschieden, jedoch liegen die fünf Punkte des Quadrates und dessen Mittelpunkt bei beiden Fällen immer noch in einer Ebene. Die Behauptung die also nun noch zu beweisen gilt ist, dass sich für die jeweiligen Abbildungen beider Drehungen jeweils eine Homographiematrix H aufstellen lässt, welche die Bilder der jeweiligen Kameräns ineinander überführen lässt. Im folgenden wird aufgeführt, wie sich die Homographiematrix im Falle einer Drehung um einen Drehpunkt geometrisch Herleiten lässt. Danach wird wieder ein Beispiel zu diesem Fall aufgezeigt. Zunächst werden wieder die benötigten Koordinatensysteme definiert. Ein Weltkoordinatensystem $(0, \delta)$ mit $\delta = [\vec{d}_1, \vec{d}_2, \vec{d}_3, O]$, ein Kamerakoordinatensystem von Kamera eins (C, β) mit $\beta = [\vec{b}_1, \vec{b}_2, \vec{b}_3, C]$ und ein Kamerakoordinatensystem für Kamera zwei (C', β') mit $\beta' = [\vec{b}'_1, \vec{b}'_2, \vec{b}'_3, C']$. Für die Bildebene I und I' der beiden Kameräns gelten die Koordinatensysteme $(I, \vec{\tau})$ und $(I', \vec{\tau}')$ mit $\vec{\tau} = [\vec{d}_1, \vec{d}_2, \vec{d}_4]$ und $\vec{\tau}' = [\vec{d}_1, \vec{d}_2, \vec{d}'_4]$ mit $d_4 = \vec{CO}$ und $d'_4 = \vec{C'O'}$. Der Punkt in der Objektebene, welcher in die jeweiligen Bildebene projiziert werden soll, wird mit $\vec{M}_\delta = [x \ y \ z \ 1]^T$ bezeichnet. M wird auf die jeweiligen Bildebene der Kameräns projiziert und zu den 2D-Bildebenekoordinaten $m = [u_\tau \ v_\tau \ 1]$ und $m' = [u'_{\tau'} \ v'_{\tau'} \ 1]$ mit Projektionsmatrix P beziehungsweise P' transformiert.

$$\gamma \vec{m}_\beta = P \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = \left[KR| - KRC_\delta \right] \cdot \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = G \cdot \vec{n}_\tau \quad (3.58)$$

$$\gamma' \vec{m}'_\beta = P' \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = \left[K'R'| - K'R'C'_\delta \right] \cdot \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = G' \cdot \vec{n}_{\tau'} \quad (3.59)$$

\vec{n}_τ und $\vec{n}'_{\tau'}$ stellen den Objektpunkt M_δ in Koordinatendarstellung von (I, τ) und (I', τ') dar. Die zwei unterschiedlichen \vec{n}_τ und $\vec{n}'_{\tau'}$ zeigen den Unterschied zu diesem Beispiel von Kameräns mit unterschiedlichen Projektionszentren und dem Beispiel mit identischen Projektionszentren.

Die beiden Projektionsmatrizen P und P' können auf Grund der unterschiedlichen Ergebnisse auf der rechten Seite der Gleichungen 3.58 und 3.59, so wie sie jetzt dastehen nicht gleichgesetzt werden. Um wie in den Gleichungen 3.9 bis 3.13, auf eine Homographiematrix schließen zu können, muss zunächst die Verbindung von \vec{n}_τ und $\vec{n}'_{\tau'}$ aufgezeigt werden. Es muss also erst mal überprüft werden, ob für $[x \ y \ 1]$ mit Hilfe von G und G' gelten kann, so dass gilt $\vec{n}_\tau = \vec{n}'_{\tau'} = [x \ y \ 1]^T$.

$$\vec{n}_\tau = (\vec{M} + \vec{CO})_\tau = \vec{M}_\tau + \vec{d}_{4\tau} = \vec{M}_{(\vec{d}_1, \vec{d}_2, \vec{d}_4)} + \vec{d}_{4(\vec{d}_1, \vec{d}_2, \vec{d}_4)} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3.60)$$

$$\vec{n}'_{\tau'} = (\vec{M} + \vec{C'O})_{\tau'} = \vec{M}_{\tau'} + \vec{d}'_{4\tau'} = \vec{M}_{(\vec{d}_1, \vec{d}_2, \vec{d}_4)} + \vec{d}'_{4(\vec{d}_1, \vec{d}_2, \vec{d}_4)} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (3.61)$$

(3.62)

Für $\lambda = \frac{\gamma'}{\gamma}$ und $C \neq 0$ und $C' \neq 0$, kann somit für H wieder die folgenden Beziehungsgleichung aufgestellt werden. [3]

$$\gamma \vec{m}_\beta = G \cdot \vec{n}_\tau \quad (3.63)$$

$$\gamma' \vec{m}'_{\beta'} = G' \cdot \vec{n}'_{\tau'} \quad (3.64)$$

$$\rightsquigarrow \gamma' \vec{m}'_\beta = G' G^{-1} \cdot \gamma \vec{m}_\beta \quad (3.65)$$

$$\rightsquigarrow \lambda \vec{m}'_{\beta'} = H \vec{m}_\beta \quad (3.66)$$

(NOTSTIONEN DER GRAFIK NOCH ANPASSEN, DEFINITION VON τ nochmal genau ansehen..... ist noch verwirrend....)

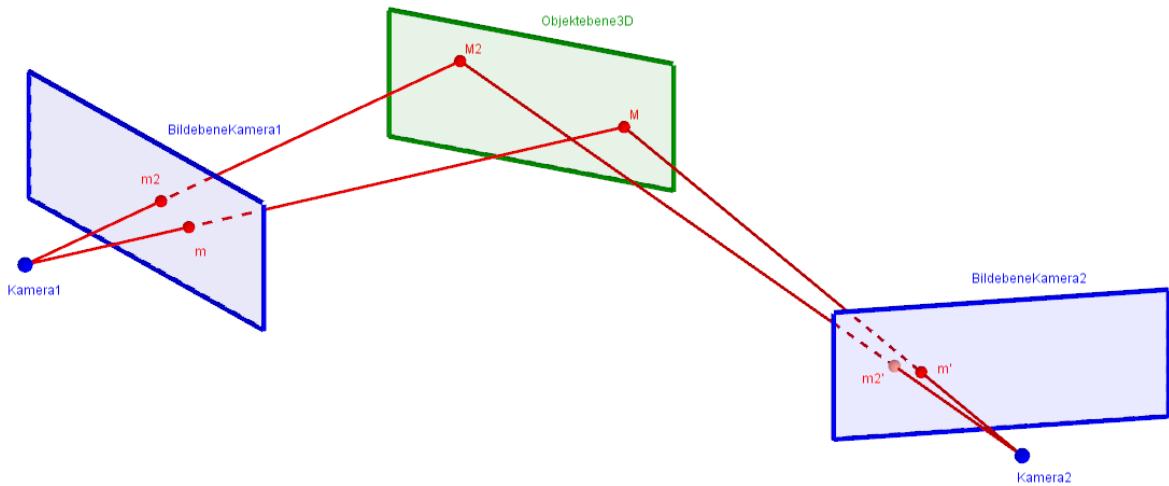


Abbildung 3.10: Veranschaulichung der Homographie bei zwei verschieden translatierten und rotierten Kameras. (\vec{CO}) und $(\vec{C'O})$ sind jeweils die Vektoren welche von den Abbildungspunkten m beziehungsweise m' ausgehen und sich im Objektpunkt M treffen.

Im folgenden wird das breits bekannte Beispiel für die Drehung um das Projektionszentrum, für die Drehung um einen Drehpunkt umgebaut. Danach soll nach dem gleichen Verfahren wie im vorherigen Beispiel aufgezeigt wurde eine Homographiematrix ermittelt werden, welche die Bildebenenkoordinaten der einen Kamera in die der anderen Kamera überführen kann. Die Herangehensweise für dieses Beispiel größtenteils die selbe wie in dem Beispiel zuvor. Der Unterschied liegt lediglich in der Transformation der Punkte in ein Koordinatensystem von der gedrehten Kamera zwei. Diese besteht, im Gegensatz zum vorherigen Beispiel, nicht nur aus hintereinander geschalteten Rotationen, sondern beinhaltet des Weiteren noch zwei Translationen. Die komplette Transformationsmatrix wird, um der vorherigen Notation gerecht zu bleiben, wieder mit R' beziehungsweise R' notiert, auch wenn es sich

hier nicht mehr um eine Reine Rotation handelt. In der Literatur wird die gesamte Transformationsmatrix ebenfalls mit R notiert[4, 3, 10]. R und R' bestehen aus insgesamt drei Transformationsmatrizen $R = \text{Verschiebung}_1 \cdot \text{Rotation} \cdot \text{Verschiebung}_2$. Da die Position des Projektionszentrums von Kamera zwei nicht mehr mit dem von Kamera eins übereinstimmt, muss diese erst mit Hilfe einer Tranformationsmatrix T berechnet werden. $C_\delta = [0 \ 0 \ 0 \ 1]$ sind die Koordinaten von Kamera eins in Weltkoordinaten. Da in diesem Beispiel das Kamerakoordinatensystem von Kamera eins und das Weltkoordinatensystem deckungsgleich sind heißt das, $C_\beta = [0 \ 0 \ 0 \ 1]$. somit ist Matrix R , welche die Transformation der ersten Kamera im Bezug auf das Weltkoordinatensystem beschreibt gleich der Einheitsmatrix.

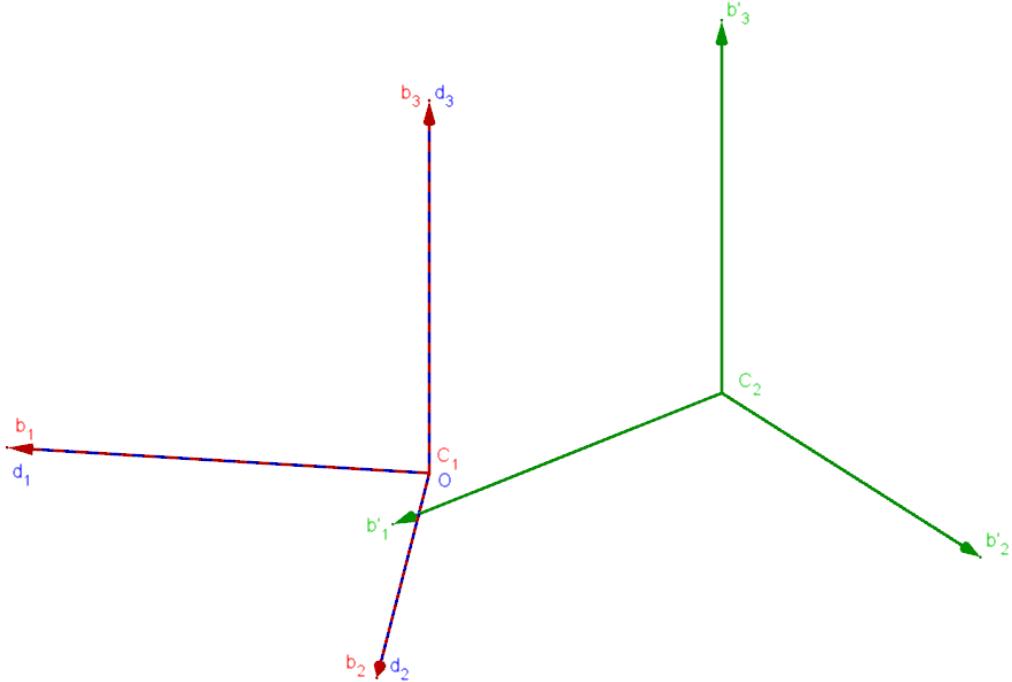


Abbildung 3.11: Weltkoordinatensystem (O, δ) mit $\delta = [\vec{d}_1, \vec{d}_1, \vec{d}_1, O]$ und Kamerakoordinatensysteme $\beta = [\vec{b}_1, \vec{b}_2, \vec{b}_3, C]$ und (C', β') mit $\beta' = [\vec{b}'_1, \vec{b}'_2, \vec{b}'_3, C']$.

Mit C' wird der Ursprung des Koordinatensystems von Kamera zwei bezeichnet. Als Objekte in der Ebene im \mathbb{R}^3 -Raum, werden die selben Punkte $A_\delta, B_\delta, C_\delta, D_\delta$ und E_δ wie im vorherigen Beispiel verwendet.

$$A_\delta = \begin{pmatrix} 0 \\ 0 \\ 2 \\ 1 \end{pmatrix}, B_\delta = \begin{pmatrix} 1 \\ 0 \\ 2 \\ 1 \end{pmatrix}, C_\delta = \begin{pmatrix} 1 \\ 1 \\ 2 \\ 1 \end{pmatrix}, D_\delta = \begin{pmatrix} 0 \\ 1 \\ 2 \\ 1 \end{pmatrix}, E_\delta = \begin{pmatrix} \frac{1}{2} \\ \frac{1}{2} \\ 2 \\ 1 \end{pmatrix} \quad (3.67)$$

Punkt E_δ bildet den Mittelpunkt des Quadrates und wird als Drehpunkt gewählt und mit *PivotPoint* bezeichnet. Als nächstes wird Matrix T aus drei Transformationsmatrizen zusammengestellt. Diese Matrix verschiebt das Projektionszentrum von C' an den gewünschten Ort im Weltkoordinatensystem. Bezeichnet werden die drei Matrizen mit T_1, T_2 und T_3 . T_1 beinhaltet die Verschiebung von des Ursprungs von Kamera eins zum *PivotPoint*, T_2 bildet die Rotationsmatrix, welche den verschobenen Punkt um die gewünschten 45° dreht. Die letzte Matrize T_3 beinhaltet wieder eine Translation, welche den Punkt vom *PivotPoint* zurück verschiebt.

$$T_1 = \begin{bmatrix} 1 & 0 & 0 & -PivotPoint_x \\ 0 & 1 & 0 & -PivotPoint_y \\ 0 & 0 & 1 & -PivotPoint_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -\frac{1}{2} \\ 0 & 1 & 0 & -\frac{1}{2} \\ 0 & 0 & 1 & -2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.68)$$

$$T_2 = \begin{bmatrix} \cos(45^\circ) & 0 & \sin(45^\circ) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(45^\circ) & 0 & \cos(45^\circ) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.69)$$

$$T_3 = \begin{bmatrix} 1 & 0 & 0 & PivotPoint_x \\ 0 & 1 & 0 & PivotPoint_y \\ 0 & 0 & 1 & PivotPoint_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \frac{1}{2} \\ 0 & 1 & 0 & \frac{1}{2} \\ 0 & 0 & 1 & 2 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.70)$$

$$T = T_3 \cdot T_2 \cdot T_1 = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & -1.26777 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0.93934 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.71)$$

$$C'_\delta = T \cdot C_\delta = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & -1.26777 \\ 0 & 1 & 0 & 0 \\ -\frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0.93934 \\ 0 & 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -1.27 \\ 0 \\ 0.94 \\ 1 \end{bmatrix} \quad (3.72)$$

Der Ursprung des Koordinatensystems von Kamera zwei befindet sich, angegeben in Weltkoordinaten, bei $C'_\delta = [-1.26777 \ 0 \ 0.93934 \ 1]^T$. Da nun die C'_δ bekannt ist, können die 3D-Objektpunkte in das Kamerakoordinatensystem von Kamera zwei transformiert werden. Hierzu wird die Matrix R' aufgestellt.

$$R' = \begin{bmatrix} [T_2]^{-1} & -[T_2]^{-1} \cdot V \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.73)$$

aufgestellt werden. V ist der Translationsvektor welcher den Wert von C'_δ bekommt. Da es sich wieder um kartesische Koordinatensysteme handelt gilt wieder $[T_2]^{-1} = [T_2]^T$.

$$-[T_2]^T \cdot C'_\delta = \begin{pmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} \end{pmatrix} \cdot \begin{pmatrix} -1.27 \\ 0 \\ 0.94 \end{pmatrix} = \begin{pmatrix} 1.56 \\ 0 \\ 0.23 \end{pmatrix} \quad (3.74)$$

Für die Transformationsmatrizen R und R' gilt dann jeweils:

$$R = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.75)$$

$$R' = \begin{bmatrix} \frac{1}{\sqrt{2}} & 0 & -\frac{1}{\sqrt{2}} & 1.56 \\ 0 & 1 & 0 & 0 \\ \frac{1}{\sqrt{2}} & 0 & \frac{1}{\sqrt{2}} & 0.23 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3.76)$$

Nachdem R und R' bestimmt sind, müssen die Punkte in Weltkoordinaten noch in die entsprechenden Kamerakoordinatensysteme und mit den Projektionsmatrizen K und K' in deren Bildkoordinatensysteme transformiert werden. Es gilt wieder wie im Beispiel zuvor, dass $K = K'$ ist.

$${}_{K_{c1}} [\pi]_{K_{c1}} = {}_{K_{c2}} [\pi]_{K_{c2}} = \begin{pmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & \zeta & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (3.77)$$

Es entstehen die folgenden beiden Punktematrizen pC für Kamera eins und pC' für Kamera zwei. Die Koordinaten der jeweiligen Punkte $pC = [A_\tau \ B_\tau \ C_\tau \ D_\tau \ E_\tau]$ und $pC' = [A_{\tau'} \ B_{\tau'} \ C_{\tau'} \ D_{\tau'} \ E_{\tau'}]$ aus Sicht der beiden Kameras befinden sich der Reihe nach in den Spalten der Punktematrix.

$$pC = \begin{bmatrix} 0 & \frac{1}{2} & \frac{1}{2} & 0 & \frac{1}{4} \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} & \frac{1}{4} \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (3.78)$$

$$pC' = \begin{bmatrix} 0.09 & 0.36 & 0.36 & 0.09 & \frac{1}{4} \\ 0 & 0 & 0.42 & 0.61 & \frac{1}{4} \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad (3.79)$$

(3.80)

Die Homographiematrix wird durch aufstellen der Koeffizientenmatrix und anschließendes Bestimmen von $H \cdot x = 0$ oder durch findes desjenigen Vektors \vec{x} für den $\|H \cdot x\|$ minimal wird. Da es sich auch hier nicht um einen überbestimmten Fall handelt, kann die homographiematrix entweder durch die Bestimmung des Kerns oder durch anwenden der Singulärwertszerlegung SVD , gewonnen werden. Die resultierende Homographiematrix sieht folgendermaßen aus.

$$H = \begin{bmatrix} 0.43 & 0 & 0.43 \\ 0 & 0.6 & 0 \\ 0.4 & 0 & 0.5 \end{bmatrix} \quad (3.81)$$

Wird H auf die Punkte pC angewandt, so liefert das Ergebnis die Punkte von pC' und wird die Inverse H^{-1} auf die Punkte pC' angewandt, so erhält man die Punkte von pC . Somit wurde bewiesen, dass Homographiematrizen immer zur Transformation von Punkten genutzt werden können, solange sich diese Punkte auf einer Ebene im Raum befinden. Die Definition der Homographie sagt aus, dass sowohl Translationen und Rotationen in der Homographiematrix vorkommen dürfen [8] [9]. Die Drehung um einen Drehpunkt ist nicht weiter als die Hintereinanderschaltung verschiedener Transformationsmatrizen. Das wichtigste Kriterium welches erfüllt sein muss, damit Homographien angewendet werden können ist, dass die Abbildungen der Punkte in allen Kameras auf einer Ebene sich befinden müssen [3].

In der Abbildung 3.7 ist ein grüner Punkt zu sehen welcher sich in Gegensatz zu den anderen Punkten nicht auf der selben Ebene befindet. Dieser Punkt lässt sich nicht mit der errechneten Homographiematrix ineinander überführen. Mit Hilfe von Homographien, können die Positionen und Orientierung der jeweiligen Kameras zueinander ermittelt werden, jedoch nur wenn sich die Szenepunkte auf einer Ebene befinden. Ein Beispiel hierfür wäre zum Beispiel die Aufnahme einer Gebäudefassade aus unterschiedlichen Kamerawinkeln und Positionen [3]. Für eine Szenenrekonstruktion einer kompletten 3D-Szene reichen pure Homographien nicht aus, hierfür muss sich der Epipolargeometrie bedient werden.

3.3 Epipolargeometrie als Grundlage der Stereokalibrierung und Szenenrekonstruktion

Die Epipolargeometrie beschreibt eine intrinsische projektive Geometrie zwischen zwei Bildern[4]. Sie dient insbesondere zur Korrespondenzanalyse von Punkten aus Bildern und zur Gewinnung von 3-D-Informationen. Ohne Kenntnis der Kamerapositionen, kann mit Hilfe der Epipolargeometrie eine einfache Beziehung zwischen korrespondierenden Punkten hergestellt werden. Abbildung 3.12 zeigt, den Aufbau zweier Kameras mit ihren Projektionszentren C und C' , deren Bildebene I und I' , welche vor den Projektionszentren platziert wurden. Die Bildebene können sich auch hinter den Projektionszentren befinden, das hat letztendlich keinen Einfluss auf die geometrischen Beziehungen[4]. Zu den Elementen der Epipolargeometrie gehören zum einen die Epipole e und e' . Betrachtet man die Basislinie zwischen den beiden Projektionszentren, so entsteht der Epipol genau am Schnittpunkt der Verbindungsgeraden mit den jeweiligen Bildebene. Tritt der Fall ein, dass die Basislinie parallel zu einer oder beiden Bildebene ist, so kommt es zu keiner Abbildung des Epipols auf den entsprechenden Bildebene, sondern der Epipol befindet sich in diesem Falle im unendlichen[1]. Das hat zur Folge, dass alle Epipolarlinien, welche durch den Epipol verlaufen, sich zueinander parallel anordnen. Epipolarlinien die Linien, welche durch einen Bildpunkt m oder m' und dem jeweiligen Epipol e oder e' des Bildes verlaufen. Der Korrespondierende Punkt zu m ist m' und die korrespondierende Epipolarlinie l' zu m , ist diejenige Linie, welche durch m' und e' verläuft.

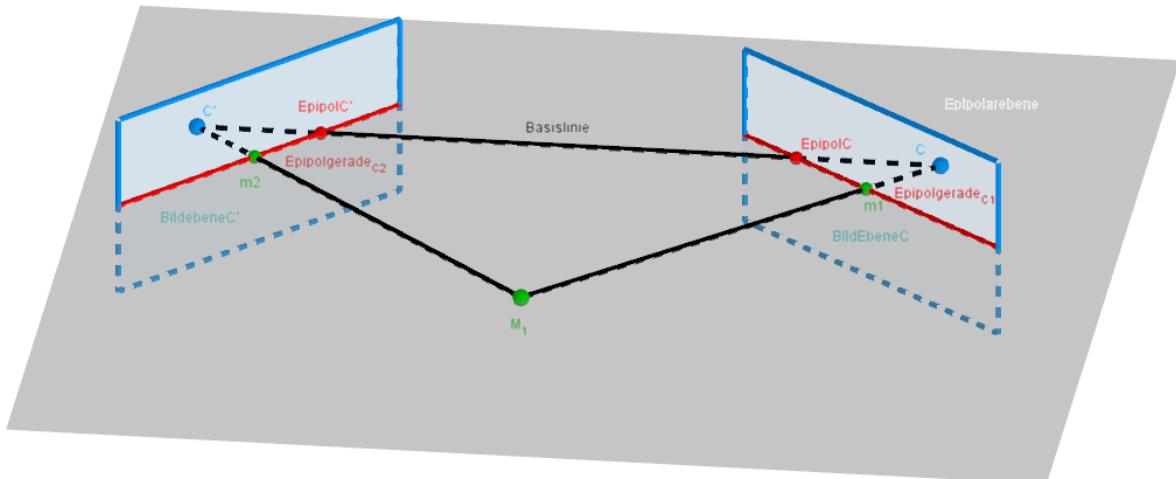


Abbildung 3.12: Grafik zu den geometrischen Eigenschaften der Epipolargeometrie zwischen zweik Bildern. C und C' sind die Projektionszentren zweier Kameras. Beide Kameras besitzen jeweils eine Bildebene. Die Basislinien verbindet die Projektionszentren der Kameras. Der Punkt an welchem die Basislinie die Bildebene schneidet, wird als Epipol bezeichnet. Durch den Epipol verlaufen alle Epipolarlinien des Bildes. M ist der Objektpunkt im 3D-Raum und m_1 und m_2 sind die jeweiligen Abbildungen dieses Punktes auf den Bildebene. Die Verbindungsvektoren zwischen C, C' und M bilden die sogenannte Epipolarebene[3, 4, 1].

Die Epipolarlinie l' , beinhaltet alle möglichen korrespondierenden Punkte zu m . Wenn C auf seiner Bildebene I einen Punkt m abbildet, so erscheint dieser Punkt immer an der selben Stelle auf dem 2D - Bild, egal wie nah der Ursprüngliche 3D-Objektpunkt M an I befindet. Rein theoretisch, könnte der

Original Szenenpunkt M , sich überall auf der Verbindungsgeraden $\bar{M}m$ befinden. Der abgebildete Punkt m wäre auf I immer an der selben Stelle abgebildet. Fährt man nun mit M die Verbindungsgerade $\bar{M}m$ entlang, dann ist m immer an der selben Stelle auf I zu sehen, während der korrespondierende Punkt m' sich entlang der Epipolarlinie bewegen würde. Die Epipolargeometrie beschreibt also eine Beziehung zwischen einem Bildpunkt m und dessen korrespondierender Epipolarlinie l' , welche wiederum alle möglichen zu m korrespondierenden Punkte m' beinhaltet[4, 5, 1].

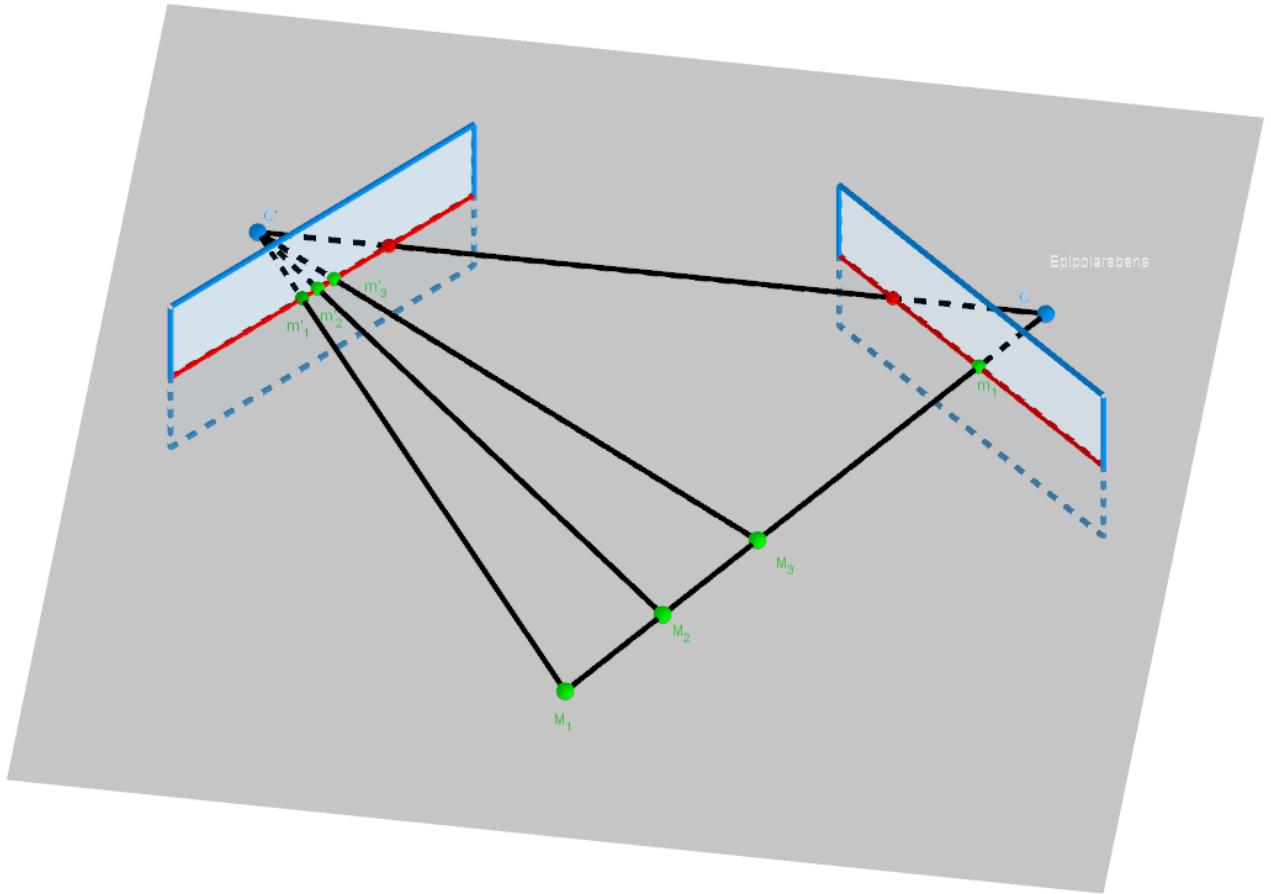


Abbildung 3.13: Die Objektpunkte M_1, M_2 und M_3 werden in I' als m'_1, m'_2 und m'_3 abgebildet, während sie in I immer den selben Bildpunkt m_1 ergeben.

Die Epipolargeometrie lässt sich, ähnlich wie eine Homographiematrix, in einer 3×3 -Matrix zusammenfassen. Diese ist jedoch singulär und besitzt somit nicht wie die Homographiematrix Rang 3 sondern Rang 2. Je nachdem ob ein kalibriertes oder unkalibriertes System vorliegen hat, handelt es sich entweder um die sogenannten Fundamental Matrix F oder die Essentielle Matrix E [3, 4, 12, 5, 1]. Von der Fundamentalmatrix F ist dann die Rede, wenn die intrinsischen Kameraparameter nicht bekannt sind, sprich wenn das System unkalibriert ist. In diesem Fall wird mit Bildpixelkoordinaten gearbeitet. Sind die intrinsischen Parameter bekannt, so wird F zur essentiellen Matrix E und es wird mit sogenannten normalisierten Bildkoordinaten gearbeitet[12]. Was genau die unterschiedlichen Koordinaten ausmacht, wird in Kapitel (KAPITELLINK) genauer erklärt. Mathematisch sagen die Matrizen F und E in Verbindung mit den korrespondierenden Punkten aus, ob für einen Bildpunkt m in einer Kamera, dessen korrespondierender Bildpunkt m' in der anderen Kamera auf der korrespondierenden Epipolarlinie liegt. Das heißt, wenn $m'Fm = 0$ oder $\hat{m}'E\hat{m} = 0$, dann ist der sogenannte *epipolar-constraint* erfüllt. Der *epipolar-constraint* gibt somit Aufschluss darüber, ob m' ein möglicher korrespondierender Punkt von m ist. Dies ist nämlich genau dann der Fall, wenn $m'Fm = 0$ oder $\hat{m}'E\hat{m} = 0$ sind[4, 5]. Ist der *epipolar-constraint* erfüllt, so wird gleichzeitig der Suchaufwand nach

weiteren Korrespondenzen reduziert, da somit nur noch eine eindimensionale Suche, entlang der Epipolarlinie, anstatt einer zweidimensionalen durchgeführt werden muss. Dieser neue *constraint* wird auch als *coplanarity constraint* oder Koplanaritätsbeschränkung bezeichnet. Dieser entsteht, da die Projektionszentren der Kameras und die korrespondierenden Bildpunkte auf ein und der selben Ebene liegen müssen [5]. Die Epipolargeometrie und die in ihr beinhalteten *constraints*, helfen bei der 3D-Szenenrekonstruktion. Szenenrekonstruktion ist dann möglich, wenn in einer Stereoaufnahme in beiden Bildern die zueinander gehörenden Bildpunkte lokalisiert wurden. Wird also eine Szene mit zwei Kameras aufgenommen, so liegen während der Aufnahme die aufgenommenen Objektpunkte, das Projektionszentrum und der zur Kamera gehörende Bildpunkt auf einer Linie. Wurde eine Objektpunkt nun zweimal aus verschiedenen Winkeln und/ oder Position aufgenommen, lassen sich nachdem die extrinsischen Parameter der Kameras ermittelt wurden, die Schnittpunkt der jeweiligen Linien aus Kamera eins und Kamera zwei berechnen. Diese Schnittpunkte ergeben den ursprünglichen Objektpunkt. Die Szene ist somit rekonstruiert[3, 1, 4].

3.4 Geometrische Erläuterung der Fundamentalmatrix und der Essentiellen Matrix

Nachdem die Theorie der geometrischen Hintergründe der Epipolargeometrie, bei der Stereokalibrierung und Szenenrekonstruktion, erläutert wurden, wird nun der mathematische Hintergrund genauer aufgezeigt. Vor allem soll auf die Herleitung der neu eingeführten Fundamental Matrix F und der essentiellen Matrix E eingegangen werden. Diese spielen nämlich eine entscheidende Rolle bei der Rekonstruktion der Kamerapose und der Szenenrekonstruktion[3, 4]. F und E bilden jeweils eine singuläre 3x3-Matrix, welche die Geometrie zwischen den Bildpunkten m_τ und m'_τ auf I und I' und dem Objektpunkt M_δ im Raum beschreibt. Die Vektoren $\vec{CM} = (\vec{M}_\delta - \vec{C}_\delta)$, $\vec{C'M} = (\vec{M}_\delta - \vec{C}'_\delta)$ und $\vec{CC'} = (\vec{C}'_\delta - \vec{C}_\delta)$ bilden das in Abbildung 3.12 sichtbare schwarze Dreieck. F und E fassen dieses Dreieck in ihren Matrizen zusammen. Um das ganze mathematisch zu erklären, wird ein Stereokamerabau definiert. Ein Objektpunkt M_δ in Weltkoordinaten(O, δ) wird von zwei Kameras C mit (C, β) und C' mit (C', β') aufgenommen und auf deren Bildebenen I und I' als m_β und m'_β abgebildet. C und C' besitzen jeweils eine eigene Projektionsmatrix P und P' . Anzumerken ist, dass die folgende Herleitung nach [3] aufgestellt wurde.

$$P = \begin{bmatrix} KR | - KR \vec{C}_\delta \end{bmatrix} \quad (3.82)$$

$$P' = \begin{bmatrix} K'R' | - K'R' \vec{C}'_\delta \end{bmatrix} \quad (3.83)$$

M wird mit P und P' auf die Bildebenen I und I' mit den jeweiligen Koordinatensystemen $I = (I, \tau)$ und $I' = (I', \tau')$ projiziert. Wichtig anzumerken, auch für den späteren Aufbau mit zwei Kameras unterschiedlicher Auflösung, ist, dass es sich bei den Koordinatensystemen von I und I' nicht um identische handeln muss.[3] Es entstehen die Bildpunkte γm_τ und $\gamma' m'_{\tau'}$ mit $\gamma \geq 0$ und $\gamma' \geq 0$. (gamma erklären)

$$\gamma \vec{m}_\tau = P \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} \quad (3.84)$$

$$\gamma \vec{m}_\tau = \begin{bmatrix} KR | - KR \vec{C}_\delta \end{bmatrix} \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} \quad (3.85)$$

$$\gamma' \vec{m}'_{\tau'} = P' \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} \quad (3.86)$$

$$\gamma' \vec{m}'_{\tau'} = \begin{bmatrix} K'R' | - K'R' \vec{C}'_\delta \end{bmatrix} \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} \quad (3.87)$$

$$(3.88)$$

$-KR\vec{C}_\delta$ und $-K'R'\vec{C}'_\delta$ verrechnet mit M sind gleich den Vektorausdrücken $(\vec{M}_\delta - \vec{C}_\delta)$ und $(\vec{M}_\delta - \vec{C}'_\delta)$, welche die Verbindungsgeraden der beiden Projektionszentren mit dem Objektpunkt M im Raum beschreiben.

$$\gamma \vec{m}_\tau = KR(\vec{M} - \vec{C}_\delta) \quad (3.89)$$

$$\gamma' \vec{m}'_\tau = K'R'(\vec{M} - \vec{C}'_\delta) \quad (3.90)$$

Gleichungen 3.89 und 3.90 werden nach $(\vec{M} - \vec{C}_\delta)$ und $(\vec{M} - \vec{C}'_\delta)$ aufgelöst.

$$\gamma R^T K^{-1} \vec{m}_\tau = (\vec{M} - \vec{C}_\delta) \quad (3.91)$$

$$\gamma R'^T K'^{-1} \vec{m}'_\tau = (\vec{M} - \vec{C}'_\delta) \quad (3.92)$$

Wie bereits erwähnt ergibt sich aus den Vektoren $(\vec{M}_\delta - \vec{C}_\delta)$, $(\vec{M}_\delta - \vec{C}'_\delta)$ und $(\vec{C}'_\delta - \vec{C}_\delta)$ das Dreieck aus Abbildung 3.12. Für das Dreieck kann, aus den drei Vektoren, die folgende Gleichung aufgestellt werden.

$$(\vec{C}'_\delta - \vec{C}_\delta) = (\vec{M}_\delta - \vec{C}_\delta) - (\vec{M}_\delta - \vec{C}'_\delta) \quad (3.93)$$

$(\vec{M} - \vec{C}_\delta)$ und $(\vec{M} - \vec{C}'_\delta)$ können durch die Ausdrücke in den Gleichungen 3.91 und 3.92 ersetzt werden.

$$(\vec{C}'_\delta - \vec{C}_\delta) = \gamma R^T K^{-1} \vec{m}_\tau - \gamma R'^T K'^{-1} \vec{m}'_\tau \quad (3.94)$$

Es gilt $\gamma \geq 0$ und $\gamma' \geq 0$, sie stehen für die Tiefe von m und m' und können mit Hilfe des Kreuzproduktes eliminiert werden[3]. Zunächst wird $(\vec{C}'_\delta - \vec{C}_\delta)$ auf die rechte Seite gebracht, so dass die Gleichung nach Null aufgelöst wird.

$$[\vec{C}'_\delta - \vec{C}_\delta] \times \gamma R^T K^{-1} \vec{m}_\tau - [\vec{C}'_\delta - \vec{C}_\delta] \times \gamma' R'^T K'^{-1} \vec{m}'_\tau = 0 \quad (3.95)$$

Gleichung 3.95 wird von links mit $\gamma' \vec{x}'_\tau^T K'^{-T} R'$. Somit kann eine der beiden Schiefsymmetrischen Matrizen aus der Gleichung eliminiert werden.

$$\gamma' \vec{m}'_\tau K'^{-T} R' [\vec{C}'_\delta - \vec{C}_\delta] \times \gamma R^T K^{-1} \vec{m}_\tau = 0 \quad (3.96)$$

Da $\gamma \geq 0$ und $\gamma' \geq 0$, kann für Gleichung 3.96 auch folgendes geschrieben werden.

$$\vec{m}'_\tau K'^{-T} R' [\vec{C}'_\delta - \vec{C}_\delta] \times R^T K^{-1} \vec{m}_\tau = 0 \quad (3.97)$$

Aus Gleichung 3.97 können nun die Matrizen F und E ausgelesen werden. Bei E handelt es sich um einen Kalibrierten Fall, dass bedeutet dass sowohl K als auch K' bekannt sind und die normalisierten Bildkoordinaten \vec{m} und \vec{m}' durch Multiplikation mit K und K' entstehen. E selbst fasst die Schiefsymmetrische Matrix $[\vec{C}'_\delta - \vec{C}_\delta] \times$ und die beiden Transformationsmatrizen R und R' zusammen.

$$\vec{m}'_\tau^T K'^{-T} E K^{-1} \vec{m}_\tau = 0 \quad (3.98)$$

$$\vec{m}_\tau^T E \vec{m}'_\tau = 0 \quad (3.99)$$

Matrix E wird zu F , wenn es sich um einen unkalibrierten Fall handelt. Unkalibriert bedeutet, dass K und K' nicht bekannt sind, die Informationen zu K und K' in F befinden. Werden K und K' zu E' multipliziert, wird E zu F .

$$\vec{m}'_{\tau'}^T K'^{-T} E K^{-1} \vec{m}_\tau = 0 \quad (3.100)$$

$$\vec{m}'_{\tau'}^T F \vec{m}_\tau = 0 \quad (3.101)$$

F und E , fassen die komplette Epipolarageometrie, sprich externe und interne Parameter, sowie die geometrische Beziehung der jeweiligen Bildpunkte zu den 3-D Objektpunkten in einer 3x3-Matrix zusammen. Für F und E gibt es nicht nur eine Lösung. Werden F oder E Beispielsweise über den *eight-Point-Algorithm* ermittelt, so sind die entstehenden 3x3-Matrixen und jedes vielfache von diesen gültige Lösungen für F und E [4, 13]. **Noch herausfinden ob das mit den Tiefen γ und γ' zusammenhängt!!.** Mit F und E kann wie bereits nachgeprüft werden, ob der *epipolar-constraint* $m'^T F m = 0$ oder $\hat{m}'^T E \hat{m} = 0$ zwischen zwei Bildpunkten gilt. Des weiteren können Epipoles e und e' und Epipolarlinien l und l' ausfindig gemacht werden, sobald E oder F bekannt ist[4, 3, 13, 1]. Um die zu m oder m' korrespondierende Epipolarlinie l' oder l zu berechnen gilt:

$$l' = Fm \quad (3.102)$$

$$l = F^T m' \quad (3.103)$$

Um die Epipoles e und e' zu berechnen die Gleichungen 3.103 und 3.104 erfüllt sein. Für e reicht es also den rechten Nullraum von F zu bestimmen und für e' muss dementsprechend der linke Nullraum von F gefunden werden.

$$Fe = 0 \quad (3.104)$$

$$F^T e' = 0 \quad (3.105)$$

Die Matrizen F und E sind die ausschlaggebenden Elemente, wenn es um die Rekonstruktion der Kameraorientierungen und der Rekonstruktion der Szene geht. In beiden folgenden Kapitel werden zwei Beispiele zur Findung der externen Kameraparameter und der Szenenrekonstruktion aufgezeigt. Beim ersten Beispiel handelt es sich um ein Minimalbeispiel mit synthetisch erzeugten reinen Daten, um die theoretische Funktionalität des Algorithmus zu beweisen. Im zweiten Beispiel, wird der Algorithmus, mit einigen Anpassungen an die Realverhältnisse, auf Stereobildpaare, aufgenommen von zwei verschiedenen Kameras, angewandt. Die implementierten Algorithmen ermitteln aus einem Satz korrespondierender Bildpunkte die Fundamental Matrix und die essentielle Matrix mit Hilfe des sogenannten *8-Point-Algorithm*, Im Anschluss werden dann die externen Kameraparameter ermittelt und die Szene mit einem Triangulationsverfahren rekonstruiert.

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

Um den Mathematischen Vorgänge der stereoskopischen Szenenrekonstruktion zu veranschaulichen und besser nachvollziehen zu können, wurde ein Minimalszenario erstellt. Mit Hilfe dieses Minimalszenarios, können Theorien besser getestet und bestimmte Situationen simuliert werden. Des Weiteren kann es von Nutzen sein, wenn es darum geht, Fehler in Realbeispielen nachzustellen oder eine Lösung für diese zu finden. Für dieses Minimalszenario wurde ein Objekt, in diesem Falle ein Quader, in ein zuvor definiertes Weltkoordinatensystem (O, δ) mit $\delta = (d_1, d_2, d_3)$ platziert. Außerdem wurden zwei Kameras (C, β) mit $\beta = (b_1, b_2, b_3)$ und (C', β') mit $\beta' = (b'_1, b'_2, b'_3)$ definiert und in (O, δ) platziert. Kamera eins (C, β) ist Deckungsgleich mit (O, δ) . Kamera zwei (C', β') wurde von C in positive d_1 -Richtung, verschoben und um einen Winkel α zu C um die eigene b'_3 -Achse rotiert. Die verwendeten kartesischen Koordinatensysteme sind in diesem Minimalbeispiel alle rechtshändig orientiert. Die äußeren und inneren Kameraparameter wurden für den Aufbau der Szene festgelegt. Dies hat den positiven Effekt, dass somit die späteren Ergebnisse besser validiert werden können. In den Abbildung 4.1 bis 4.3 wird der Aufbau noch einmal genauer veranschaulicht.

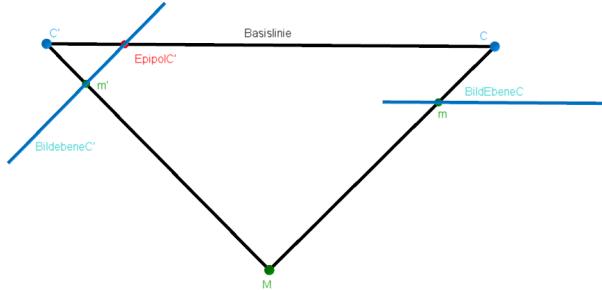


Abbildung 4.1: vereinfachte Top-Down-Ansicht des Szenenaufbaus des Minimalbeispiele

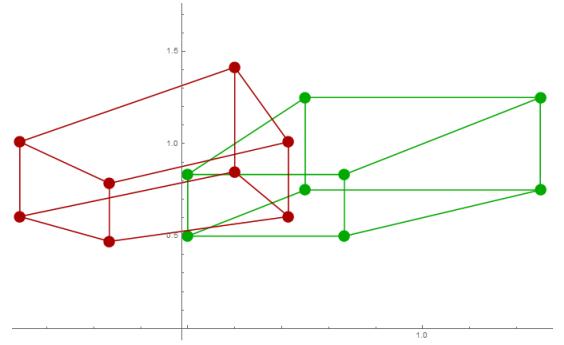


Abbildung 4.2: In Grün ist die Abbildung auf der Bildebenen I von C und in rot ist die Abbildung auf der Bildebenen I' von C'

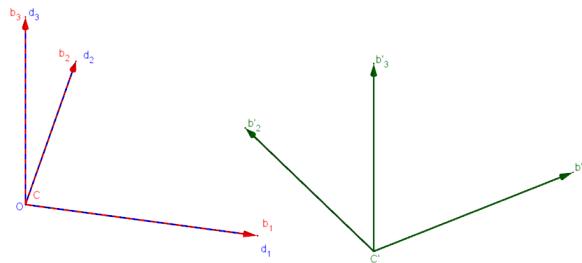


Abbildung 4.3: In Blau und Rot sind jeweils das Welt- und Kamerakoordinatensystem von Kamera eins zu sehen. In grün ist das gedrehte Koordinatensystem von Kamera 2 zu sehen.

4.1 Vorgehen: Projektion eines Quaders in zwei verschiedene transformierte Kameras

Für die Stereokamerakalibrierung wird C' relativ zur zu C um einen Vektor \vec{V}' verschoben und anschließend um einen Winkel α um die b'_3 Achse gedreht. Für die Rotation um b'_3 wird eine Drehmatrix D' aufgestellt.

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

Um die Transformationsmatrix R' , welche die Rotation und die Translation von C' beinhaltet, zu erhalten werden D' und \vec{V}' miteinander verrechnet.

$$\vec{V}' = \begin{pmatrix} 1 & 0 & 0 & -v'_1 \\ 0 & 1 & 0 & -v'_2 \\ 0 & 0 & 1 & -v'_3 \end{pmatrix} \quad (4.2)$$

$$R' = D'^T \cdot \vec{V}' \quad (4.3)$$

$$R' = \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -v'_1 \\ 0 & 1 & 0 & -v'_2 \\ 0 & 0 & 1 & -v'_3 \end{pmatrix} \quad (4.4)$$

$$R' = \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) & -v'_1 \cos(\alpha) + v'_3 \sin(\alpha) \\ 0 & 1 & 0 & -v'_2 \\ \sin(\alpha) & 0 & \cos(\alpha) & -v'_1 \sin(\alpha) - v'_3 \cos(\alpha) \end{pmatrix} \quad (4.5)$$

Die entstandene Matrix R' beschreibt die Transformation C' und somit auch die Transformation von Punkten des Koordinatensystems (C, β) in (C', β') . Da $(C, \beta) = (O, \delta)$ ist, beinhaltet die Transformationsmatrix R für C weder eine Translation noch eine Rotation.

$$V = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (4.6)$$

$$R = [I|V] \quad (4.7)$$

$$R = \begin{bmatrix} 1 & 0 & 0 & v_1 \\ 0 & 1 & 0 & v_2 \\ 0 & 0 & 1 & v_3 \end{bmatrix} \quad (4.8)$$

4.2 Berechnung der Projektionsmatrizen

Die Eckpunkte des Quaders $A_\delta, B_\delta, C_\delta, D_\delta, A'_\delta, B'_\delta, C'_\delta, D'_\delta$ sind bekannt. Neben den Eckpunkten des Quaders wird noch ein neunter Punkt E_δ außerhalb des Quaders platziert und zwar so, dass es zu keinen linearen Abhängigkeiten zwischen E_δ und den anderen Punkten kommt. So kann vermieden werden, dass die später aufgestellt Koeffizientenmatrix, zum berechnen der Fundamentalmatrix, einen Rang kleiner als acht bekommt und somit zwei linear unabhängige Lösungen ausgibt[4]. Im Unterkapitel (KAPITELLINK FUNDAMENTALMATRIX) wird nochmal genauer drauf eingegangen, was das für F bedeutet. Fürs erste wird festgelegt, dass insgesamt neun Punkte sich in der Szene befinden. Um diese Punkte auf die Bildebenden (I, τ) und (I', τ) zu projizieren, müssen neben den Transformationsmatrizen R und R' noch die Kameramatrizen K und K' festgelegt werden.

$$K = \begin{bmatrix} \zeta_C & 0 & 0 & 0 \\ 0 & \zeta_C & 0 & 0 \\ 0 & 0 & \zeta_C & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.9)$$

$$K' = \begin{bmatrix} \zeta_{C'} & 0 & 0 & 0 \\ 0 & \zeta_{C'} & 0 & 0 \\ 0 & 0 & \zeta_{C'} & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.10)$$

Sind R, R', K und K' bekannt, können die Projektionsmatrizen P und P' berechnet werden. Vorher werden R und R' noch projektiv erweitert.

$$R = \begin{bmatrix} 1 & 0 & 0 & v_1 \\ 0 & 1 & 0 & v_2 \\ 0 & 0 & 1 & v_3 \end{bmatrix} \quad (4.11)$$

$$P = K \cdot R \quad (4.12)$$

$$P = \begin{bmatrix} \zeta_C & 0 & 0 & 0 \\ 0 & \zeta_C & 0 & 0 \\ 0 & 0 & \zeta_C & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.13)$$

$$P' = K' \cdot R' \quad (4.14)$$

$$P' = \begin{bmatrix} \zeta_{C'} \cos(\alpha) & 0 & \zeta_{C'} \sin(\alpha) & -\zeta_{C'}(v'_1 \cos(\alpha) + v'_3 \sin(\alpha)) & 0 \\ 0 & 1 & 0 & \zeta_{C'} - v'_2 & 0 \\ \zeta_{C'} \sin(\alpha) & 0 & \zeta_{C'} \cos(\alpha) & -\zeta_{C'}(v'_1 \sin(\alpha) + v'_3 \cos(\alpha)) & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \quad (4.15)$$

4.3 Transformation der Weltpunkte in Koordinaten der Koordinatensysteme von beiden Kameras

Die 3D-Punkte $A_\delta, B_\delta, C_\delta, D_\delta, A'_\delta, B'_\delta, C'_\delta, D'_\delta, E_\delta$, werden um eine Homogene Komponenten erweitert und mit den Projektionsmatrizen P und P' zu, auf die Bildebenen I und I' projizierten, Punkten der Kamerakoordinatensysteme (C, β) und C', β' transformiert.

$$Q_\beta = P \cdot \begin{bmatrix} \begin{pmatrix} A_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} B_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} C_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} D_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} A'_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} B'_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} C'_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} D'_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} E_\delta \\ 1 \end{pmatrix} \end{bmatrix} \quad (4.16)$$

$$Q_{\beta'} = P' \cdot \begin{bmatrix} \begin{pmatrix} A_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} B_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} C_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} D_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} A'_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} B'_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} C'_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} D'_\delta \\ 1 \end{pmatrix} & \begin{pmatrix} E_\delta \\ 1 \end{pmatrix} \end{bmatrix} \quad (4.17)$$

Die entstanden Bildebenenkoordinaten zur Basis der Kamerakoordinatensysteme müssen dann wieder auf eine homogene Form gebracht werden, indem sie durch ihre jeweils letzte Komponenten dividiert werden.

$$A_\beta = \begin{pmatrix} b_1 \\ b_3 \\ b_2 \\ \gamma \end{pmatrix} \rightsquigarrow A_\beta = \begin{pmatrix} \frac{b_1}{\gamma} \\ \frac{b_3}{\gamma} \\ \frac{b_2}{\gamma} \\ \frac{\gamma}{\gamma} \end{pmatrix} \rightsquigarrow A_\beta = \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \\ 1 \end{pmatrix} \quad (4.18)$$

$$\simeq \begin{pmatrix} t_1 \\ t_2 \\ \zeta \\ 1 \end{pmatrix} \quad (4.19)$$

Der in Gleichung 4.18 verwendete Kamerapunkt A_β liegt direkt auf der Bildebene I , was durch die Projektionsmatrix P bedingt wurde. Die Tiefenkomponente entspricht deshalb nach der homogenisierung des Punktes dem Wert ζ , welcher den Abstand \overline{IC} beschreibt. Um die 3D-Kamerapunkte in 2D-Bildebenepunkte zur Basis (I, τ) mit $\tau = (t_1, t_2)$ und für C' entsprechend (I', τ') mit $\tau' = (t'_1, t'_2)$ umzuwandeln, muss lediglich der Tiefenwert ζ entfernt und durch die Homogene Komponenten ersetzt werden. In Abbildung 4.4, sind die entstehenden Abbildilder auf die jeweiligen Bildebene in 2D-Bildebenekoordinaten (I, τ) und (I', τ') zu sehen.

$$A_\beta = \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \\ 1 \end{pmatrix} \simeq \begin{pmatrix} t_1 \\ t_2 \\ \zeta \\ 1 \end{pmatrix} \quad (4.20)$$

$$\rightsquigarrow A_\tau = \begin{pmatrix} t_1 \\ t_2 \\ 1 \end{pmatrix} \quad (4.21)$$

(4.22)

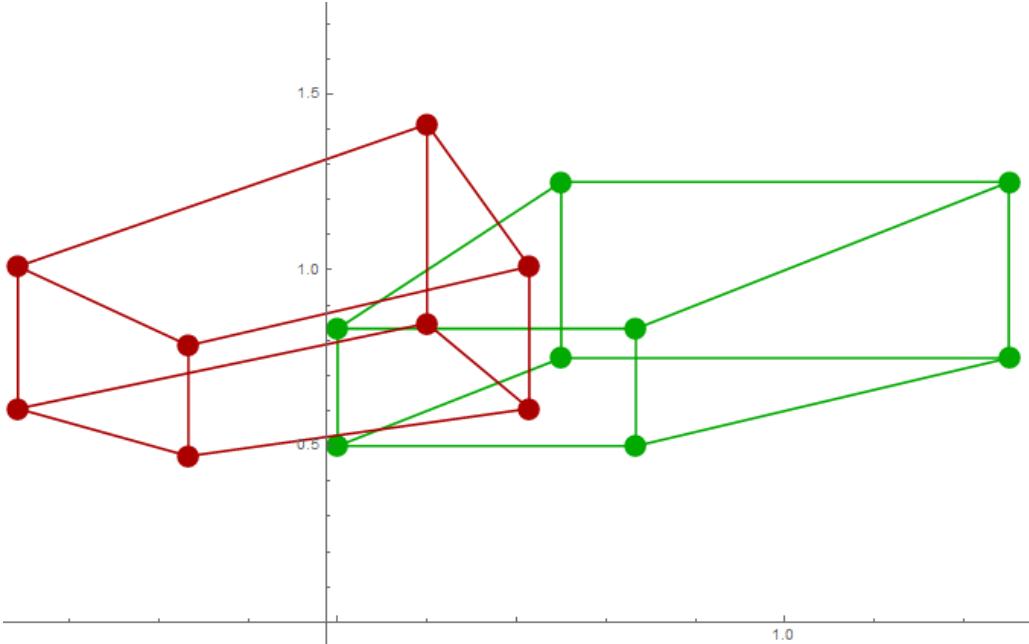


Abbildung 4.4: Grün zeigt den Quader welcher auf I von C abgebildet wird. Das größere Quadrat sind die vorderen Punkte A, B, C, D , das kleinere Quadrat sind die hinteren Punkte A', B', C', D' . Der Punkt E ist weiter weg von den Abbildungen und deshalb auf dieser Abbildung momentan nicht zu sehen. Rot zeigt denselben Quader auf I' von C' abgebildet.

4.4 Umrechnung von Bildebenenkoordinaten in Sensorkoordinaten

Für die Umrechnung der Bildebenenkoordinaten $A_\tau, B_\tau, C_\tau, D_\tau, A'_\tau, B'_\tau, C'_\tau, D'_\tau, E_\tau$ und $A_{\tau'}, B_{\tau'}, C_{\tau'}, D_{\tau'}, A'_{\tau'}, B'_{\tau'}, C'_{\tau'}, D'_{\tau'}, E_{\tau'}$ in Sensorkoordinaten (S, σ) mit $\sigma = (\vec{u}, \vec{v})$, muss der sogenannte Pixelpitch des Sensors bekannt sein. Für das Minimalbeispiel wird ein PixelPitch von 1 angenommen. Die Bildebenenkoordinaten werden so 1:1 in die Sensorkoordinaten umgesetzt. In Realbeispielen ist dies aber eher selten der Fall. Die Sensorkoordinaten sind im normalfall in Pixeleinheiten gegeben und werden für die Berechnung der Fundamentalmatrix benötigt.

$$\sigma = (\vec{u}, \vec{v}) \quad (4.23)$$

$$\vec{u} = u_1 t_1 + u_2 t_2 \quad (4.24)$$

$$\vec{v} = v_1 t_1 + v_2 t_2 \quad (4.25)$$

$$S_\sigma = I_\tau + p_{x\tau} + p_{y\tau} \quad (4.26)$$

$$(\vec{u}, \vec{v}, S_\sigma) = (t_1, t_2, I_\tau) \cdot \begin{bmatrix} u_1 & u_2 & z_1 \\ v_1 & v_2 & z_2 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.27)$$

$$M = \begin{bmatrix} u_1 & u_2 \\ v_1 & v_2 \end{bmatrix} \quad (4.28)$$

Mit dem Berechnen der Sensorkoordinaten ist der Aufbau der Szene vollendet. Das Ergebnis sind zwei Bilder des Quaders von zwei verschiedenen Kameras mit momentan noch gleicher Auflösung. In den nächsten Schritten soll nun anhand der Bildpunkte eine Rekonstruktion der externen Kameraparameter und eine Rekonstruktion der synthetischen Szene folgen. Die intrinsischen Parameter werden in dieser Arbeit immer als bereits bekannt festgelegt. (Hier vllt noch erwähnen, dass die calibrierung der inneren Kameraparameter in einer anderen Arbeit behandelt wurde, außerdem kann hier auch eine getrennt kalibrierung beider kameras durchgeführt werden)

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

Nachdem nun die Bildkoordinaten in Pixel des Quaders auf den Bildern der beiden Kameras bekannt sind, kann nun die eigentliche Rekonstruktion der externen Kameraparameter und die 3D-Szenenrekonstruktion beginnen. Der erste Schritt beinhaltet die Ermittlung der Fundamentalmatrix F mit den korrespondierenden Bildpunkten der beiden Kamerabilder mit Hilfe des sogenannten *8-Point-Algorithm*. Der *8-Point-Algorithm* ist eine lineare Technik die angewandt wird, um die essentielle Matrix und die Fundamentalmatrix aus $n \geq 8$ Punkten zu schätzen [5, 4]. Der Algorithmus benötigt $n \geq 8$ Punkte, um ein valides eindeutiges Ergebnis zu liefern [4, 14]. Es besteht noch die Möglichkeit den *7-Point-Algorithm* anzuwenden, welcher nach dem selben Prinzip wie sein Verwandter *8-Point-Algorithm* verfährt, jedoch kein eindeutiges Ergebnis liefert. Mit sieben Punkten bekommen wir als Lösung zwei linear unabhängige Kerne als Lösung der zuvor aufgestellten Koeffizientenmatrix mit welchen für ein eindeutiges Ergebnis noch weiter verfahren werden muss [4, 14]. Die aufgestellte Koeffizientenmatrix, hat im Fall des *7-Point-Algorithm* nämlich nur den Rang 7 [4]. In der Schätzung von F wurde in diesem Minimalbeispiel und auch später in Realbeispiel der *8-Point-Algorithm* verwendet. Im Falle des Minimalbeispiels, wurde mit einem zusätzlichen neunten Punkt auch noch dafür gesorgt, dass die aufgestellte Koeffizientenmatrix auch den Rang 8 besitzt. Somit kann, wie bei der Bestimmung der Homographie, einfach der Nullraum der Koeffizientenmatrix bestimmt werden, welcher die Einträge der 3×3 -Matrix von F liefert [4]. Besitzt die Koeffizientenmatrix einen Rang größer 8, so wird auch hier mit einem *least-square*-Verfahren, mit Hilfe der Singulärwertszerlegung, angewendet werden, so dass $\|Koeffizientenmatrix \cdot f\|$ minimal wird. f ist der Singulärvektor, welcher mit dem kleinsten Singulärwert korrespondiert [4]. Die genaue Abfolge des Verfahrens, wird im Kapitel (KAPITELLINK ZU F IN REALBEISPIEL) genauer aufgezeigt. Da im Minimalbeispiel der Rang 8 der Koeffizientenmatrix erzwungen wurde, reicht es vorerst den Nullraum zu bestimmen. Die Koeffizientenmatrix wird

aus dem Ausdruck *epipolar-constraint* in Gleichung 4.29 aufgestellt. Der ermittelte Kern und alle seine Vielfache sind mögliche Lösungen für F [4, 14].

$$m'^T \cdot F \cdot m_\sigma = 0 \quad (4.29)$$

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

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

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

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

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

4.6 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.

$$l' = F \cdot m \quad (4.35)$$

$$l = F^T \cdot m' \quad (4.36)$$

l' ist die zu m korrespondierende Epipolargerade. l ist die zu m' korrespondierende Epipolargerade. Zu Berechnung des Epipols e muss der Rechte Kern von F ermitteln werden und für den Epipol e' brauchen wir den linken Kern. Die Abbildung 4.5 zeigt, dass Ergebnis der Epipole im Minimalbeispiel.

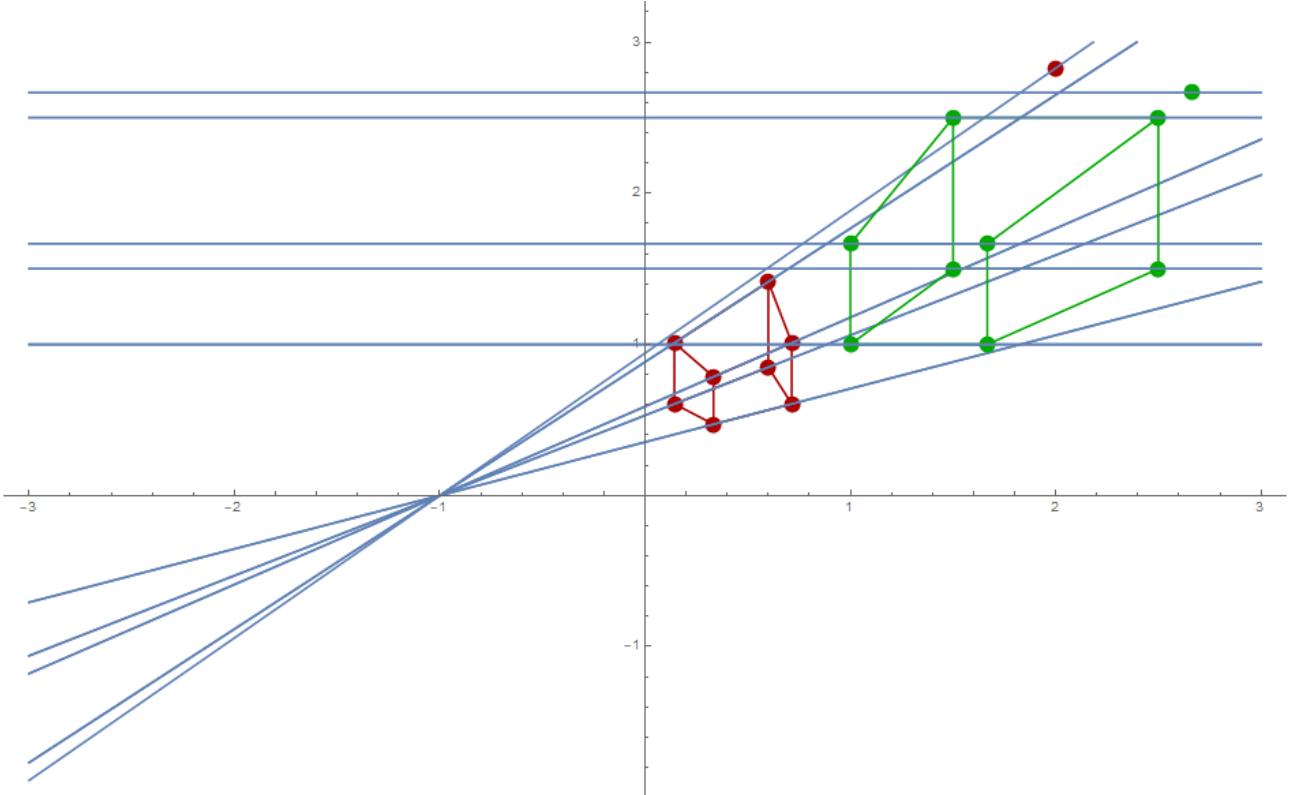


Abbildung 4.5: Die blauen Geraden zeigen die jeweiligen Epipolgeraden. Die vom roten Quader schneiden sich bei -1 im Epipol e' . Die Epipolgeraden vom grünen Quader schneiden sich im Epipol e' im Unendlichen, weshalb die Epipolarlinien Parallel zueinander verlaufen.

4.6.1 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 nochmal eine rein geoemtrische Konstruktion der Epipole und der Epipolgeraden durchgeführt. Im nachfolgenden Beispiel wird der Epipol e' geometrisch konstruiert. Umrechnung von allen Punkten in ein gemeinsames Koordinatensystem, zum Beispiel in (C, β) . Für die Konstruktion der Epipole wird die BasisLinie $B = C'_\beta - C_\beta$, ein Bildebenenpunkt $A_{C'\beta}$ auf I' und die Position von I' selbst benötigt.

$$B = C'_\beta + t \cdot (C'_\beta - C_\beta) \quad (4.37)$$

Um den Epipol zu ermitteln wird der Schnittpunkt der Gerade B mit der Bildebenen I' von C' benötigt. Die Bildebene I' wird in einer Ebenen Normalenform dargestellt. Als Bildebenenpunkt wird $A_{C'\beta}$ von I' von C' gewählt.

$$\vec{n}_0 \cdot [\vec{x} - A_{C'\beta}] \quad (4.38)$$

Um den Schnittpunkt der Gerade B mit der Ebene I zu berechnen, wird die Gerade B in die Ebenengleichung eingesetzt und es wird ein Wert für t ermittelt.

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

$$(4.40)$$

Danach wird die Geradengleichung *BaseLine* in die Ebenengleichung *ImagePlane* eingesetzt und es wird ein Wert für t ermittelt. Der berechnete Wert für t wird dann wiederum in G eingesetzt und das Ergebnis ist der Epipol e'_β . Dieser kann dann wieder über bereits bekannte Transformationen in die Sensorkoordinaten von (S', σ) bezüglich C' überführt werden.

$$e'_\sigma = P' \cdot e'_\beta \quad (4.41)$$

Für die Epipolarlinien $l'_{\sigma'}$ durch $e'_{\sigma'}$, müssen einfach Geradengleichungen eines Sensorpunktes, beispielsweise $A_{\sigma'}$, durch $e'_{\sigma'}$ aufgestellt werden. Die entstandenen Linie ist dann die Epipolarlinie zum Punkt $A_{\sigma'}$ und gleichzeitig die korrespondierende Epipolarlinie zum Sensorpunkt A_σ von I von C .

$$g := A_{\sigma'} + t \cdot (A_{\sigma'} - e'_{\sigma'}) \quad (4.42)$$

4.7 Ermitteln der Essentiellen Matrix über die Fundamentalmatrix

Für die Rekonstruktion der externen Kameraparameter, gibt es verschiedene Ansätze [4]. Es ist zum einen möglich, ohne Vorwissen der Kameramatrizen oder Transformationsmatrizen, aus der Fundamentalmatrix über den sogenannten *Stratified approach* die komplette Projektionsmatrix P und P' zu ermitteln, jedoch ohne weitere Informationen über die 3S-Szene nur bis zu einem Projektiven Vieldeutigkeit[4]. Da in dieser Arbeit davon ausgegangen wird, dass die Kameras zuvor einzeln kalibriert wurden und die intrinsischen Kameraparameter bereits bekannt sind(hier vllt vorherige Arbeit erwähnen), wird für die Rekonstruktion der externen Kameraparameter ein Ansatz verfolgt, in welchen die essentielle Matrix zum Einsatz kommt. Die essentielle Matrix ist eine Spezialform der Fundamentalmatrix und beschreibt den *epipolar constraint*, im kalibrierten Fall, zwischen den normalisierten Bildkoordinaten \hat{m}_σ und $\hat{m}'_{\sigma'}$ [4, 3, 1, 5, 14].

$$\hat{m}'_{\sigma'}^T \cdot E \cdot \hat{m}_\sigma = 0 \quad (4.43)$$

Ist die Fundamentalmatrix F bereits ermittelt worden und die Kameramatrizen K und K' bekannt, kann die essentielle Matrix wie in Kapitel (KAPITELLINK GEOMETRIE F UND E) aus F berechnet werden. Die essentielle Matrix ist eine Fundamentalmatrix, welche zu einem paar normalisierten Projektionsmatrizen P mit $P = [I|0]$ und P' mit $P' = [R'|V']$ korrespondierend ist[4, 5, 1, 14]. Um eine Projektionsmatrix $P' = K'[R'|V']$ auf die normalisierte Form zu bringen, müssen die intrinsischen Parameter für K' bekannt sein.

$$m'_\sigma = P' \cdot M_\delta \quad (4.44)$$

$$m'_\sigma = K'[R'|V'] \cdot M_\delta \mid \cdot K'^{-1} \quad (4.45)$$

$$K'^{-1} \cdot m'_\sigma = K'^{-1} \cdot K'[R'|V'] \cdot M_\delta \quad (4.46)$$

$$\hat{m}'_\sigma = [R'|V']M_\delta \quad (4.47)$$

M_δ ist der ein Objektpunkt im 3D-Raum, welcher mit P' zum Sensorpunkt m'_σ , abgebildet wird. Nach der Normalisierung, wird \hat{m}'_σ als normalisierte Koordinat bezeichnet und die entstandene Projektionsmatrix $P' = [R'|V']$ als normalisierte Projektionsmatrix[4, 14]. Die essentielle Matrix E beschreibt dann die Korrespondenz zwischen den korrespondierenden normalisierten Koordinaten \hat{m}_σ und \hat{m}'_σ , wie in Gleichung 4.43 gezeigt. Um E aus F zu gewinnen, werden die Kameramatrizen K und K' mit F multipliziert. Zu Erinnerung in Kapitel (KAPITELLINK GEOMETRIE E UND F) wurde genau diese Beziehung zwischen E und F in den Gleuchungen 3.98 bis 3.101 aufgeführt.

$$E = K'^T \cdot F \cdot K \quad (4.48)$$

Das Ergebnis dieser Gleichung 4.48 und alle deren Vielfache, sind mögliche Lösungen für die essentielle Matrix. Wie die Fundamentalmatrix muss auch die essentielle Matrix einen Rang von 2 haben. Des Weiteren darf eine 3x3-Matrix nur dann als essentielle Matrix bezeichnet werden, wenn die Singulärwerte, welche durch eine Singulärwertszerlegung ersichtlich werden, bestimmte Merkmale aufweisen. So müssen zwei der drei Singulärwerte gleich und die dritte null sein[4]. Des Weiteren muss für die Determinante gelten, dass $\det(E) = 0$ ist und die Quadratwurzel der Eigenwerte müssen wieder die Singulärwerte ergeben.

Die essentielle Matrix kann, wie die Fundamentalmatrix auch, über den *8-Point-Algorithm* ermittelt werden. Jedoch müssen auch hier die intrinsischen Kameraparameter K und K' bekannt sein. Die korrespondierenden Punkte, werden in normalisierte Bildkoordinaten umgerechnet und dann in eine Koeffizientenmatrix gleich der Matrix in Gleichung 4.34 eingetragen. Danach kann E entweder über die Bestimmung des Kerns oder, im Falle von Realen Bilddaten, dem *least-square-* Verfahren geschätzt werden[4, 14].

4.8 Ermitteln der externen Kameraparameter mit Hilfe der Essentiellen Matrix

Hier kann auch [14] zitiert werden...

Mit der essentiellen Matrix ist es möglich die Transformationsmatrix R' zu ermitteln. Es wird davon ausgegangen, dass für R gilt $R = [I|0]$. Die aus E ermittelte Matrix R' beschreibt dann die Transformation von C' relativ zu C [4, 14].

Um die externen Kameraparameter zu bestimmen, wird zunächst die essentielle Matrix E mit Hilfe der Singulärwertszerlegung in drei Matrizen zerlegt.

$$E = U\Sigma V^T \quad (4.49)$$

Die Singulärwerte befinden sich in der mittleren Matrix $\Sigma = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$ wieder. Damit die Matrix E sich essentielle Matrix nennen darf, muss für zwei der Diagonaleinträge für die Singulärwerte gelten das $\sigma_1 = \sigma_2$ und für σ_3 muss dann dementsprechend gelten, dass $\sigma_3 = 0$. Sollten die Singulärwerte diesen Bedingungen nicht entsprechen, so wird ein so genannten *singularity-constraint*, der Form $\Sigma = \text{diag}(\frac{\sigma_1+\sigma_2}{2}, \frac{\sigma_1+\sigma_2}{2}, 0)$, erzwungen[4]. Im Minimalbeispiel kommt es auf Grund der eigens erstellten Bildpunkte zu keinen Bildfehlern, wie im Realbeispielen. Deswegen ist der *singularity-constraint* nicht notwendig, da die Bedingungen für E erfüllt sind.

Ein wichtiges Detail, was sich später auch auf die vier möglichen Ergebnisse für P auswirkt, ist noch zu beachten. Zum einen sollte einem bewusst sein, dass die Zerlegung der essentiellen Matrix keine eindeutige Lösung hervorbringen muss. Für E muss gelten, dass $\det(E) = 0$ ist. Es wird also vorausgesetzt, dass die Determinante der aus der SVD gewonnenen Matrizen $UV^T = 1$ ist. Angenommen aus der SVD von E ergibt sich für die Determinanten von $UV^T = -1$, so können die Determinanten der Matrizen U und V getrennt voneinander bestimmt werden. Sollte $\det(U) = -1$ oder $\det(V) = -1$ oder beides zusammen der Fall sein, so kann die jeweilige Matrix einfach mit -1 multipliziert werden. E setzt sich zusammen aus einer Rotationsmatrix R und einer schiefsymmetrischen Matrix S .

$$E = [v]_x R \quad (4.50)$$

$$S = [v]_x \quad (4.51)$$

$$E = SR \quad (4.52)$$

Zur Schätzung von S und R werden des Weiteren die schiefsymmetrische Matrix W und die Blockdiagonale Matrix Z eingeführt[4]. Mit diesen Matrizen lassen sich die gesuchten R und S für C' rekonstruieren, jedoch nur bis zu einer gewissen Skaleninvarianz[4, 14].

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (4.53)$$

Mit dem, je nach dem leicht modifizierten, Ergebnis der SVD von E lassen sich nun jeweils zwei mögliche Lösungen für S und R aufstellen.

$$S_1 = -UZU^T \quad R_1 = UW^TV^T \quad (4.54)$$

$$S_2 = UZU^T \quad R_2 = UWV^T \quad (4.55)$$

Um sicher zu gehen, dass es sich bei R_1 und R_2 auch um gültige Rotationsmatrizen handelt, kann eine Probe durchgeführt werden. Zum einen muss $R \cdot R^T = I_{3x3}$ sein. I_{3x3} bezeichnet in diesem Fall die 3x3-Einheitsmatrix. Des Weiteren kann man überprüfen, ob die Determinante $\det(R_1) = 1$ ist. Ist die Determinante = -1, so ist die Rotation für ein Rechtsdrehendes System, ist die Determinante = 1, so handelt es sich um ein linksdrehendes System[4]. S_1 und S_2 sind jeweils schiefsymmetrische Matrizen aus welchen, jetzt noch zwei Lösungen für v finden lassen müssen[4].

$$St = [v]_\times \cdot v = v \times v \quad (4.56)$$

Um v zu ermitteln muss also lediglich der Kern von S_1 und S_2 gefunden werden. Die jeweiligen Ergebnisse für v_1 und v_2 sind bis auf ihre Vorzeichen die selben.

Die externen Kameraparameter lassen sich wie gesagt nur bis zu einem Skalierungsfaktor genau bestimmen. Die Rotationen R ist von dieser Skaleninvarianz nicht betroffen, es betrifft nur den Translationsvektor \vec{v} . Wie mit der Skaleninvarianz weiter Verfahren wird, wird im nachfolgenden Abschnitt der Szenenrekonstruktion durch Triangulierung noch aufgeführt. Letztendlich können, für die Rekonstruktion der externen Kameraparameter vier Lösungen für P gefunden werden.[4, 14]. λv heißt dabei, dass sowohl v also auch alle Vielfache von v , Lösungen sein können, was durch die Skaleninvarianz der Lösung bedingt ist[4, 14].

$$P = [UWV^T] + \lambda v \quad \text{or} \quad [UW^TV^T] + \lambda v \quad (4.57)$$

$$\text{or} \quad [UWV^T] - \lambda v \quad \text{or} \quad [UW^TV^T] - \lambda v \quad (4.58)$$

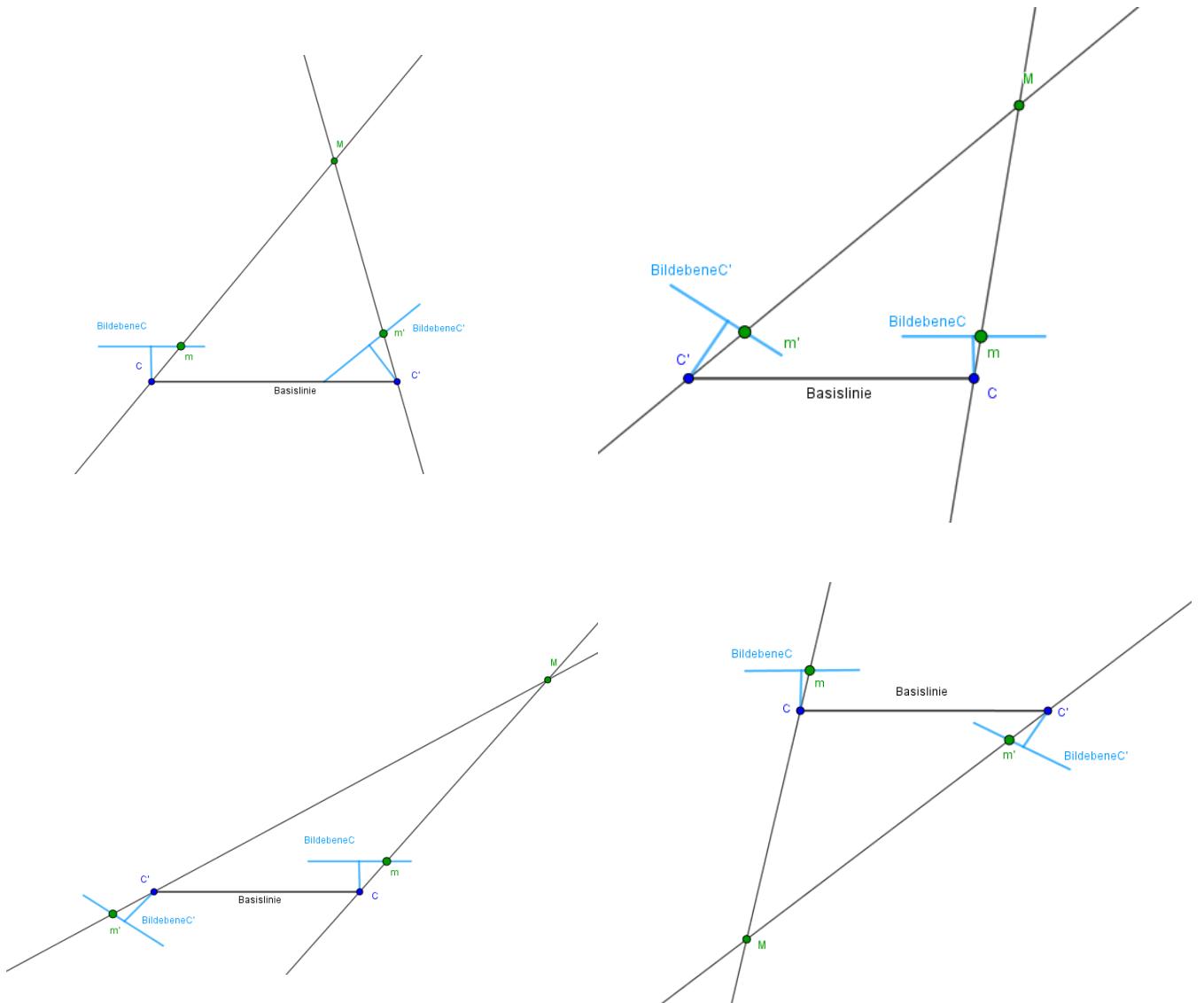


Abbildung 4.6: Anordnung der Kameras bei den vier verschiedenen Lösungen für P

4.9 Szenenrekonstruktion durch Triangulation

Unter Triangulierung versteht man das Rekonstruieren der 3D-Szenenpunkte durch Schnittpunktberechnung derjenigen Geraden, welche durch die jeweiligen Projektionszentren der Kameras und deren korrespondierenden Bildpunkte auf deren Bildebenen gehen, so dass sich ein Dreick bildet.

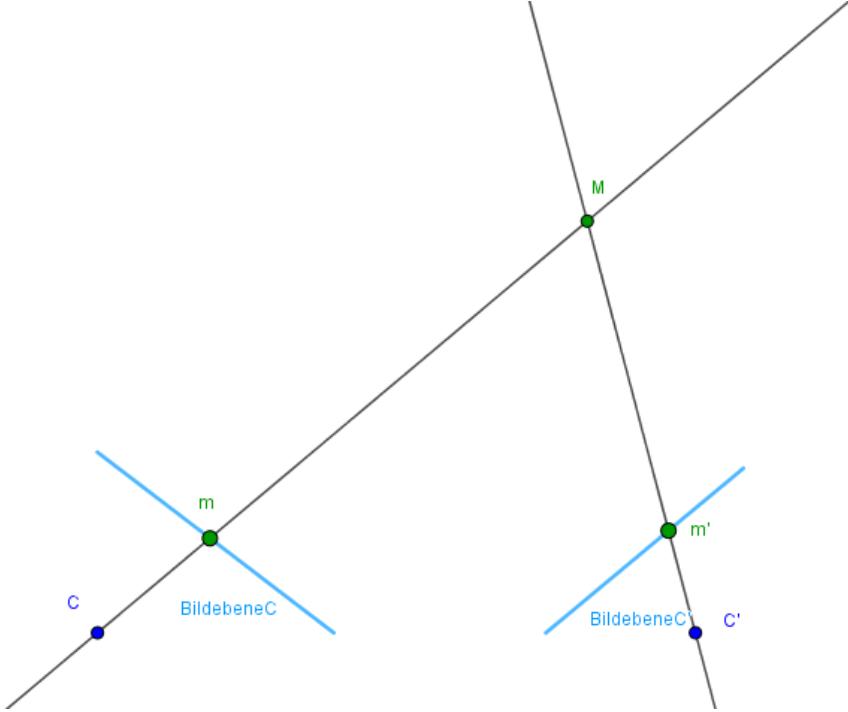


Abbildung 4.7: Optimale Triangulierung: Beide Geraden Treffen sich in einem Punkt im 3D-Raum

Da das Minimalbeispiel mit reinen Daten arbeitet, ist damit garantiert, dass die Linien sich in einem Schnittpunkt treffen. Anders hingegen wäre es in einem realen Beispiel mit korrespondierenden Punkten, welche Beispielsweise über einen *SURF*-Algorithmus detektiert wurden[15]. In Realbildern, können Bildfehler wie beispielsweise Rauschen nicht vermieden werden, des Weiteren können korrespondierende Punkte nicht immer exakt auf den Pixel genau bestimmt werden. Diese Fehler führen dazu, dass wenn ein Schnittpunkt der Geraden durch die vermeintlichen korrespondierenden Punkte nicht gefunden werden kann, da die Geraden sich sehr wahrscheinlich nicht in einem Punkt treffen werden[15, 4]. Für diese Fälle gibt es mehrere Näherungsverfahren, wovon eines, im Kapitel (KAPITELLINK ZU SAMPSON-APPROXIMATION), im Realbeispiel eingeführt wird[4]. In diesem Minimalbeispiel tritt der optimale Fall ein. Das bedeutet, dass ein zu Bildpunkt m korrespondierender Bildpunkt m' auf der zu m korrespondierenden Epipolarlinie l' liegt und somit garantiert ist, dass sich die Geraden \overline{mM} und $\overline{m'M}$ auf jedenfall in einem gemeinsamen Punkt schneiden. Um den Schnittpunkt beider Geraden zu berechnen, werden zum einen die Bildpunkte $A_\tau, B_\tau, C_\tau, D_\tau, A2_\tau, B2_\tau, C2_\tau, D2_\tau$ und $A_{\tau'}, B_{\tau'}, C_{\tau'}, D_{\tau'}, A2_{\tau'}, B2_{\tau'}, C2_{\tau'}, D2_{\tau'}$, sowie die korrekt ermittelte Projektionsmatrix P' von zuvor benötigt. Bei der Berechnung der externen Kameraparameter wurde festgesetzt, dass die Projektionsmatrix von Kamera eins $P = [I|0]$ und dementsprechend $P' = [R^T | -R^T V]$ die Transformation für Kamera zwei, ausgehend von Kamera eins ist. Also ist die Position von Kamera eins in Weltkoordinaten $C_\delta = [0 \ 0 \ 0 \ 1]^T$. Was jetzt noch fehlt ist die Position von Kamera zwei in Weltkoordinaten. Da die Szene dieses Minimalbeispiels durchkonstruiert wurde, ist C' eigentlich bekannt. Es soll nun aber angenommen werden, dass die Position von Kamera zwei nicht bekannt ist und diese wie im realen Fall erst einmal aus P berechnet werden muss. Um C' zu ermitteln, wird der Translationsvektor V aus $P' = [R^T | -R^T V]$ benötigt.

$$C'_\delta = -R * (-R^T V) \quad (4.59)$$

Im nächsten Schritt, müssen die Bildebenenpunkte von C' noch in das selbe Koordinatensystem wie die Bildebenepunkte von C transformiert werden. Da Kamera eins Deckungsgleich mit dem Weltkoordinatensystem ist, sind die Bildpunkt der Bildebene I von C bereits in Weltkoordinaten gegeben, diese müssen nur noch mit, ζ als dritte Tiefenkomponenten z , erweitert werden. Die Bildpunkte von C' , werden ebenfalls mit ihrem ζ erweitert und danach mit der Projektionsmatrix P , welche als V die Koordinaten von C' besitzt, in das Weltkoordinatensystem überführt. Sind die Basen der

Bildpunkte von C_δ und C'_δ , mit $\delta = \beta$ im selben Koordinatensystem, so können nun die Geraden-Gleichungen durch die Projektionszentren C und C' und den entsprechenden Bildebenenkoordinaten $A_\delta, B_\delta, C_\delta, D_\delta, A2_\delta, B2_\delta, C2_\delta, D2_\delta$ und den neu umgerechneten Punkten $A'_\delta, B'_\delta, C'_\delta, D'_\delta, A2'_\delta, B2'_\delta, C2'_\delta, D2'_\delta$ gezogen. Beispielhaft wird dies Anhand der korrespondierenden Punkte A_τ und dem umgerechneten $A'\tau$ aufgezeigt.

$$A_\delta + t * (A_\delta - C_\delta) = 0 \quad (4.60)$$

$$A'_\delta + t' * (A'_\delta - C'_\delta) = 0 \quad (4.61)$$

Beide Geraden gleichsetzen

$$A_\delta_x - t_x - C_\delta_x = A'_{\delta x'} - t'_x - C'_{\delta' x} \quad (4.62)$$

$$A_\delta_y - t_y - C_\delta_y = A'_{\delta y'} - t'_y - C'_{\delta' y} \quad (4.63)$$

Nun muss für jedes Linienpaar eine Lösung für t und t' gefunden werden und die Lösungen in die Gleichungen 4.65 und 4.66 eingesetzt werden. Es sollte für beide Gleichungen die selbe Lösung für den rekonstruierten Punkt A im Raum ergeben. Die Lösung entspricht meist noch nicht exakt dem eigentlichen Ergebnis, das liegt an der zuvor erwähnten Skaleninvariants der Rekonstruktion der externen Kameraparameter. Bei den zuvor ermittelten externen Kameraparametern, ist der Translationsvektor Skaleninvariant, was dazu führt, dass die rekonstruierten Objekte nach der Szenenrekonstruktion noch nicht ihrer Originalgröße entsprechen. Es wird noch ein Skalierungsfaktor benötigt, welcher die Szene auf Originalgröße skaliert. Hierfür ist es in einer Realzene ratsam, wenn man zuvor von zwei Punkten in Szene den Abstand zueinander abmisst, um anhand dessen einen Skalierungsfaktor zu berechnen. Im hier beschriebenen Minimalbeispiel sind die Originalkoordinaten der Objektpunkte im Raum bekannt, weshalb hier nach dem passenden Vielfachen der Rekonstruierten Punkte gesucht werden kann. Da die Verhältnisse der Abstände der Punkte zueinander bei der Skalierung beibehalten wird, kommt zu keinen Verfälschungen des Objektes, da die Rotationen der beiden Kameras unangetastet bleibt. Abbildung 4.6 zeigt, dass sich durch verändern des Translationsvektors nur die Größe des Objektes ändert nicht aber seine Orientierung im Raum.

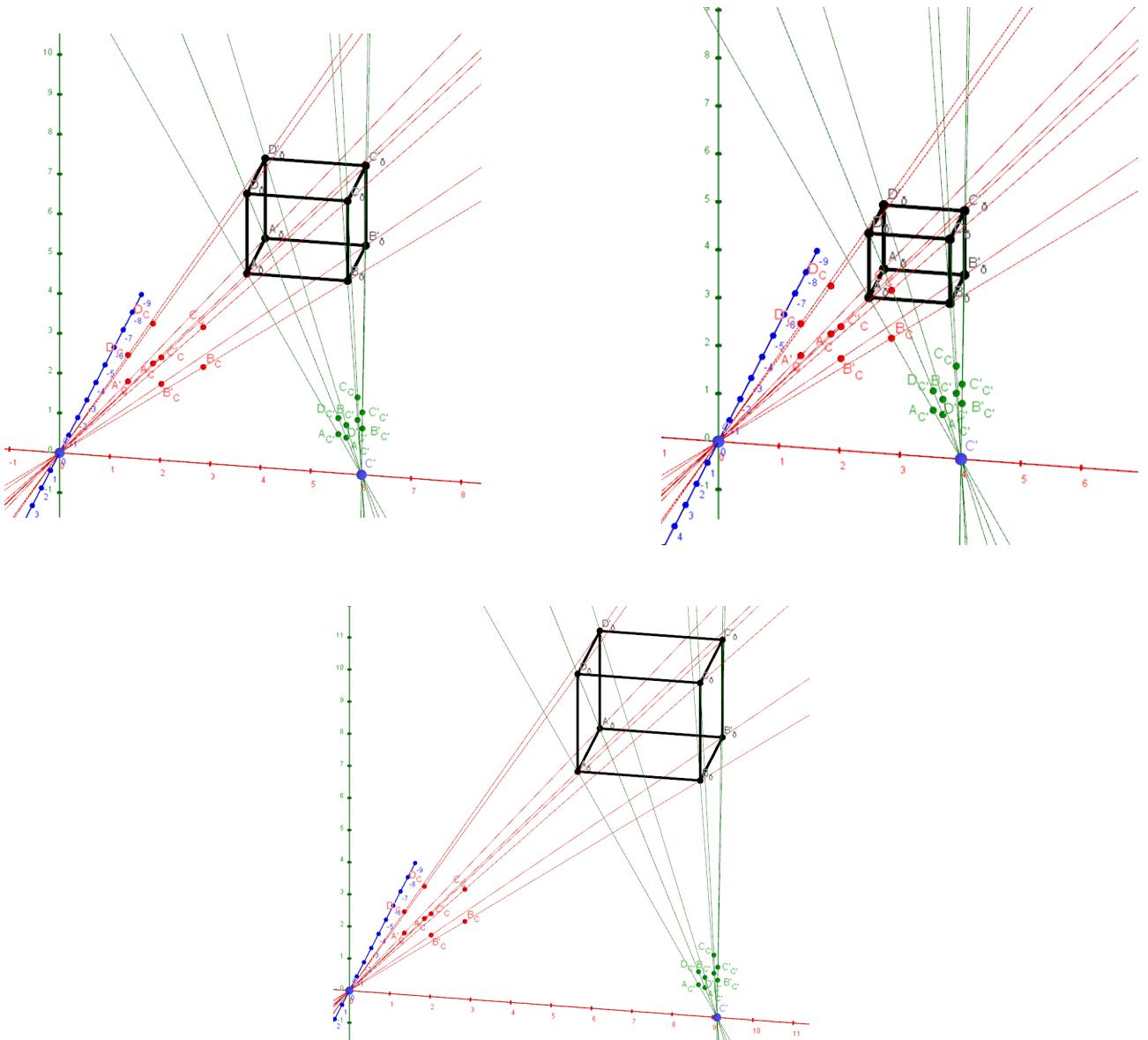


Abbildung 4.8: Veranschaulichung der Skaleninvarianz und dessen Auswirkung auf die geometrische Form der Objekte

Abbildung 4.7 zeigt die Komplett rekonstruierte Szene des Minimalbeispiels, welche beweist, dass die beschriebenen Methoden für das Minimalbeispiel mit reinen Daten und auch bei Kameras mit unterschiedlichen Auflösungen, funktioniert hat.

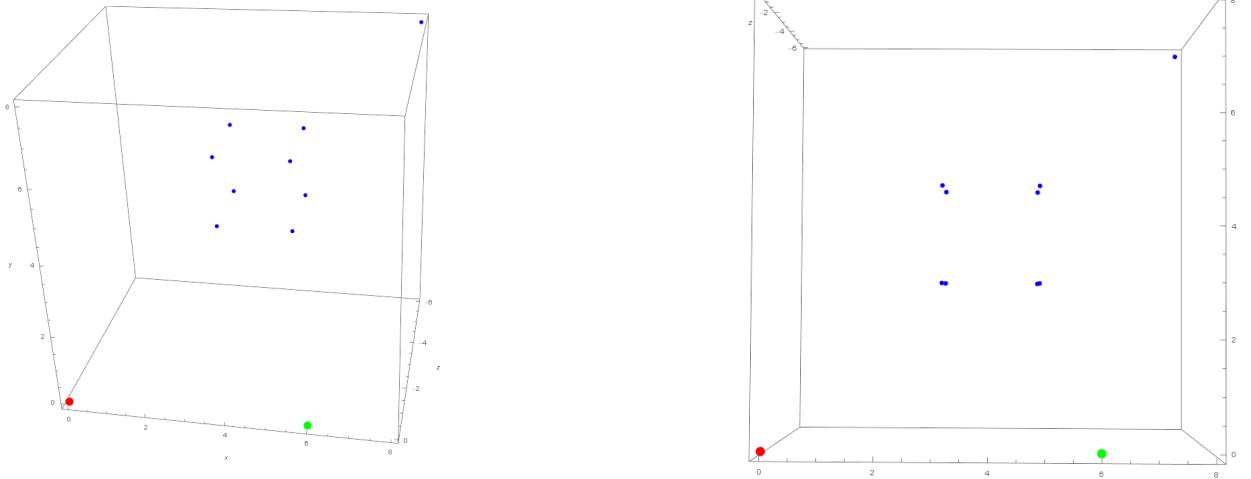


Abbildung 4.9: auf Originalgröße skalierte rekonstruierte Szene

4.10 Rektifizierung

Im Minimalbeispiel ist die Triangulierung und Rekonstruktion der 3D-Weltpunkte ohne Kommunikationen durchführbar. Das liegt daran, dass mit reinen Werten gerechnet wird und Fehler wie Beispielsweise Bildrauschen nicht vorkommen. Im Minimalbeispiel kann davon ausgegangen werden, dass die Linien zweier korrespondierender Punkte sich, welche durch die jeweiligen Kamerazentren und Bildpunkte gehen, sich ziemlich sicher in einem Punkt treffen werden. In einem Beispiel mit realen Daten ist es nicht unwahrscheinlich, dass die herausgefilterten korrespondierenden Punkte, nicht zu hundert Prozent stimmen. Es kommt immer zu kleineren Abweichungen, was dazu führen kann, dass wenn die Linien der korrespondierenden Punkte im Realbild sich nicht treffen. Ein Grund dafür ist, dass sie nicht hundertprozentig auf der selben Höhe im Bild liegen und die Linien sich somit verfehlten

(BILD EINFÜGEN. weiß noch nicht genau wie das aussehen soll....).

Im Realbeispiel dieser Arbeit wird das Auftreten solcher Fehler durch das sogenannte *Sampson-Approximation* - Verfahren behoben, welches bei kalibrierten Fällen zum Einsatz kommt[4]. Mehr zu diesem Verfahren wird in Kapitel (HIER KAPITEL LINK) aufgeführt. Das ist eine Möglichkeit um eine Szenerekonstruktion trotz Fehlerhafter korrespondierender Punkte zu ermöglichen. Ein weiteres weit verbreitetes Verfahren, ist cor der Szenenrekonstruktion durch Triangulierung eine Rektifizierung beider Bilder vorzunehmen[16, 17, 18, 19]. Da bestimmte Formen der Rektifizierung keine vorherige Kalibrierung der Kameras benötigen, wird diese Methode in den meisten gängigen Echtzeit-Szenenrekonstruktionen eingesetzt. [19, 18, 20]. Rektifizierte Bilder müssen zwei Eigenschaften erfüllen. Zum einen müssen alle Epipolargeraden parallel zur x-Koordinatenachse verlaufen und zweitens müssen alle korrespondierenden Punkte die selben y-Koordinaten besitzen[17]. Mit Hilfe dieser Eigenschaften ist es somit möglich die entstandenen korrespondierenden Epipolarlinien als horizontale Scanlinien zu benutzen[18, 17]. Mit Hilfe dieser Scanlinien und den darauf sich befindenden korrespondierenden Punkten ist es zum Beispiel möglich eine Tiefenkarte des Bildes zu berechnen allein durch die Differenz der horizontalen Lage der korrespondierenden Punkte[18, 17].

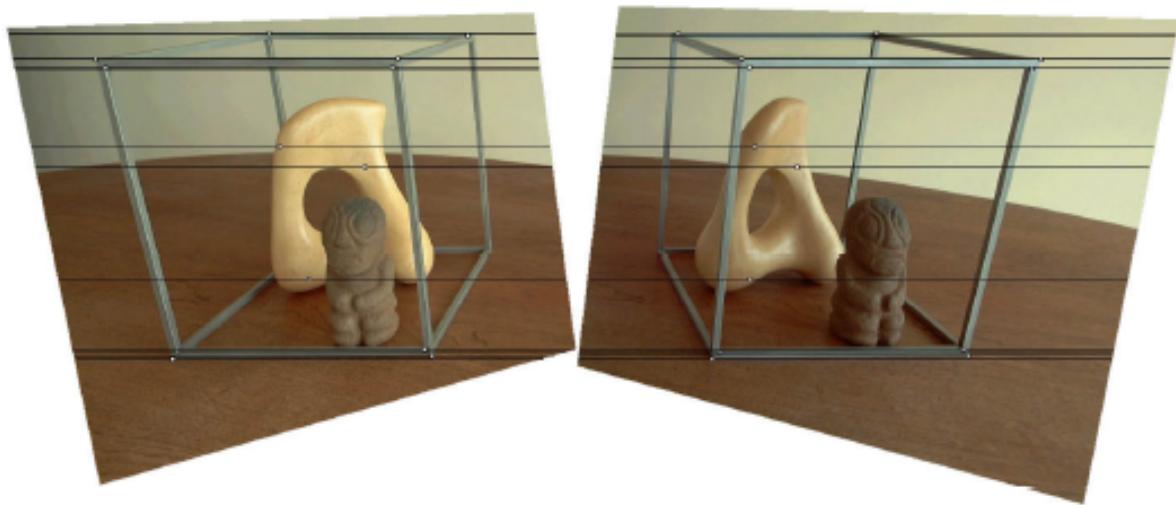


Abbildung 4.10: Beispiel eines rektifizierten Bildes. Quelle: [17]



Abbildung 4.11: Beispiel einer einfachen Tiefenkarte eines Stereobildpaars nach der Rektifizierung.
Quelle: [18]

Die Rektifizierung, allem voraus vor allem die Optimierung des Rektifizierungsvorgangs, von Stereo-

oder auch multplen- Kamerasystemen, wird heutzutage von vielen Entwicklergruppen der Computer Vision untersucht(Der satz ist mist!). Es gibt mittlerweile viele Ansätze, jedoch funktionieren nicht alle bei den selben Fällen. So setzten zum Beispiel manche Rektifizierungsalgorithmen voraus, dass die Bilder von Kameras mit selber Auflösung aufgenommen wurden. Ein Beispiel ist die Rektifizierung welche in *Matlab* verwendet wird [16]. Die Rektifizierung wurde anhand einer Methode implementiert, welcher sich ähnlich verhält wie in [21] beschrieben. Die Grundidee hier hinter ist, dass die Kameramatrizen von zwei Kameras so aufgebaut sind dass die intrinsischen Parameter die selben sind, sie sich aber in ihren Rotationen und Translationen voneinander unterscheiden. Die extrinsischen Kameraparameter werden dann dementsprechend so manipuliert, dass die Bildebenen Achsenparallel zueinander stehen[21, 19]. Um horizontale Epipolarlinien zu erhalten muss gleichzeitig die Basislinie zwischen den zwei Kamerazentren parallel zur neuen x-Achse beider Kameras sein. Zudem soll, um eine angemessene Rektifizierung zu gewährleisten, müssen konjugierende Punkte die selbe vertikale Koordinate haben. Dies wird hier durch die Bedingung gewährleistet, dass beide Kameras die selben intrinsischen Parameter haben[21]. Eine Frage welche mit unter in dieser Arbeit beantwortet werden sollte, war, ob es möglich ist, ohne deutlich größeren Aufwand eine Kamerakalibrierung und Szenerekonstruktion mit Kameras unterschiedlicher Auflösung zu gewährleisten. Im Kapitel (KAPITELLINK EINFÜGEN), in welchem ausführlich die Epipolargeometrie vorgestellt wurde, wurde bereits bezug auf die unterschiedlichen Auflösungen genommen. Prinzipiell spielen unterschiedliche intrinsische Kameraparameter keine Rolle, wenn es um die Rekonstruktion der Kameraposen geht, da die Fundamental Matrix und die essentielle Matrix die Information über die intrinsischen und extrinsischen Kameraparameter besitzen und es klar gestellt wurde, dass die Bildkoordinatensysteme der Kameras nicht identisch sein müssen. [3]. In dieser Arbeit wurde ein Rektifizierungsalgorithmus nach *Zhang*[17] implementiert, welcher sich die Fundamentalmatrix zu nutzen macht. *Loop* und *Zhang* zerlegen jede Kollinearität in eine Ähnlichkeitstransformation, eine Schertransformation und eine projektive Transformation. Die projektive Komponenten wird dabei in einem nichtlinearen Optimierungsprozess so affin wie möglich gemacht.[19, 17]. Im folgenden wird zunächst der genaue Vorgang des implementierten Algorithmus genauer erklärt und **des Weiteren werden zwei Beispiele vorgestellt, welche die Bilder des Minimalbeispiels einmal mit gleichen intrinsischen Parametern und einmal mit unterschiedlichen intrinsischen Parametern der Kamera aufzeigt. Es wird sich Herausstellen, dass beide Beispiele eine gelungene Rektifizierung der Bilder aufweisen.(Nochmal genau nachprüfen ob das geht!!!)**. Während sich einige Rektifizierungsverfahren im 3D-Raum abspielen, wird beim Verfahren nach *Zhang*, hauptsächlich im 2D-Raum gearbeitet. Des Weiteren wird vorausgesetzt, dass die Fundamental Matrix F und somit auch korrespondierende Punkte bereits bekannt sind. Sind die intrinsischen Kameraparameter bekannt, so wird aus der Fundamentalmatrix die Essentielle Matrix. Das Verfahren kann sowohl in einem kalibrierten als auch in einen unkalibrierten Fall angewendet werden[17]. Im Algorithmus wurde der unkalibrierte Fall implementiert und somit wird in der Erläuterung und in den danach folgenden Beispielen die Fundamentalmatrix F verwendet. Die korrespondierenden Punkte werden mit x für das erste beziehungsweise x' für das zweite Bild definiert, die Kamerazentren dementsprechend mit C und C' . Bildebene der ersten Kamera wird mit I definiert und die Bildebene von Kamera zwei mit I' , die entsprechenden Epipole mit e und e' . Der Prozess der im Algorithmus erfolgt kann quasi als eine Transformation der Epipolar Geometrie eines Bildpaars in eine kanonische Form angesehen werden. Diese Transformation wird durch eine Homographiematrix durchgeführt, welche sich aus den bereits erwähnten drei Komponenten zusammenstellt. Zu Beginn sei noch erwähnt dass wir pro Bild zwei unterschiedliche Homographien H und H' brauchen. Die Fundamentalmatrix liefert, die Epipolarbedingung, dass $x'^T F x = 0$ ergibt, wenn x' auf der zu x korrespondierenden Epipolarlinie liegt. Die korrespondierenden Punkte x und x' werden, für die Rektifizierung, jeweils mit den Homographien H und H' verrechnet.

$$\bar{x} = Hx \quad (4.64)$$

$$\bar{x}' = Hx' \quad (4.65)$$

Die Fundamentalmatrix, welche sich aus durch die Rektifizierten korrespondierenden Punkte resultiert, wird mit \bar{F} bezeichnet. Daraus folgt für die Fundamentalmatrix folgendes:

$$\bar{x}'^T \bar{F} \bar{x} = 0 \quad (4.66)$$

$$\rightsquigarrow x'^T H' T \bar{F} H x = 0 \quad (4.67)$$

$$\rightsquigarrow F = H'^T [i]_x H \quad (4.68)$$

Das Ziel ist es diese zwei Homographien in deren bereits erwähnten projektiven und affinen Komponenten zu zersetzten, wobei diese die jeweils entstehenden Bildverzerrungen minimieren sollen. Die Homographiematrizen bestehen aus drei Linien, welche jeweils durch den Epipol verlaufen. Des Weiteren werden noch ein paar weitere Bedingungen für die jeweils drei Linien festgelegt. So müssen die Linien v und v' sowie w und w' korrespondierende Epipolarlinien sein. Diese Bedingung schafft eine geometrische Verbindung beider Bilder zueinander und ist gerade bei der Minimierung der durch die Rektifizierung entstehenden Bildverzerrung von Bedeutung.

$$H = \begin{bmatrix} u^T \\ v^T \\ w^T \end{bmatrix} = \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & w_c \end{bmatrix} \quad (4.69)$$

$$H' = \begin{bmatrix} u'^T \\ v'^T \\ w'^T \end{bmatrix} = \begin{bmatrix} u'_a & u'_b & u'_c \\ v'_a & v'_b & v'_c \\ w'_a & w'_b & w'_c \end{bmatrix} \quad (4.70)$$

Für die Bestimmung der einzelnen Komponenten von H und H' werden diese in ihre projektiven und affinen Teilstücke zerlegt. Davor wird noch die letzte Komponente w_c raus dividiert, um somit skaleninvariante Matrizen H und H' zu bekommen.

$$H = \begin{bmatrix} u^T \\ v^T \\ w^T \end{bmatrix} = \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & 1 \end{bmatrix} \quad (4.71)$$

$$H' = \begin{bmatrix} u'^T \\ v'^T \\ w'^T \end{bmatrix} = \begin{bmatrix} u'_a & u'_b & u'_c \\ v'_a & v'_b & v'_c \\ w'_a & w'_b & 1 \end{bmatrix} \quad (4.72)$$

Beide Matrizen werden nun auf die selbe Weise in ihre projektiven und affinen Bestandteile zerlegt.

$$H = H_p \cdot H_a \quad (4.73)$$

$$H' = H'_p \cdot H'_a \quad (4.74)$$

H_p ist die projektive Komponente, sie bezieht sich nur auf die letzte Zeile der Matrix H und wirkt sich somit auch nur auf die homogenen Komponenten der mit ihr verrechneten Punkte aus.

$$H_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ w_a & w_b & 1 \end{bmatrix} \quad (4.75)$$

Die affine Komponenten H_a lässt sich aus H und H_p konstruieren. Es gilt:

$$H_a = H \cdot H_p^{-1} = \begin{bmatrix} u_a - v_c w_b & v_c w_a - v_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad (4.76)$$

Für die Matrizen H'_p und H'_a gilt das selbe nur mit den Epipolarlinien u' , v' und w' . Die projektive Matrix sogt dafür, dass die Epipole beider Bilder ins unendliche gesetzt werden und die Epipolarlinien der Bilder jeweils parallel zueinander verlaufen. Zu Beginn wurde erwähnt dass es eine Zerlegung in eine projektive, eine Ähnlichkeits- und eine Scherungstransformation gibt. Die projektive Komponente ist mit H_p und H'_p bereits vollständig definiert. Was nun noch fehlt ist die Zerlegung der affinen Matrizen H_a und H'_a in ihre jeweiligen Ähnlichkeits- und Scherungstransformationen.

$$H_a = H_s \cdot H_r \quad (4.77)$$

$$H_r = \begin{bmatrix} v_b - v_c w_b & v_a - v_c w_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad (4.78)$$

$$H_s = \begin{bmatrix} u_a & u_b & u_c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.79)$$

H_r und auch H'_r definieren eine Rotation und auch eine Verschiebung, welche die bereits parallelen Epipolarlinien beider Bilder zueinander parallel und horizontal ausrichtet. Durch die Verschiebung werden die korrespondierenden Epipolarlinien noch auf die selbe Höhe verschoben. Somit entstehen die gewünschten Scanlinien in den Bildern. Die Matrix H_s und H'_s wirken sich nur auf die u -Elemente der Matrix H und H' aus und definieren eine Scherung. Sie haben keine Auswirkung auf die Rektifizierung an sich aber sorgen dafür, dass die horizontale Verzerrung der beiden Bilder zueinander reduziert wird.

4.10.1 Projektive Transformation

Die projektiven Matrizen H_p und H'_p werden von den Linien w und w' bestimmt. w und w' sind dabei jedoch nicht unabhängig. Definiert werden sie durch einen Punkt $z = [\lambda \ \mu \ 0]^T$, welche die, durch die Rektifizierung entstehende, Bildverzerrung minimieren soll. Für beide Bilder werden w und w' folgendermaßen gewählt

$$w = [e]_x \cdot z \quad (4.80)$$

$$w' = F \cdot z \quad (4.81)$$

Jedes beliebige z würde zwei korrespondierende Epipolarlinien definieren, um ein z zu finden, welches die Verzerrung der Bilder minimiert, wird ein Kriterium aufgestellt, welches ein z finden soll, dass die Verzerrung minimal halten wird. Minimierung bedeutet in diesem Falle, dass versucht wird die Matrizen H_p und H'_p so affin wie möglich zu machen. So affin wie möglich bedeutet, dass die Werte von w_a und w_b so nah wie möglich an den Wert 0 gebracht werden sollen.

$$H_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ w_a & w_b & 1 \end{bmatrix} \quad (4.82)$$

Jedoch sollen sie nicht ganz null werden, da die projektive Matrix dann keine projektive mehr wäre, sondern eine affine. Deswegen heißt es auch sie soll so affin wie möglich gemacht werden. Das selbe gilt natürlich auch für w'_a und w'_b aus H'_p . Wäre das der Fall, so wären die beiden Epipole e und e' bereits im unendlichen und die Matrizen H_p und H'_p hätten keine Auswirkungen auf die Punkte. Für die Minimierung wird die Methode des *least-square-fitting*, also die Anpassung des kleinsten Quadrats, genutzt[22]. Es werden also die Gewichtungen der Punkte in beiden Bildern in der Methode der Anpassung der kleinsten Quadrate verbaut, welche versucht eine Funktion zu finden, die einen Wert für z berechnen soll welcher die Bildverzerrung minimal hält. **Anders ausgedrückt man sucht einen Wert für z , welcher am nächsten an den gegebenen Punktesammlungen der jeweiligen Bildern dran liegt, wobei für z bereits gilt, dass es sich um einen Punkt im Unendlichen handeln soll**[17, 22]. Angenommen, dass die Annäherungsfunktion $g(x)$ eine Funktion $f(x)$, mit $x \in [a, b]$, annähern soll, dann versucht die Methode, die Summe der Quadrate der ordinatischen Differenzen, welche zwischen den von der Funktion generierten Punkten und den Punkten aus den Daten gewonnen wird, zu minimieren[22, 23]. Zum Beispiel werden n Datenpunkte angenommen, dann gilt:

$$e = \sum_{i=1}^n [f(x_i) - g(x_i)]^2 \quad (4.83)$$

Für die Minimierung der Bildverzerrung werden die Gewichtungen der Punkte beider Bilder benötigt. p_i beinhaltet alle Punkte von Bild eins und p_j beinhaltet alle Punkte von Bild zwei. Angenommen wir nehmen einen Punkt aus Bild eins $p_{i1} = [p_{i1,u} \ p_{i1,v} \ 1]^T$, so soll dieser Punkt mit der Matrix H_p zu einem Punkt der Form $p_{i1} = [\frac{p_{i1,u}}{w_i} \ \frac{p_{i1,v}}{w_i} \ 1]^T$ transformiert werden. w_i ist die Gewichtung welche durch die Verrechnung von w mit p_i zustande kommt.

$$w_i = w^T p_i \quad (4.84)$$

Ist die Gewichtung der Punkte identisch gibt es keine projektive Verzerrung und die Homographie ist eine affine Transformation. Jedoch wenn die Epipole der Bilder ins Unendliche transformiert werden sollen, so können H_p und H'_p keine affine Homographien sein. Sonst könnte man die Epipole nur innerhalb der affinen Ebenen, sprich den Bildebenen, verschieben. Also bildet der Versuch H_p und H'_p so affin wie möglich zu machen die Basis für die Minimierung. Im Realbeispiel werden alle Pixel des Bildes verwendet. Die Rektifizierung wurde im aufgeführten Beispiel anhand des erstellten Minimalbeispiels durchgeführt, somit wurden die Eckpunkte des Quaders des jeweiligen Bildes für die das Minimierungskriterium verwendet. Es wird eine Funktion nach dem Prinzip der Anpassung der kleinsten Quadrate aufgestellt, welche die Abweichung der Gewichtung der Punkte in Bezug auf die Gewichtung des Bildzentrums p_c berechnet. p_c ergibt sich aus der Mittelung aller verwendeten Punkte eines Bildes $p_c = \frac{1}{n} \sum_{i=1}^n p_i$, dessen Gewichtung ergibt sich aus $w_c = w^T p_c$. Die gesuchte Abweichung ausgedrückt in der Anpassung der kleinsten Quadrate ergibt dann die folgende Formel.

$$\sum_{i=1}^n \left[\frac{w_i - w_c}{w_c} \right]^2 \quad (4.85)$$

$$\rightsquigarrow \sum_{i=1}^n \left[\frac{w^T(p_i - p_c)}{w^T p_c} \right]^2 \rightsquigarrow \sum_{i=1}^n \left[\frac{w^T(p_i - p_c)(p_i - p_c)^T w}{w^T p_c p_c^T w} \right] \quad (4.86)$$

Vereinfacht lässt sich das auch in einer Matrixgleichung angeben

$$\frac{w^T P P^T w}{w^T p_c p_c^T w} \quad (4.87)$$

in welcher für P gilt:

$$P = \begin{bmatrix} p_{1,u} - p_{c,u} & p_{2,u} - p_{c,u} & \dots & p_{i,u} - p_{c,u} \\ p_{1,v} - p_{c,v} & p_{2,v} - p_{c,v} & \dots & p_{i,v} - p_{c,v} \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad (4.88)$$

die Gleichungen 4.79 bis 4.83 werden ebenfalls für die Punkte p_j in Bild zwei aufgestellt. So ergibt sich für das zweite Bild die Matrixgleichung:

$$\frac{w'^T P' P'^T w'}{w'^T p'_c p_c'^T w'} \quad (4.89)$$

Das Ziel ist es einen Wert für z zu finden, welches bis jetzt noch nicht ersichtlich in den Gleichungen vorkommt. Also werden w und w^T noch mit ihren Definitionen aus den Gleichungen 4.75 und 4.76 ersetzt. Gleichzeitig werden die Gleichungen 4.82 und 4.84 summiert um die Gleichung zu erhalten, welche sich auf beide Bilder gleichzeitig bezieht und somit eine Lösung für z , das für beide Bilder gilt, gesucht werden kann.

$$\frac{z^T [e]_x^T P P^T [e]_x z}{z^T [e]_x^T p_c p_c^T [e]_x z} + \frac{z^T F^T P' P'^T F z}{z^T F^T p'_c p_c'^T F z} \quad (4.90)$$

Für den weiteren Verlauf werden die Ausdrücke noch durch die Variablen A, B, A' und B' vereinfacht.

$$A = [e]_x^T P P^T [e]_x \quad (4.91)$$

$$B = [e]_x^T p_c p_c^T [e]_x \quad (4.92)$$

$$A' = F^T P' P'^T F \quad (4.93)$$

$$B' = F^T p'_c p_c'^T F \quad (4.94)$$

$$\rightsquigarrow \frac{z^T A z}{z^T B z} + \frac{z^T A' z}{z^T B' F z} \quad (4.95)$$

Da die dritte Komponente von z laut definition null sein soll, wird zu $z = \begin{pmatrix} \lambda \\ \mu \end{pmatrix}$ umgeschrieben. A, B, A' und B' sind 3x3-Matrizen, von welchen uns dann nur noch der erste 2x2-Block interessiert. Bei dem somit aufgestellten Minimalisierungs Kriterium, handelt es sich um ein nicht lineares optimierungs Problem. Die Gleichung 4.90 ist dann minimiert, wenn die erste Ableitung dieser Funktion nach $\lambda =$ gleich null ist. Es entsteht also ein Polynom mit dem Grad sieben, da die 4.90 die Summe zweier rationaler Funktionen ist, welche jeweil das Verhältnis von quadratischen Polynomen darstellt.

Hier soll das Polynom aufgestellt werden, ist aber nicht mehr klar wie das ging!!! (4.96)

Für die nicht lineare Optimierung wird das gesamte Polynom aufgeteilt, so minimieren wir zunächst $\frac{z^T A z}{z^T B z}$ und danach $\frac{z^T A' z}{z^T B' z}$. So entstehen für z zunächst zwei Lösungen \hat{z}_1 und \hat{z}_2 , welche über eine Mittelung eine ersten Schätzung für z geben, welche schon ziemlich nah an den optimalen Wert heranreicht.

$$z = \frac{\frac{\hat{z}_1}{\|\hat{z}_1\|} + \frac{\hat{z}_2}{\|\hat{z}_2\|}}{2} \quad (4.97)$$

Da es sich um eine nicht lineare Optimierung handelt ist die Minimierung von $\frac{z^T A z}{z^T B z}$ gleichzusetzen mit der Maximierung von $\frac{z^T B z}{z^T A z}$. Beide als eine Funktion von $f(z)$. Matrix A wird mit der Choleskyzerlegung in zwei höhere Dreiecksmatrizen zerlegt $A = D^T D$ [24]. Dies geht nur da A nachweislich eine symmetrische und positiv-definite Matrix ist.[24] positiv-Definite bedeutet, dass die Singulärwerte von A immer positiv bleiben, egal mit welchem Vektor z diese multipliziert wird. (**HIER NOCH LITERATUR FINDEN UND NOCHMAL PRÜFEN OB DEFINITION SO STIMMT**). Des Weiteren wir definiert, dass $y = Dz$ ist und $f(z)$ wird dann zu einen $\hat{f}(y)$

$$A = D^T D \quad (4.98)$$

$$y = Dz \rightsquigarrow z = D^{-1}y \quad (4.99)$$

$$f(z) = \frac{z^T B z}{z^T A z} \quad (4.100)$$

$$\rightsquigarrow f(z) = \frac{z^T B z}{z^T D^T D z} \quad (4.101)$$

$$\hat{f}(y) = \frac{y^T D^{-T} B D^{-1} y}{y^T y} \quad (4.102)$$

Durch die Defintion von $y = Dz$ ist y bis auf einen Skalierungsfaktor definiert. $\hat{f}(y)$ ist maximiert, wenn y gleich dem Eigenvektor von $D^{-T} B D - 1$ ist, welcher mit dem größten Eigenwert assoziiert wird. Zum Schluss erhalten wir dann einen Wert für \hat{z}_1 mit $\hat{z}_1 = D^{-1}y$. Exakt das selbe Verfahren wird für die Findung von z_2 mit $\frac{z^T B' z}{z^T A' z}$ angewandt. Sind z_1, z_2 und eine erste Schätzung für z gefunden, so kann ein Wert für z gesucht werden, welcher noch näher an ein optimales Ergebnis heranreicht. Beide Lösungen z_1 und z_2 , werden in die Funktion $f(z)$ eingesetzt und es jeweils ein wert ermittelt, welcher am nächsten an einem Nullpunkt sich befindet. So kann iterativ eine optimale Lösung für z gefunden werden. Ist der Wert für z bestimmt, so kann dieser die Gleichungen 4.75 und 4.76 eingesetzt werden und w beziehungsweise w' bestimmt werden, welche die Elemente für die Matritzen H_p und H'_p bereitstellen. Abbildung 4.7 zeigt in rot den Quader des Minimalbeispiels wie er in Kamera zwei abgebildet ist und in grün wie er in Kamera eins abgebildet ist. Kamera zwei ist horizontal zu Kamera eins verschoben und um 45° zu Kamera eins um die eigene vertikale Achse gedreht. Die Auflösungen beider Kameras sind identisch, sprich die intrinsischen Kameraparameter sind die selben. Abbildung 4.8 zeigt die momentanen Epipolarlinien. Die Epipolarlinien von Bild eins, also dem grünen Abbild, sind bereits Parallel, was aber keine Voraussetzung für die Funktion des Rektifizierungsalgorithmus ist. Der Schnittpunkt der Epipolarlinien von Bild zwei, also dem Roten Abbild, treffen sich in einem Punkt und bilden somit den Epipol von Bild zwei.

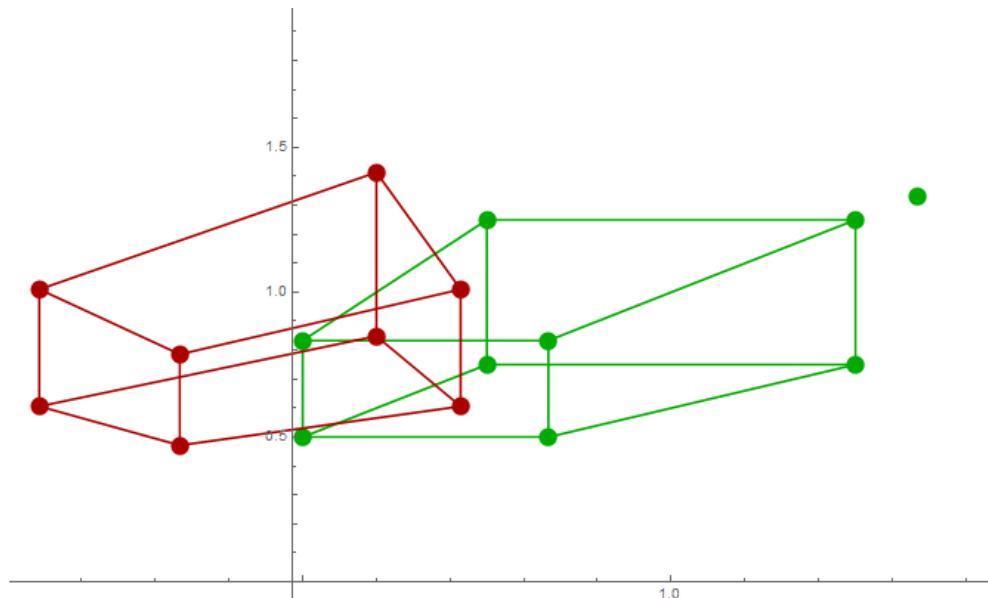


Abbildung 4.12: Aufnahmen zweier Kameras mit den selben Auflösungen, Kamera eins(Grün) und Kamera(rot) zwei gelten jeweils $\zeta = 1$

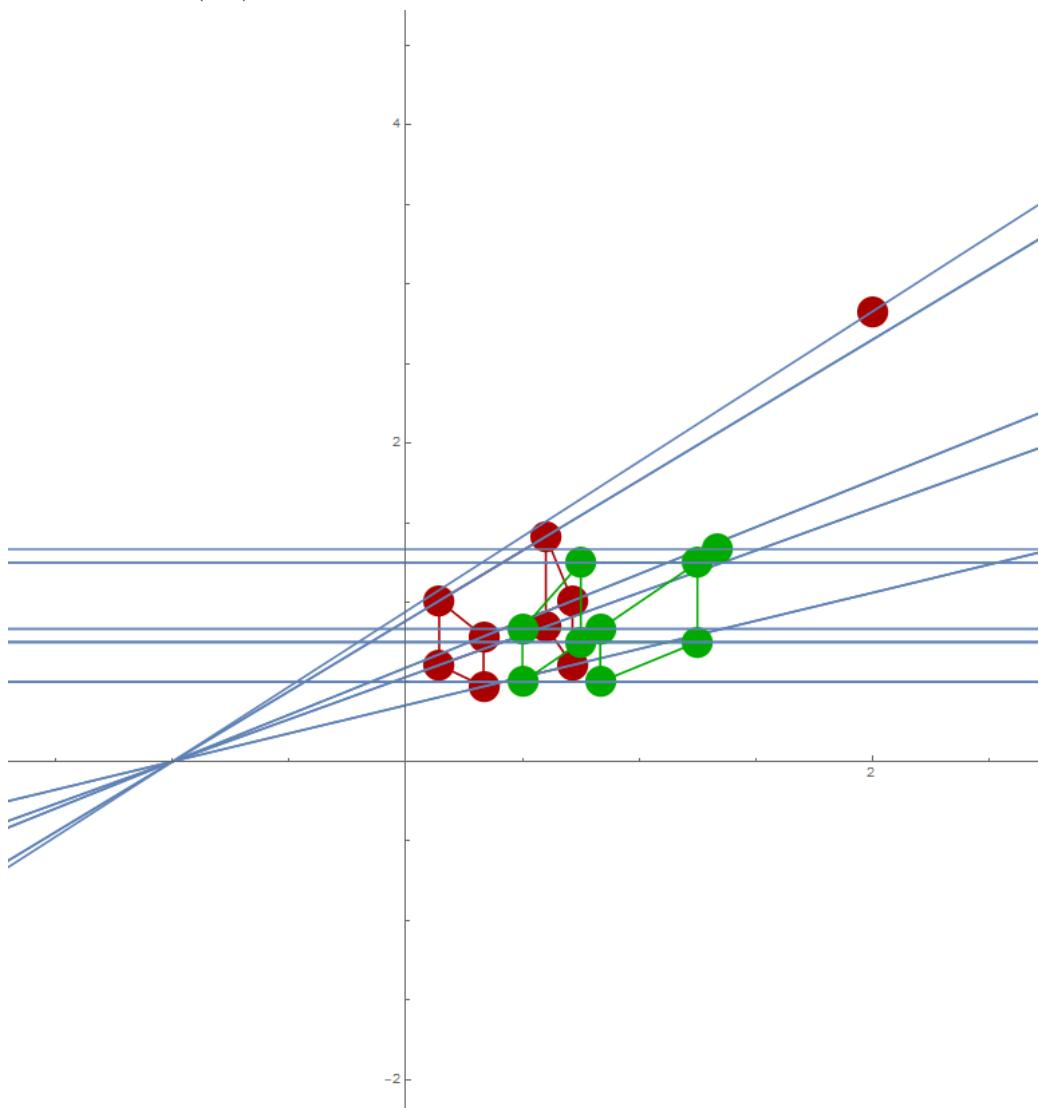


Abbildung 4.13: Epipole für Kamera eins und Kamera zwei vor der Rektifizierung

Werden nun die Matrizen H_p und H'_p auf die jeweiligen Punkte der Bilder, p_i für Bild eins und p_j für Bild zwei, angewandt, so kann man eine erste Veränderung beobachten. Abbildung 4.9 zeigt beide Quader aus Abbildung 4.7 nachdem die jeweiligen Bildpunkte mit den projektiven Matrizen multipliziert wurden. Der Epipol in Bild eins bleibt natürlich wie zuvor im unendlichen, jedoch kann man erkennen, dass der rote Quader aus Bild zwei sich verändert hat. Sein Epipol wurde ins Unendliche transformiert und parallele Linien sind nun auch auf dem Bild parallel. Das die Epipolarlinien bereits horizontal parallel zur x-Achse verlaufen ist Zufall und ist nach der Anwendung der projektiven Matrizen auch noch nicht verlangt. Das Anpassen der Epipolarlinien, dazu gehört sie zunächst von beiden Bildern parallel zur x-Achse verlaufen zu lassen und dann noch sie so zueinander anzupassen, dass sie zu Scanlinien über beide Bilder verlaufen, vergleiche Abbildung 4.12, folgt im nächsten Schritt.

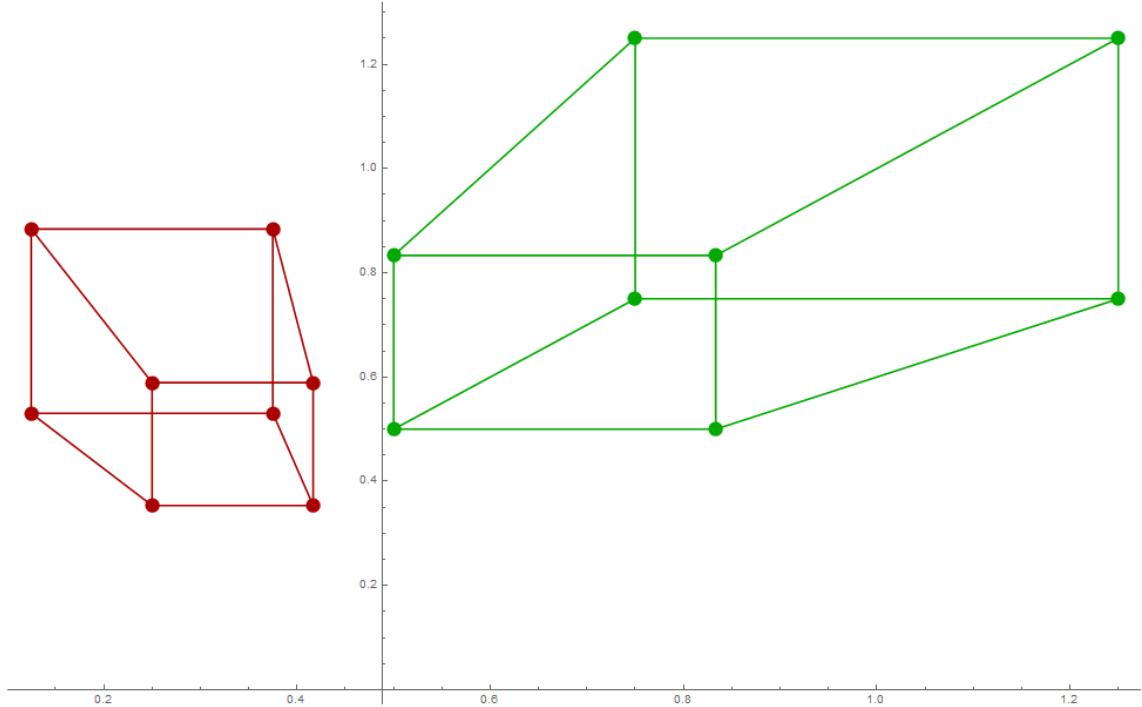


Abbildung 4.14: Abbildung beider Bilder nach anwenden der Matrizen H_p und H'_p . Die Epipole bei den Bildern sind nun im unendlichen. Das die entstehenden parallelen Epipolarlinien auch hier schon horizontal ausgerichtet sind ist Zufall. Die Epipolarlinien sind immer parallel nach dieser Transformation aber die Richtung ist nicht immer automatisch bereits $i = [1,0,0]$.

4.10.2 Ähnlichkeitstransformation

Nachdem die Epipole ins Unendliche verschoben wurden, müssen diese nun so rotiert und verschoben werden, dass die Epipolarlinien als Richtung $i = [1 \ 0 \ 0]$ haben und die Epipolarlinien beider Bilder zu einheitlichen Scanlinien werden. Für die Ähnlichkeitstransformation wird davon ausgegangen, dass w und w' bereits bekannt sind. H_r und H'_r wurden bereits aus der Zerlegung von H_a und H'_a gewonnen.

$$H_r = \begin{bmatrix} v_b - v_c w_b & v_a - v_c w_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad (4.103)$$

$$H'_r = \begin{bmatrix} v'_b - v'_c w'_b & v'_a - v'_c w'_a & 0 \\ v'_a - v'_c w'_a & v'_b - v'_c w'_b & v'_c \\ 0 & 0 & 1 \end{bmatrix} \quad (4.104)$$

$$(4.105)$$

w und w' sind bereits bekannt. Mit Hilfe von F , können v_a und v_b ersetzt werden. Dazu kann die letzte Zeile von F nach v_a, v_b und v_c aufgelöst werden. Für v'_a, v'_b und v'_c wird die letzte Spalte von F verwendet. So können folgende Gleichungen für $v_a, v'_a, v_b, v'_b, v_c$ und v'_c gewonnen werden.

$$F = H'^T[i]_x H \quad (4.106)$$

$$F = \begin{bmatrix} v_a w'_a - v'_a w_a & v_b w'_a - v'_a w_b & v_c w'_a - v'_a \\ v_a w'_b - v'_b w_a & v_b w'_b - v'_b w_b & v_c w'_b - v'_b \\ v_a - v'_c w_a & v_b - v'_c w_b & v_c - v'_c \end{bmatrix} \quad (4.107)$$

$$v_a = F_{31} + v'_c w_a \quad (4.108)$$

$$v_b = F_{32} + v'_c w_b \quad (4.109)$$

$$v_c = F_{33} + v'_c \quad (4.110)$$

$$v'_a = v_c w'_a - F_{13} \quad (4.111)$$

$$v'_b = v_c w'_b - F_{23} \quad (4.112)$$

$$v'_c = v_c - F_{33} \quad (4.113)$$

Eingesetzt in die jeweiligen Matrizen H_r und H'_r , entstehen die folgenden Matrizen in Gleichungen 4.114 und 4.115, welche nur noch die unbekannte v'_c beinhalten. Die gemeinsame Variable v'_c zeigt die geometrische Verbindung beider Bilder in ihrer Verschiebung entlang ihrer v-Richtung. Es wird also ein Offset von F_{33} benötigt, um die Epipolarlinien horizontal zu Scanlinien auszurichten. **Den Wert für v_c wird so ermittelt, dass das Minimum einer v-Koordinaten eines Pixel als minimum den Wert null besitzt**

$$H_r = \begin{bmatrix} F_{32} - w_b F_{33} & w_a F_{33} - F_{31} & 0 \\ F_{31} - w_a F_{33} & F_{32} - w_b F_{33} & F_{33} + v'_c \\ 0 & 0 & 1 \end{bmatrix} \quad (4.114)$$

$$H'_r = \begin{bmatrix} w'_b F_{33} - F_{23} & F_{13} - w'_a F_{33} & 0 \\ w'_a F_{33} - F_{13} & w'_b F_{33} - F_{23} & v'_c \\ 0 & 0 & 1 \end{bmatrix} \quad (4.115)$$

Das Ergebnis der Bildpunkte p_i und p_j multipliziert mit den Matrizen $H_r H_p$ und $H'_r H'_p$ mit ist in Abbildung 4.10 zu sehen. Als letztes folgt noch die Scherungstransformation H_s und H'_s für die horizontale Entzerrung beider Bilder.

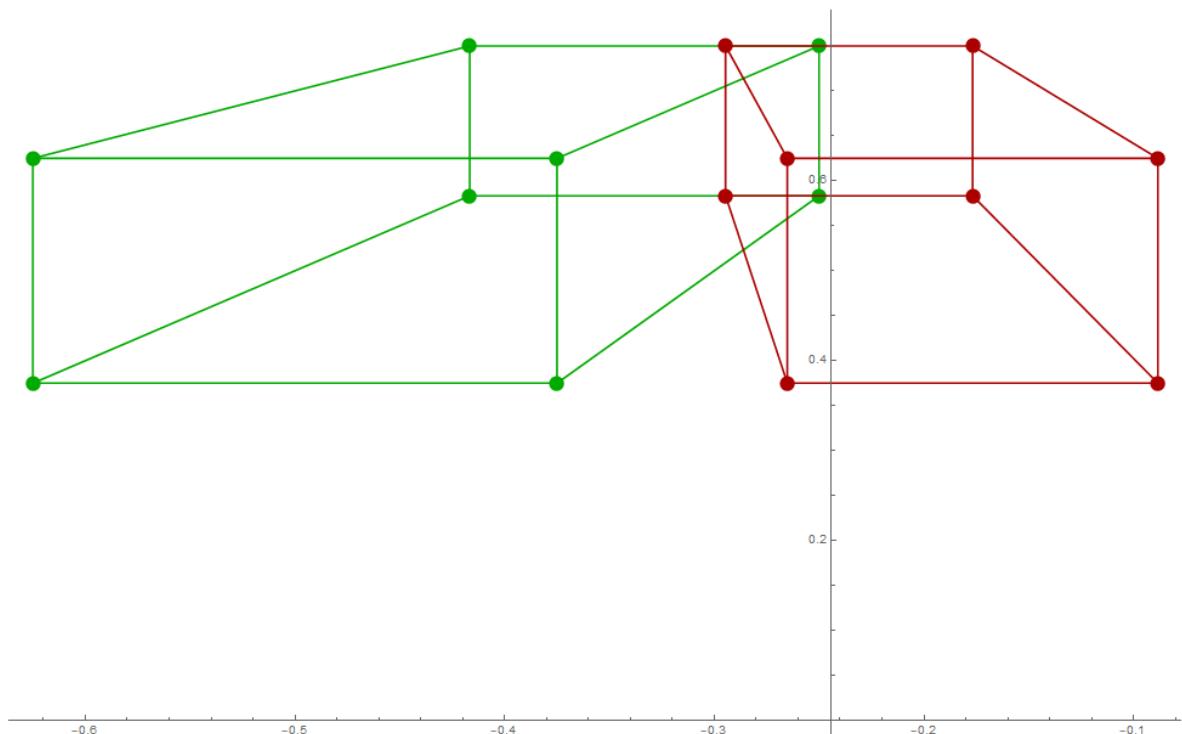


Abbildung 4.15: Abbildung beider Bilder nach anwenden der Matrizen $H_r \cdot H_p$ und $H'_r \cdot H'_p$. Die Epipolarlinien sind nun horizontal zueinander ausgerichtet

4.10.3 Scherungstransformation

?????

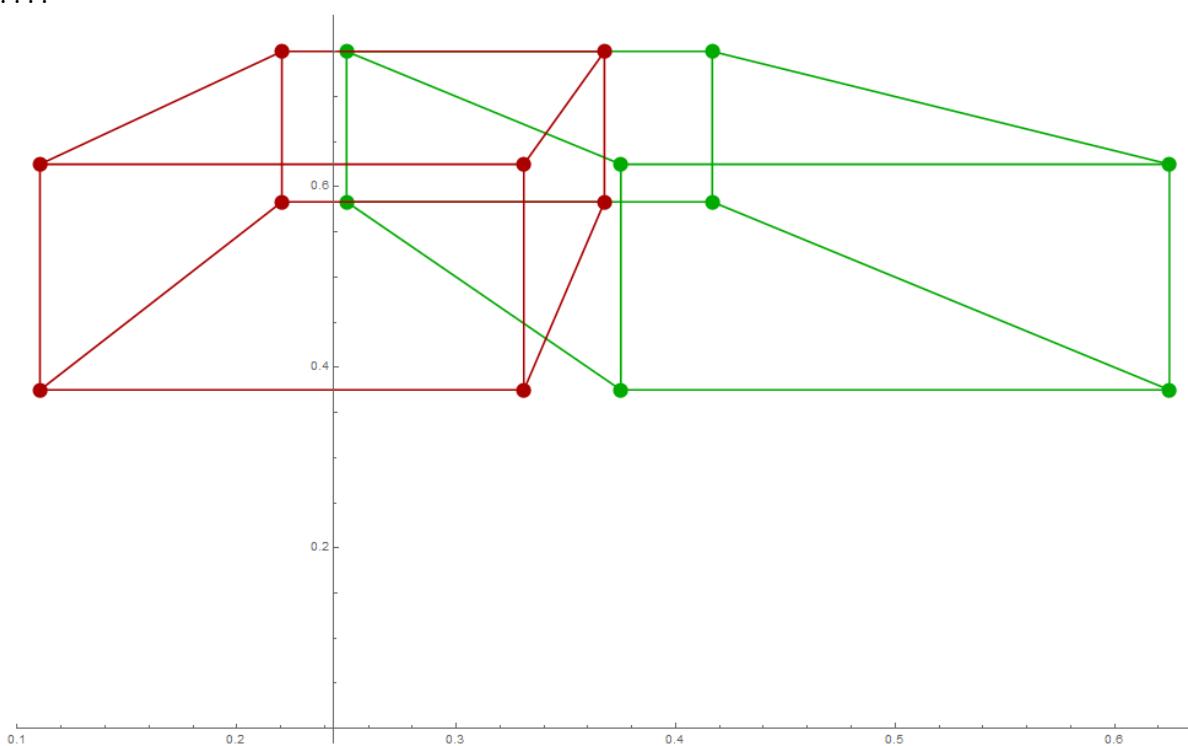


Abbildung 4.16: Abbildung beider Bilder nach anwenden der Matrizen $H_s \cdot H_r \cdot H_p$ und $H'_s \cdot H'_r \cdot H'_p$. Die horizontale Verzerrung wurde reduziert.

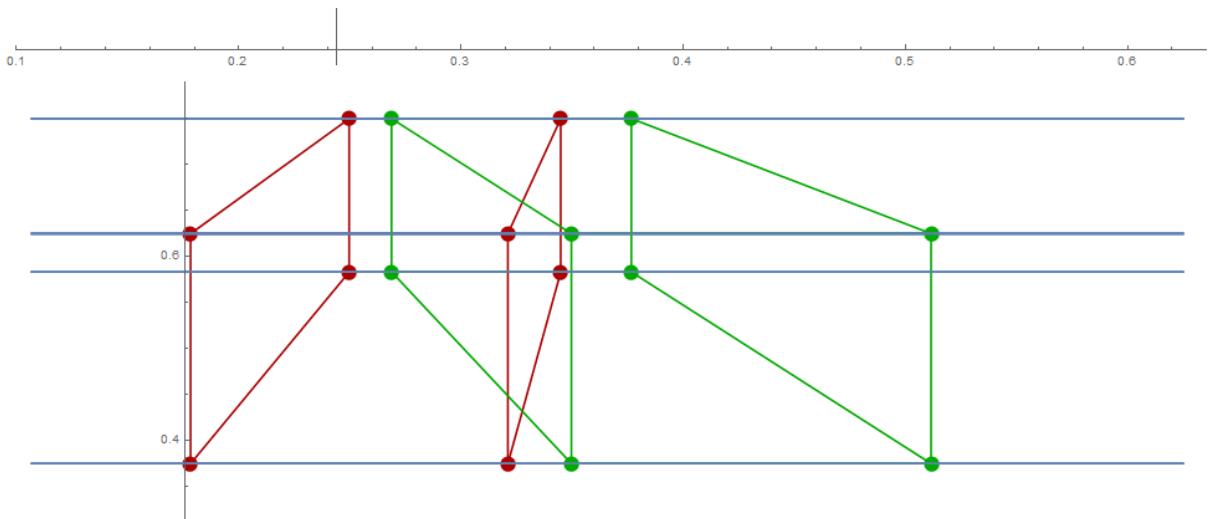


Abbildung 4.17: In dieser Abbildung wurden die Epipolarlinien noch in den Grafikplot mit eingebaut

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

Wenn man sich mit digitalen Bilddaten auseinandersetzt, so kommt man nicht drum herum sich auch mit den verschiedenen Auflösungsarten beschäftigen zu müssen.

Im vorherigen Kapitel wurde gezeigt, wie eine 3D-Szene aus zwei heterogenen Bildquellen, welche von zwei Kameras gleicher Auflösung aufgenommen wurden, rekonstruiert werden konnte und gleichzeitig noch die externen Kameraparameter von C' in Relation zu C geschätzt werden konnten. Dies wirft die Frage auf, welche Auswirkungen Bilder zweier Kameras mit unterschiedlichen Auflösungen für die Rekonstruktionen haben können. Aus der Stereokalibrierungsapp, welche *MatLab* anbietet, ist bekannt, dass diese nicht mit Bildern unterschiedlicher Auflösung eine Szenenrekonstruktion durchführen kann. Der erste Schritt bestand erstmal darin zu überprüfen, warum zwei unterschiedliche Auflösungen in *MatLab* Probleme machen. *MatLab* verfolgt einen etwas anderen Rekonstruktionsansatz. Zu aller erst werden die Kameras kalibriert. Dies geschieht über die Matlab-Funktion `estimateCameraParameters`[25]. Diese Funktion funktioniert auch bei Bildern unterschiedlicher Auflösung noch ohne Probleme. Das Problem, welches sich als eigentlich minimales Problem herausstellt, ist, dass die darauf folgenden Rektifizierung der Stereobilder nicht mit zwei Bildern unterschiedlicher Auflösung funktioniert. In den *MatLab* references, steht es nicht explizit drin [16]. Die Rektifizierung in Matlab funktioniert nach einem Schema, welches ähnlich dem aufgezeigten Beispiel in Kapitel (KAPITELLINK REKTIFIZIERUNG) bereits erwähnt wurde und in [19, 21] nochmal genau aufgeführt wird. Das Problem liegt also nicht daran, dass Bilder unterschiedlicher Auflösung nicht rektifiziert werden können, sondern das Problem liegt an dem in *MatLab* verwendeten Algorithmus für die Rektifizierung zweier Stereobilder (Foren Zitieren????). Warum *MatLab* überhaupt rektifiziert, liegt daran, dass ein Ansatz der Szenenrekonstruktion gewählt wurde, welcher die essentielle Matrix nicht benötigt. In diesem Falle, werden zwei Stereobilder aufgenommen, danach rektifiziert und anschließend über eine sogenannte *Disparity-Map*, die Szenenpunkte rekonstruiert[26, 27, 19, 18]. Der in dieser Arbeit gewählte Rektifizierungsalgorithmus, ist nicht auf gleiche Kameraauflösungen beschränkt. Mittlerweile gibt es natürlich deutlich fortgeschrittenere Rektifizierungsansätze, jedoch wurde für diese Arbeit der Ansatz von [17] gewählt, um ein Gefühl zu vermitteln, dass wenn man sich auf die Epipolargeometrie bezieht, die Auflösungen der Kameras keine Rolle spielen[3]. Was genau unterschiedliche Auflösungen der Kameras für die einzelnen Bilder bedeutet und was genau sich bei der Aufnahme mit dem Sensor dabei ändert, soll im folgenden Unterkapitel kurz erläutert werden. Danach wird aufgezeigt, was genau diese Veränderungen für die Epipolargeometrie bedeuten und letztendlich wird das Minimalbeispiel so angepasst, als hätten zwei Kameras unterschiedlicher Auflösung die selbe Szene wie davor aufgenommen und die Resultate miteinander verglichen.

5.1 Abbildungsunterschiede bei unterschiedlichen Kameraauflösungen

Das Herz einer modernen Kamera ist der Halbleiter-Bildsensor. Die Bildsensoren spielen in der optischen Messtechnik und industriellen Bildverarbeitung eine große Rolle[7]. Die in dieser Arbeit verwendeten Kameras für die Aufnahme der Stereobildpaare, waren zum einen die Vollformatkamera Canon EOS 6D und die Halbformatkamera Canon EOS 60D. Beide besitzen einen CMOS-Sensorchip, welcher zu den Halbleiterbildsensoren gehört[28]. Die Geometrie dieser Sensoren, ist grob gesprochen als eine $M \times N$ -Matrix, bestehend aus $M \times N$ Pixeln. Die Pixel bedecken nur einen Teil der Sensorfläche und können in ihrer geometrischen Beschaffenheit von der Form des Sensors unterschieden werden[7]. Abbildung 5.1 zeigt einen schematischen Aufbau eines Halbleiterbildsensors

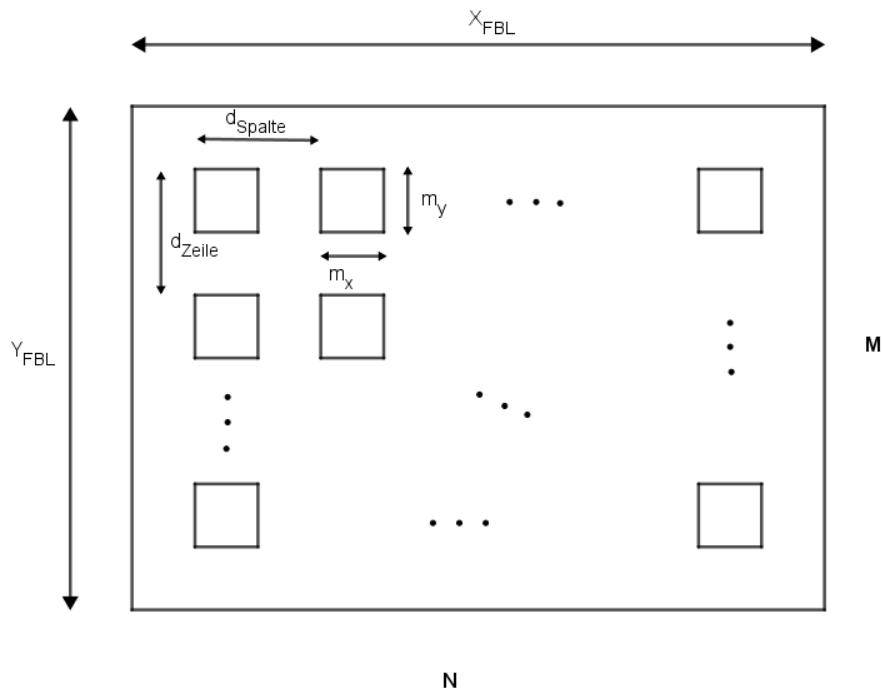


Abbildung 5.1: Rechteckiger Bildsensor mit darauf sich befindenden quadratischen Pixeln

Die Auflösung des Sensors hängt von den horizontalen und vertikalen Pixelabständen ab und gibt wieder, welche Objektdetails mit dem Sensor gerade noch erkannt werden können.[7]. Ein Sensor hat eine maximale Auflösung, welche durch die Anzahl seiner fest installierten Pixel bestimmt wird. Die Bildqualität ist abhängig von der Größe des Sensorchips und der Menge der sich darauf befindenden Pixel. Der CMOS-Sensor einer Canon 6D hat beispielsweise eine Größe von $36 \times 24\text{mm}$ und eine maximale Auflösung von 5.472×3.648 Pixel. Jedoch ist das nicht die einzige Auflösung, welche beim fotografieren oder filmen mit der Kamera genutzt werden kann. Es können sowohl die Pixelanzahl als auch das Seitenverhältnis der entstehenden Bilder eingestellt werden. Bei der Canon EOS 6D können insgesamt vier verschiedene Seinsverhältnisse eingestellt werden [3 : 2], [4 : 3], [16 : 9] und [1 : 1][28]. Die Bildauflösungen unterscheiden sich pro Seitenverhältnis in sechs Auflösungseinstellungen L , M , $S1$, $S2$, $S3$.

Tabelle 5.1: Auflösungen Canon EOS 6D

	3:2	4:3	16:9	1:1
L	5478×3648 px	4864×3648 px	5472×3072 px	3648×3648 px
M	4104×2736 px	3648×2736 px	4104×2310 px	2736×2736 px
$S1$	2736×1824 px	2432×1824 px	2736×1536 px	1824×1824 px
$S2$	1920×1280 px	1696×1280 px	1920×1080 px	1280×1280 px
$S3$	720×480 px	640×480 px	720×408 px	480×480 px

Je geringer die Auflösung, desto geringer ist die Anzahl der Pixel. Die Anzahl der Pixel auf einem Sensorchip kann natürlich nicht variieren. Eine geringere Pixelanzahl bei gleichbleibender Bildgröße, bedeutet, dass sich ein Pixel mit den um sich befindenden Pixeln interpoliert, so dass ein neuer Pixel bestehend aus mehreren kleinen Pixeln entsteht. Dieser Prozess wird Nachbarschaftsoperation genannt. Für die Berechnung des neuen Bildpixels px' an der Stelle (m, n) wird nicht nur das Bildpixel p des Originalbildes an der Stelle (m, n) verwendet, sondern auch einige seiner Nachbarpunkte[7]. Bei der Canon 6D bietet das Seitenverhältnis [3 : 2] die Möglichkeit die maximale Pixelanzahl des Sensors

zu verwenden, vergleiche hierzu Tabelle 5.1. Bei einem Seitenverhältnis von [4 : 3] ist die Anzahl der maximal möglichen Pixel nur noch 4864×3648 . Anders sich das Seitenverhältnis des Bildausschnitts, so wird auch nicht mehr die gesamte Fläche des Sensors benutzt.

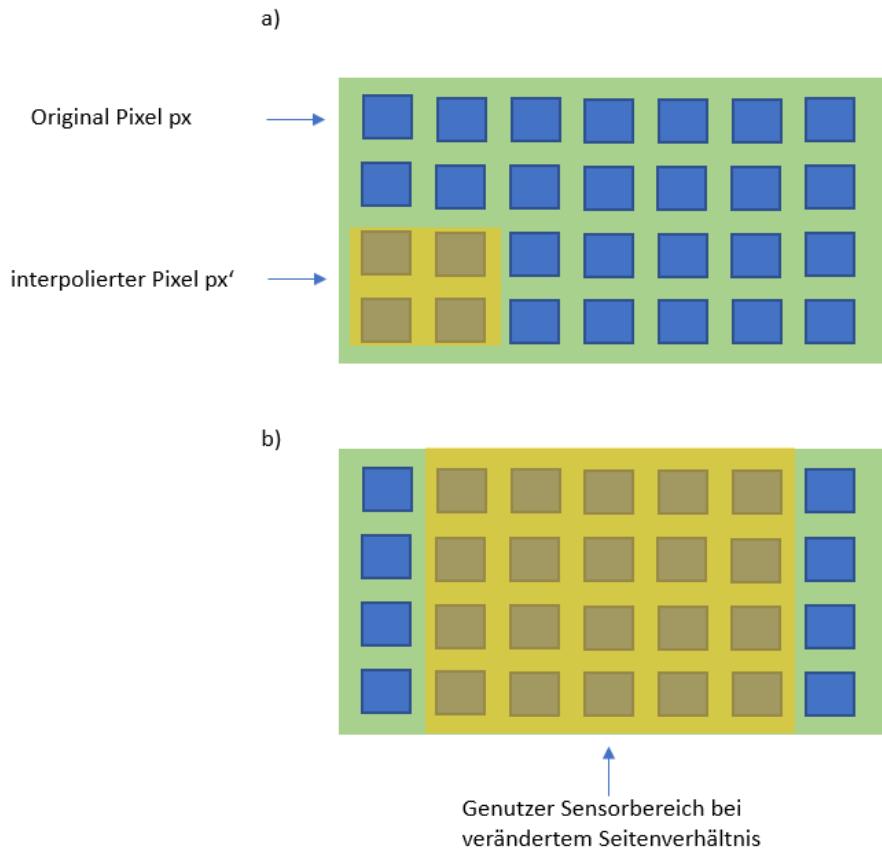


Abbildung 5.2: Bild a) zeigt die Interpolation von Pixeln, wenn bei gleichbleibenden Seitenverhältnissen weniger Pixel für das Bild verwendet werden sollen. Die interpolierten Pixel leiten dann alle das selbe Signal weiter. Bild b) zeigt in gelb markiert, den verwendeten Bereich des Sensors, wenn sich die Seitenverhältnisse ändern und nicht mehr der volle Sensor genutzt wird.

5.2 Auswirkung unterschiedlicher Auflösungen auf die Epipolargeometrie

Um nun auf die Fragestellung der Auswirkung unterschiedlicher Auflösungen auf die Szenenrekonstruktion zu kommen, kann nun folgende Behauptung aufgestellt werden. Beide Kameras besitzen einen Bildsensor, welcher fest in der Kamera installiert ist und sich weder in Position noch seiner Form ändern kann. Dieser Bildsensor beinhaltet sowohl das Bildebenenkoordinatensystem, bei welchem der Hauptpunkt den Koordinatenursprung bildet als auch das Sensorkoordinatensystem, dess Ursprung leicht außerhalb einer der Ecken des Sensors sich befindet. Abbildung ??? zeigt einen stereoskopischen Szenenaufbau mit Kameras gleicher Auflösung. Die Sensorkoordinatensysteme besitzen die selbe Skalierung.

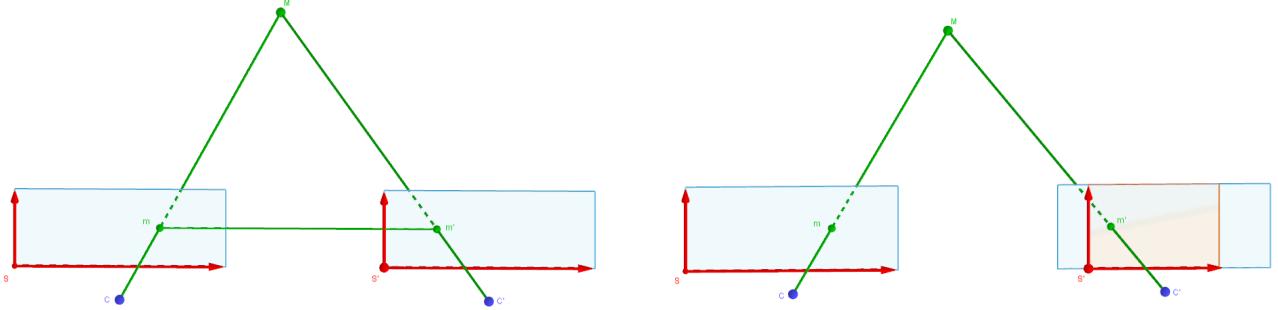


Abbildung 5.3: C und C' haben die selbe Auflösung
eingestellt

Abbildung 5.4: C und C' haben unterschiedliche
Auflösungen eingestellt

Es bildet sich wieder das bekannte Dreieck zwischen den Bildpunkten m_σ und m'_σ und dem Objektpunkt M_δ . Das in diesem Falle einen korrekten Szenenrekonstruktion funktioniert, ist in Kapitel (MINIMALBEISPIEL) anhand des Minimalbeispiels aufgezeigt worden. Wird die Auflösung auf einer der beiden Kameras verringert und damit einhergehend auch noch das Seitenverhältnis geändert, so ändert sich zum einen der aktive Teilbereich des Sensorschips, sowie die Skalierung der Werte auf den Koordinatenachsen des Sensorkoordinatensystems. Die Skalierung der Koordinatenachsen hängt mit der Interpolation der mehreren Pixel zu einem neuen Pixel zusammen. Abbildung 5.3 zeigt schematisch, was sich nach veränderten Auflösungseinstellungen am Sensor verändert.

(GRAFIK EINFÜGEN)

Epipolaremetrisch ändert sich wie man in Abbildung ??? sehen kann nichts. Das zuvor erwähnte Dreieck zwischen den Bildpunkten und dem Objektpunkt bleibt unverändert. Wie in Kapitel (EPI-POLARGEOMETRIE) bereits erwähnt, dürfen die Bildkoordinatensysteme und somit auch die Sensorkoordinatensysteme unterschiedlich voneinander sein[3]. Für die relative Position des Bildpunktes auf dem Sensor ändert sich nichts, dieser bleibt statisch, einzige seine Koordinatenwerte ändern sich im Bezug auf das Sensorkoordinatensystems. Für die Fundamentalmatrix und die Essentielle Matrix ergeben sich lediglich andere Vielfache voneinander, welche wie erwähnt ebenfalls gültige Lösungen sind [4, 14].

5.3 Minimalbeispiel mit unterschiedlichen Auflösungen

Als Beweise der aufgestellten Behauptung wurde im Minimalbeispiel die Kameramatrix einer der beiden Kameras unterschiedlich modifiziert. Während für die Kameramatrix von C der Wert $\zeta = -1$ in der Kameramatrix K bleibt, wurde das ζ in C' verändert, so dass sie drei verschiedene neue Kameramatrizen K'_1, K'_2 und K'_3 ergeben. Die resultierenden Abbildungen des Quaders sind in den Abbildungen ??? bis ??? zu sehen.

$$K = \begin{bmatrix} -1 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.1)$$

$$K'_1 = \begin{bmatrix} -2 & 0 & 0 & 0 \\ 0 & -2 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.2)$$

$$K'_2 = \begin{bmatrix} - & 0 & 0 & 0 \\ 0 & -3.2 & 0 & 0 \\ 0 & 0 & -1.2 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.3)$$

$$K'_3 = \begin{bmatrix} -0.5 & 0 & 0 & 0 \\ 0 & -4.3 & 0 & 0 \\ 0 & 0 & -1 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.4)$$

(5.5)

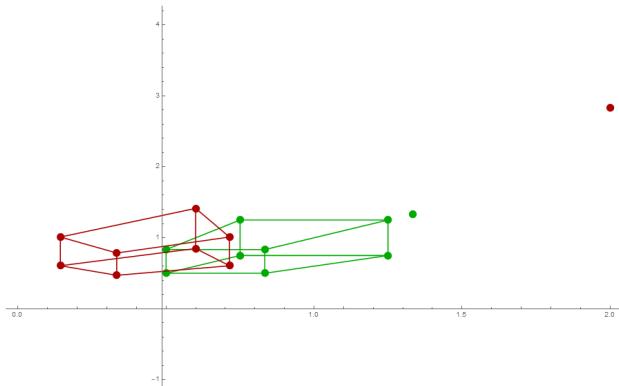


Abbildung 5.5: C und C' haben die selbe Auflösung eingestellt

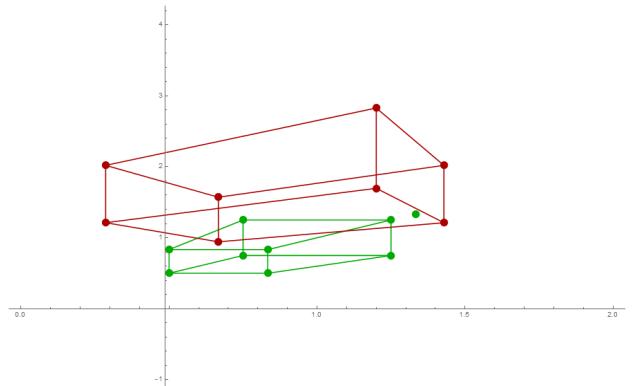


Abbildung 5.6: C und C' haben unterschiedliche Auflösungen eingestellt

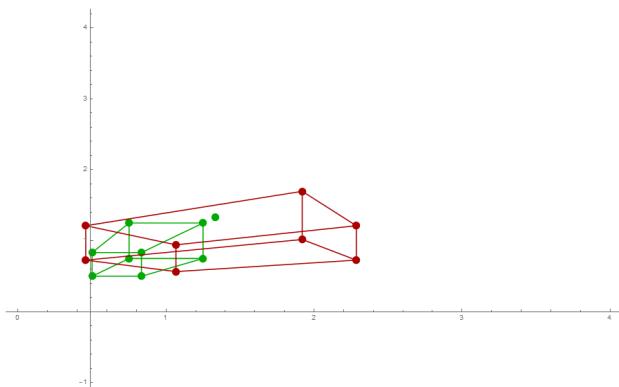


Abbildung 5.7: C und C' haben die selbe Auflösung eingestellt

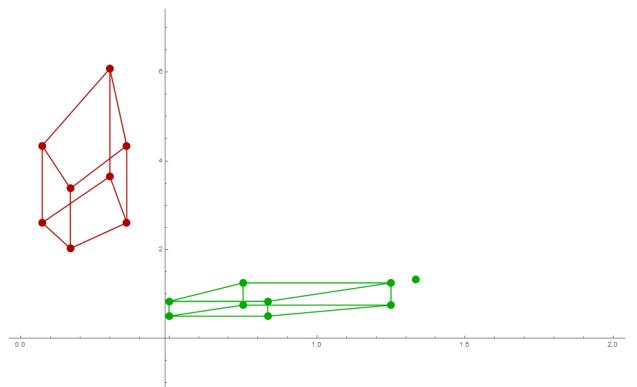


Abbildung 5.8: C und C' haben unterschiedliche Auflösungen eingestellt

Während die Abbildung von C Unverändert bleibt, wird in Abbildung ??? Die Abbildung des Quaders in C' "vergrößert", was für eine höhere Anzahl an verwendeten Pixeln steht. In Abbildung ??? werden

die Pixel in horizontaler Richtung um das 3.2-fache und in vertikaler Richtung um das 1.2-fache erweitert und in Abbildung ??? wird in horizontaler Richtung die Anzahl der Pixel um das 0.5-fache und in vertikaler Richtung um das 4.3-fache skaliert. Für die Fundamentalmatrizen und die essentielle Matrix ergeben sich verglichen mit denen aus dem Beispiel mit gleicher Abbildung folgende Matrizen.

(SCREENSHOT DER MATRIZEN IM VERGLEICH UND ZEIGEN DAS WIE VIEL FACHE DIE EINZELNEN ZEILEN JEWELS VOM ORIGINAL SIND)

$$\text{zeta_x} = 1; \quad F_1 = \begin{pmatrix} 0 & -0.5 & 0 \\ 0 & 0 & 0.707 \\ 0 & -0.5 & 0 \end{pmatrix} | :0.5 \quad F_1 = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}$$

$$\text{zeta_x} = 2; \quad F_2 = \begin{pmatrix} 0 & 0.378 & 0 \\ 0 & 0 & -0.534 \\ 0 & 0.756 & 0 \end{pmatrix} | :0.756 \quad F_2 = \begin{pmatrix} 0 & 0.5 & 0 \\ 0 & 0 & -0.707 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\text{zeta_x} = 3.2; \quad F_3 = \begin{pmatrix} 0 & 0.198 & 0 \\ 0 & 0 & -0.747 \\ 0 & 0.634 & 0 \end{pmatrix} | :0.634 \quad F_3 = \begin{pmatrix} 0 & 0.312 & 0 \\ 0 & 0 & -1.178 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\text{zeta_x} = 0.5; \quad F_4 = \begin{pmatrix} 0 & 0.885 & 0 \\ 0 & 0 & -0.145 \\ 0 & 0.442 & 0 \end{pmatrix} | :0.442 \quad F_4 = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & -0.328 \\ 0 & 1 & 0 \end{pmatrix}$$

Abbildung 5.9: Die Fundamentalmatrizen sind bei jeder Auflösung, die gewählt wurden immer Vielfache voneinander

`zeta_x = 1;`
`zeta_y = 1;`

$$E_1 = \begin{pmatrix} 0 & -0.5 & 0 \\ 0 & 0 & -0.707 \\ 0 & 0.5 & 0 \end{pmatrix} | : 0.5$$

$$E_1 = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}$$

`zeta_x = 2;`
`zeta_y = 2;`

$$E_2 = \begin{pmatrix} 0 & 0.756 & 0 \\ 0 & 0 & 1.069 \\ 0 & -0.756 & 0 \end{pmatrix} | :-0.756$$

$$E_2 = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}$$

`zeta_x = 3.2;`
`zeta_y = 1.2;`

$$E_3 = \begin{pmatrix} 0 & 0.634 & 0 \\ 0 & 0 & 0.897 \\ 0 & -0.634 & 0 \end{pmatrix} | :-0.634$$

$$E_3 = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}$$

`zeta_x = 0.5;`
`zeta_y = 4.3;`

$$E_4 = \begin{pmatrix} 0 & 0.442 & 0 \\ 0 & 0 & 0.626 \\ 0 & -0.442 & 0 \end{pmatrix} | :-0.442$$

$$E_4 = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}$$

Abbildung 5.10: Die essentiellen Matrizen sind bei jeder Auflösung, die gewählt wurden immer Vielfache voneinander

Bei der Rekonstruktion der externen Kameraparameter ergibt sich daraus stehts die selbe Matrix für P' . Was wie gezeigt daran liegt, dass sich geometrisch nichts ändert, sondern lediglich die Skalierung der Koordinatenwerte der Bildpunkte und somit auch eine Skalierung der Einträge in F und E , welche aber ebenfalls als Skaleninvariant definiert sind[4].

(MATRIZEN ZEIGEN)???

Die Ergebnisse der darauffolgenden Szenenrekonstruktionen, der einzelnen Szenen zeigt, dass sich immer die selbe Szene ergibt, welche mit der eigens aufgebauten Szene übereinstimmen.

Reconstructed scaled Points 3D = $\begin{pmatrix} 3. & 3. & -4. \\ 5. & 3. & -4. \\ 5. & 5. & -4. \\ 3. & 5. & -4. \\ 3. & 3. & -6. \\ 5. & 3. & -6. \\ 5. & 5. & -6. \\ 3. & 5. & -6. \\ 8. & 8. & -6. \end{pmatrix}$

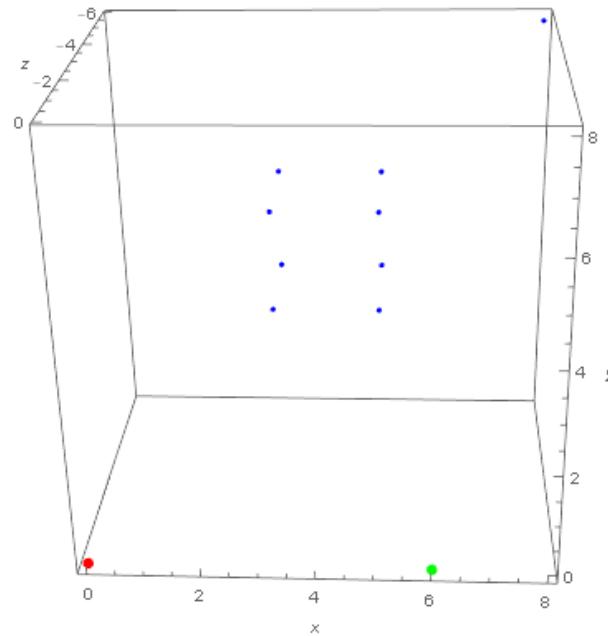


Abbildung 5.11: Die rekonstruierten Szenenpunkte und Kamerapositionen bleibt auch bei unterschiedlichen Auflösungen die selben

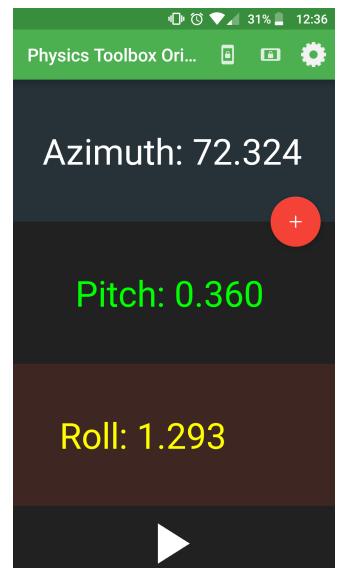
Die Behauptung, dass die Auflösung der Kamera bei dem in dieser Arbeit gewählten Workflow für die Rekonstruktion der Szene keine Auswirkung hat, kann für das Minimalbeispiel bestätigt werden.

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

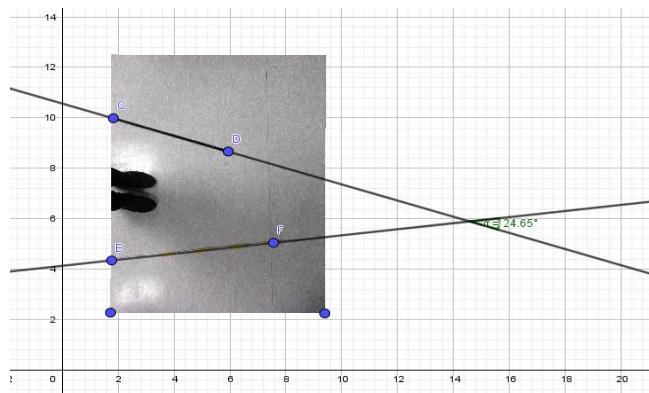
K in px	2663,969662	0	0
	0	2664,587487	0
	945,6292097	707,7870042	1
K in mm (transponiert)	17,3158028	0	6,146589863
	0	17,31981867	4,600615527
	0	0	1

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(-)

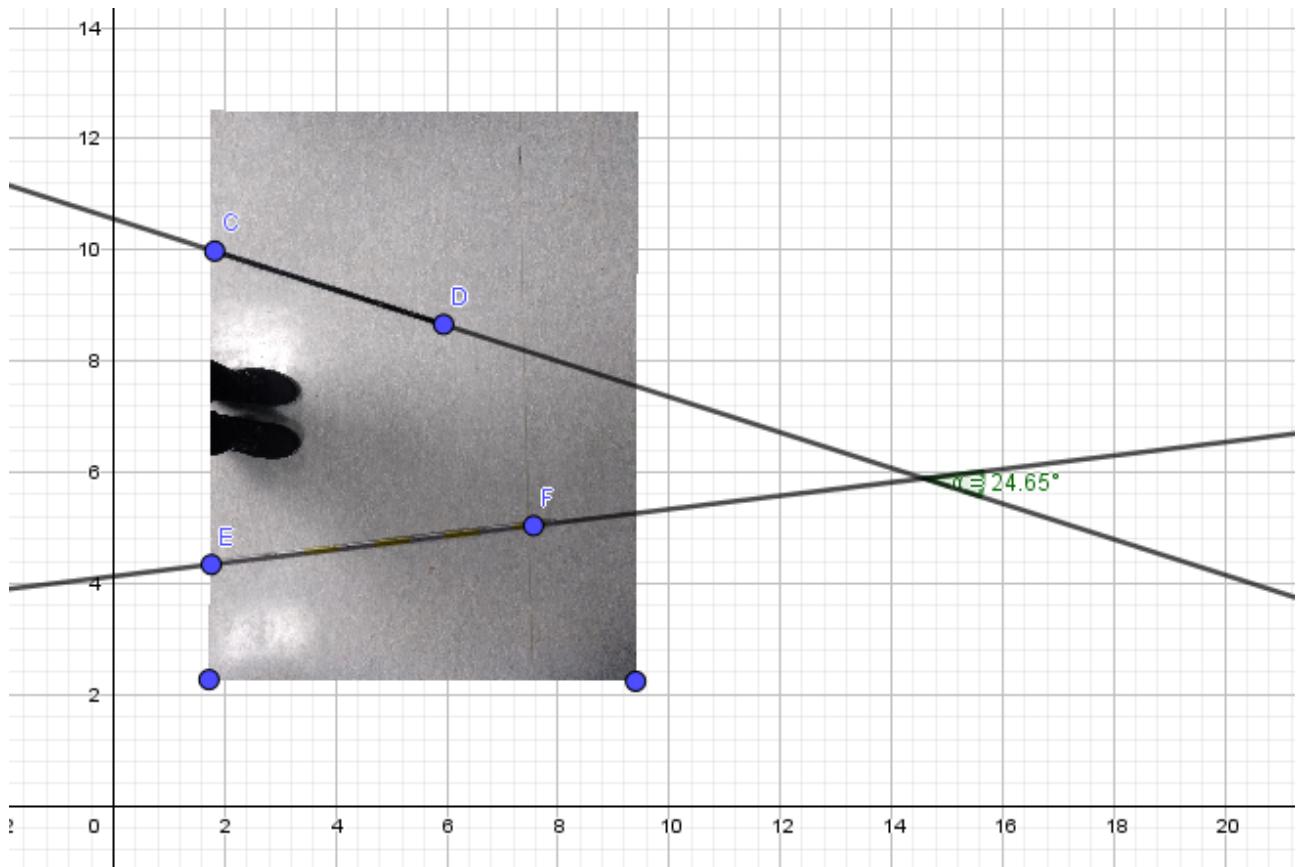


Abbildung 6.1: Rekonstruktion des Gier-Winkels der Kameras zueinander

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

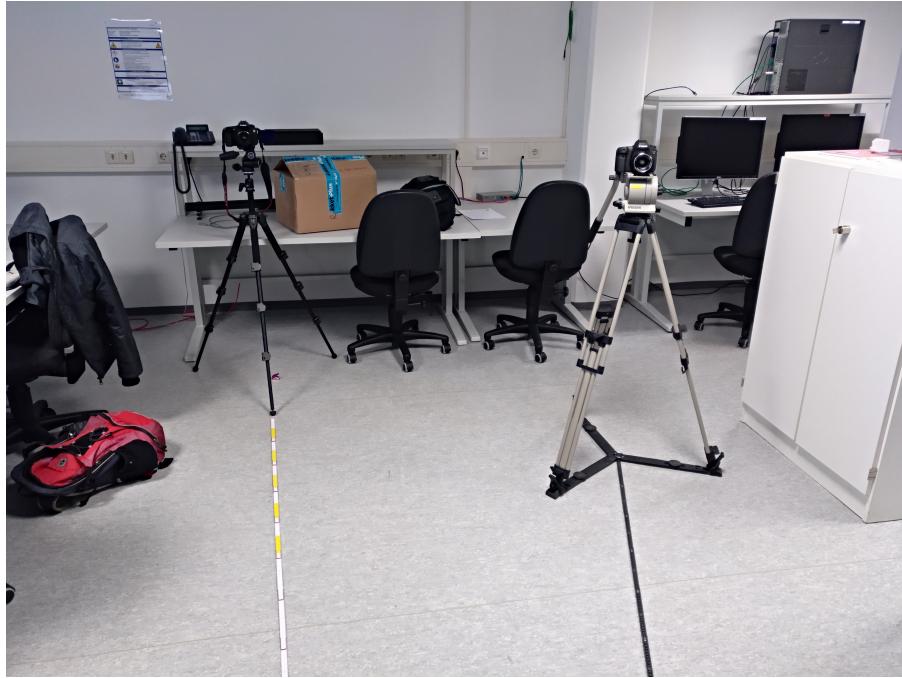


Abbildung 6.4: Stereosetup

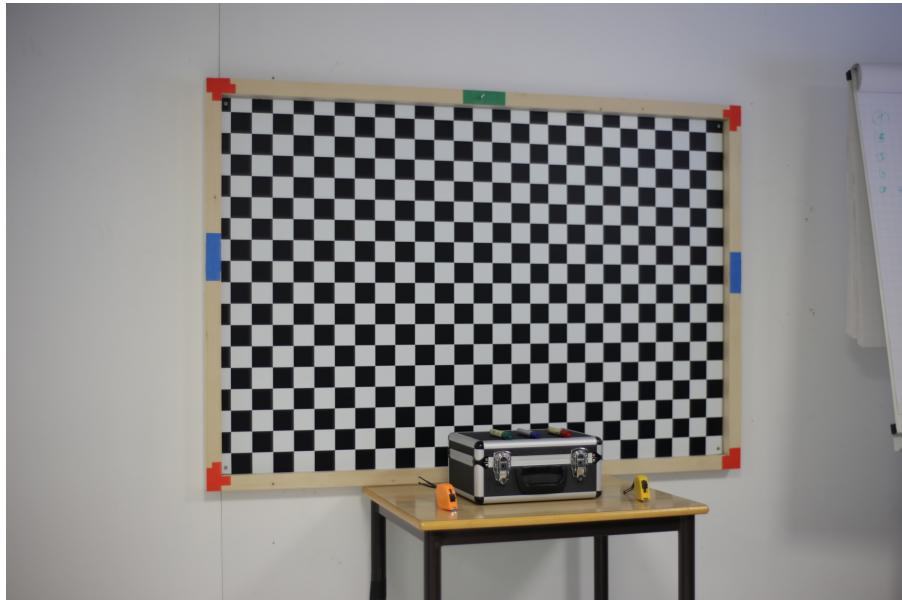


Abbildung 6.5: Szenenbild der primären Kamera 6D



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 [13], 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.[13]. 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[4].

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

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

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

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

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

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

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.

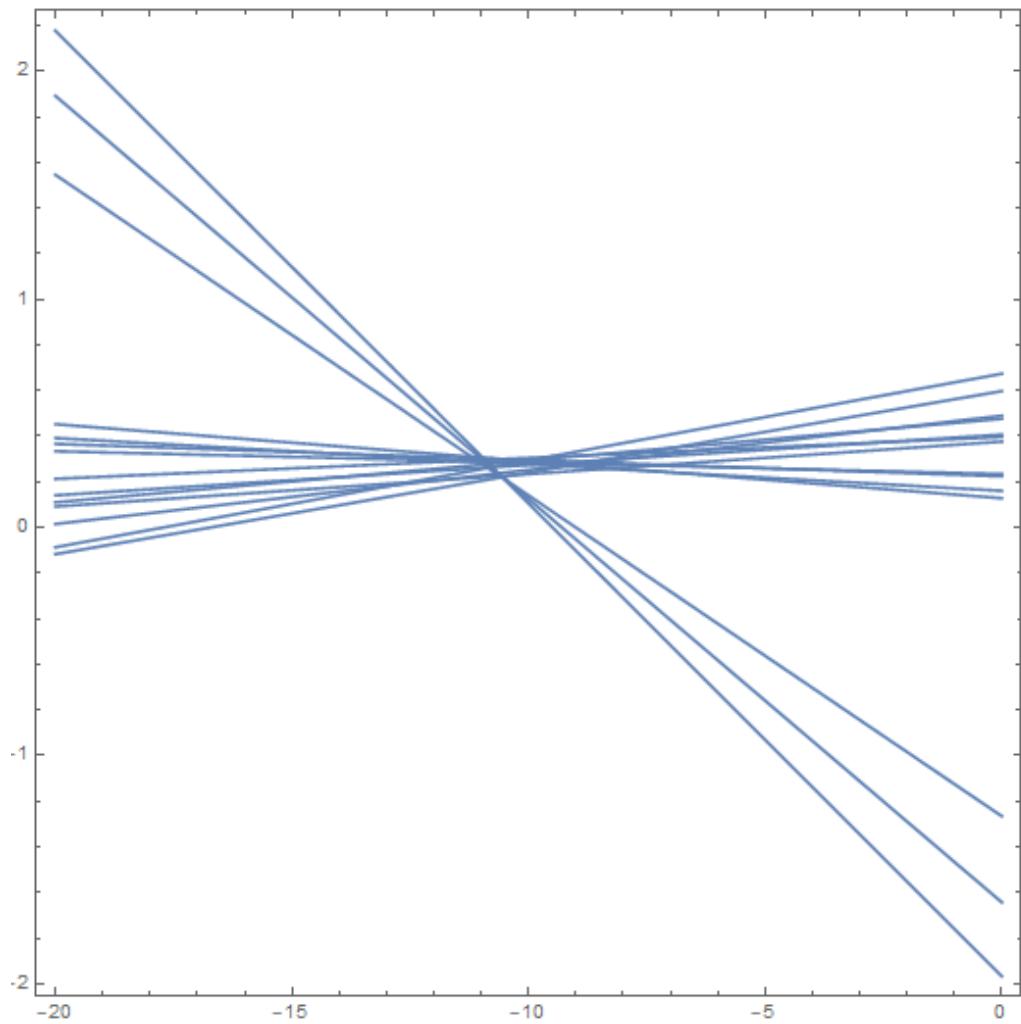


Abbildung 6.8: Resultierende Epipolarlinien des Bildes der Canon 6D

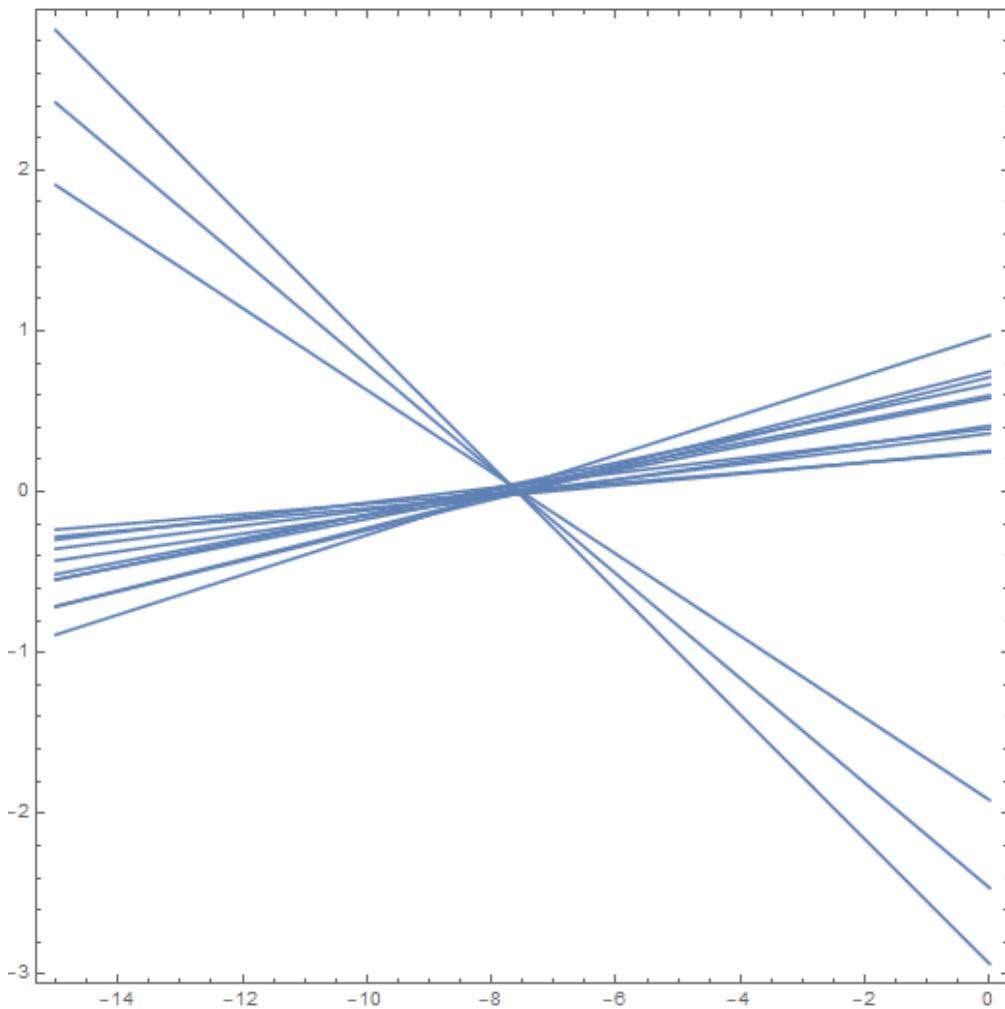


Abbildung 6.9: Resultierende Epipolarlinien des Bildes der Canon 60D

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* [4] 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 [4]. Das neue \hat{F} führt, wenn wir uns wieder die Epipolarlinien ausgeben lassen zu folgendem Ergebnis.

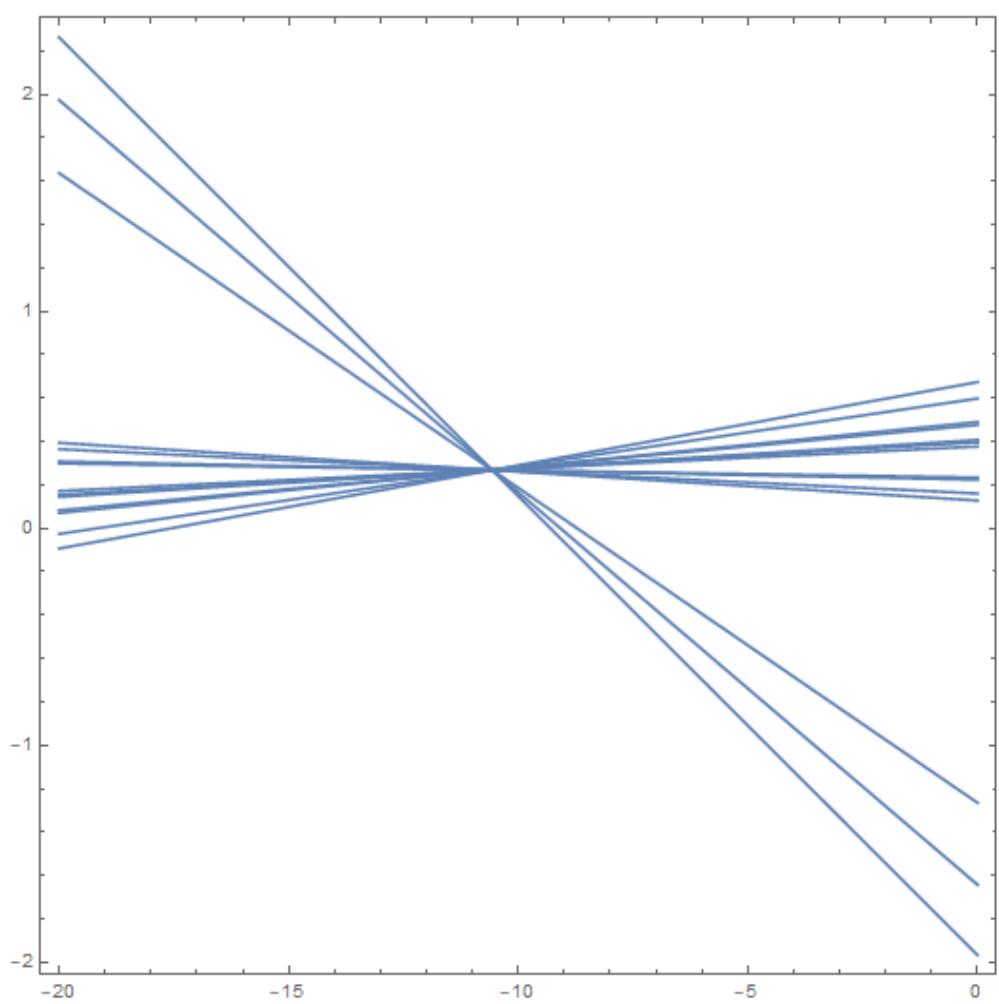


Abbildung 6.10: Resultierende Epipolarlinien des Bildes der Canon 6D nach singularity-constraint

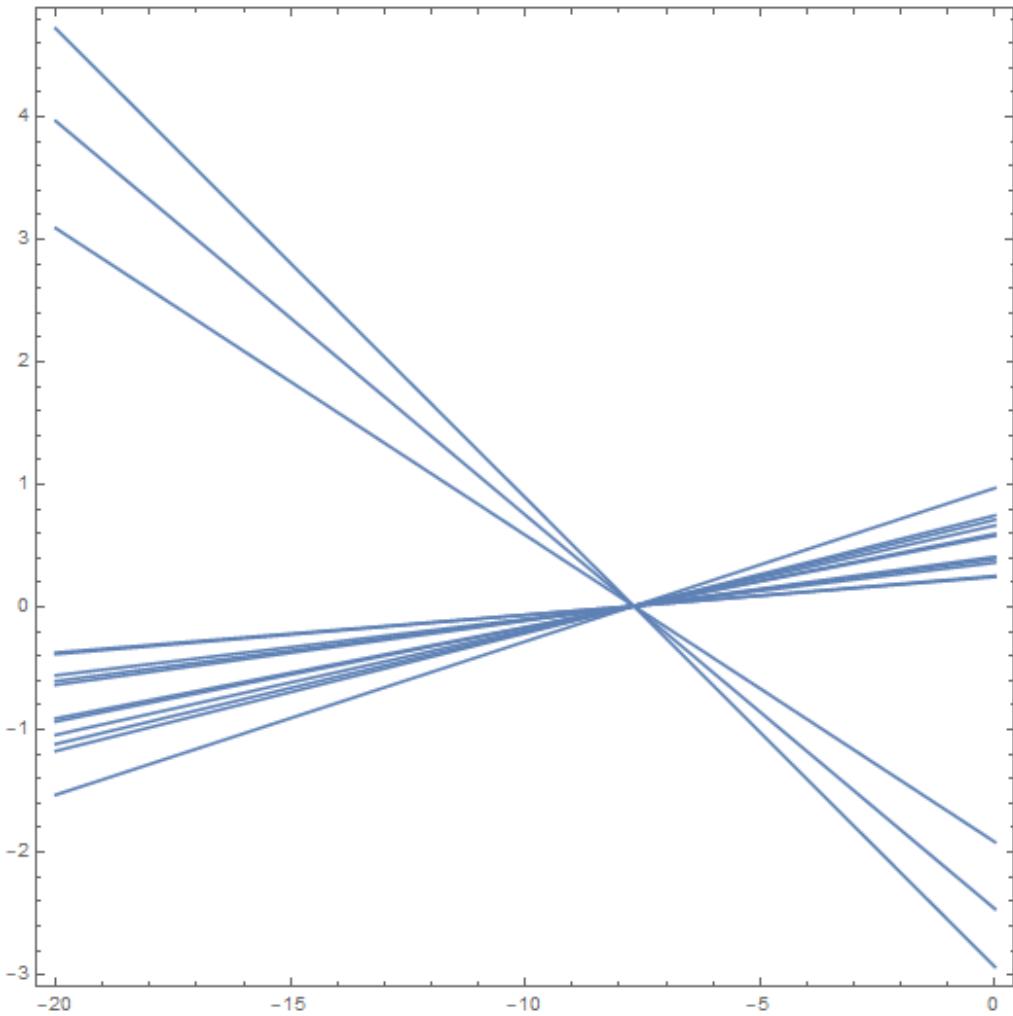


Abbildung 6.11: Resultierende Epipolarlinien des Bildes der Canon 60D nach singularity-constraint
Letztendlich wird dann \hat{F} noch mit Hilfe der beiden Matrizen T und T' denormalisiert

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

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 [4]. 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[4].

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

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	pI, 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, pI	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, pI	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 pI durch. Innerhalb der For-Schleife wird dann überprüft, welche Punkte aus pI 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 VecJ 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.

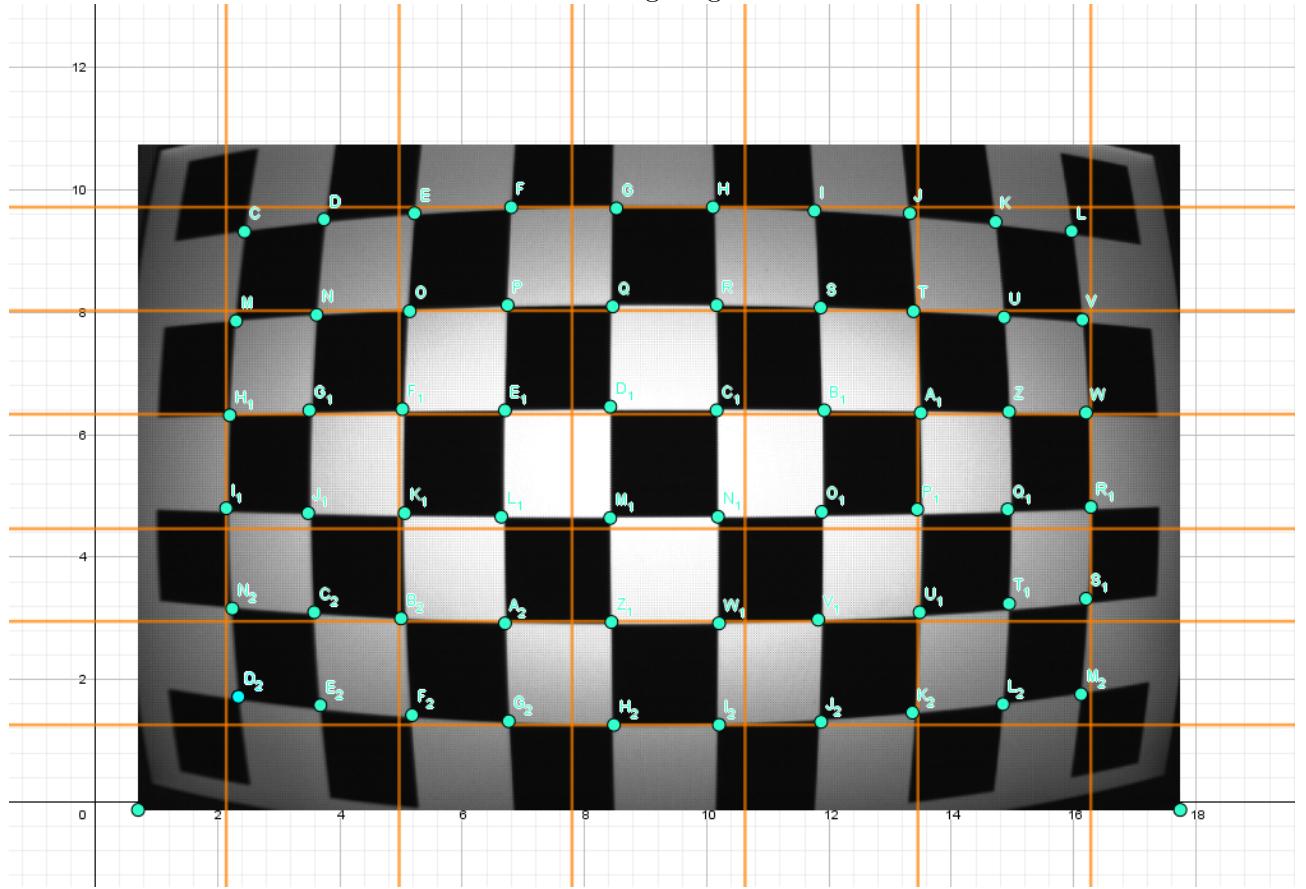


Abbildung 8.6: Bild eines Tonnenförmig verzeichneten Schachbretts

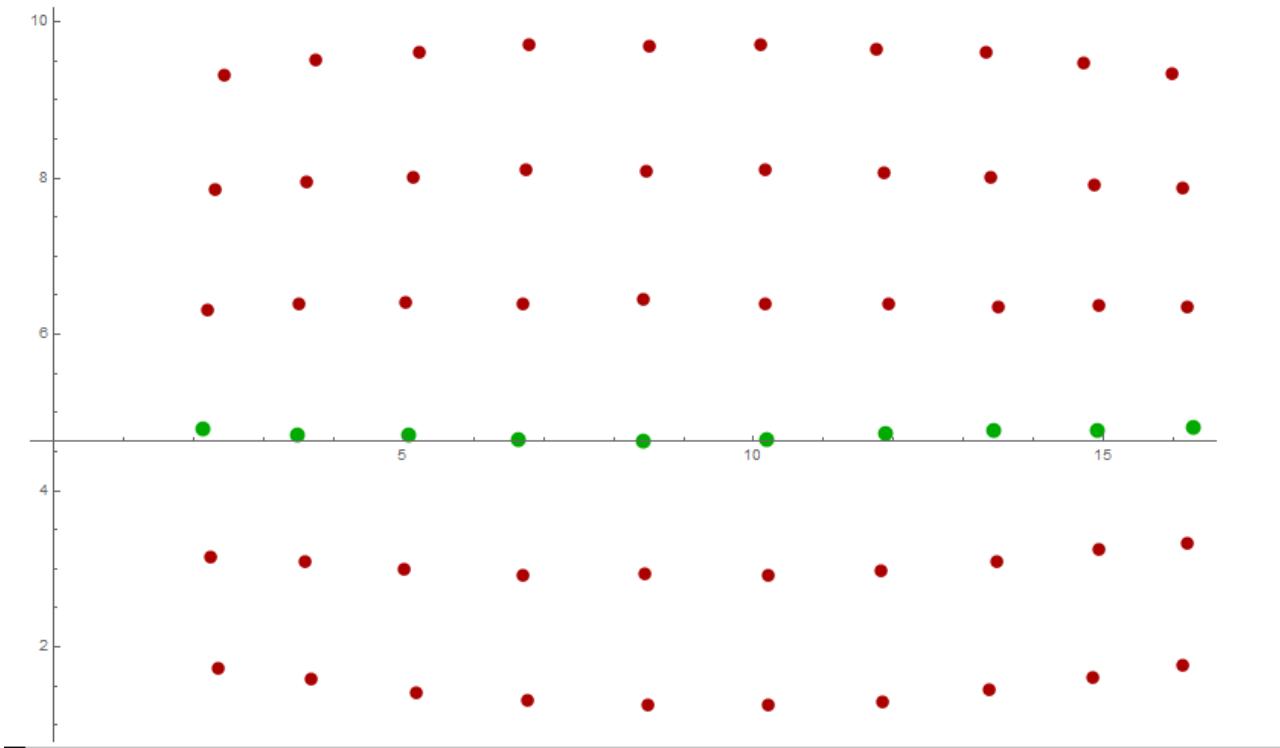


Abbildung 8.7: Algorithmisch detektierte Linie der dritten i-Reihe

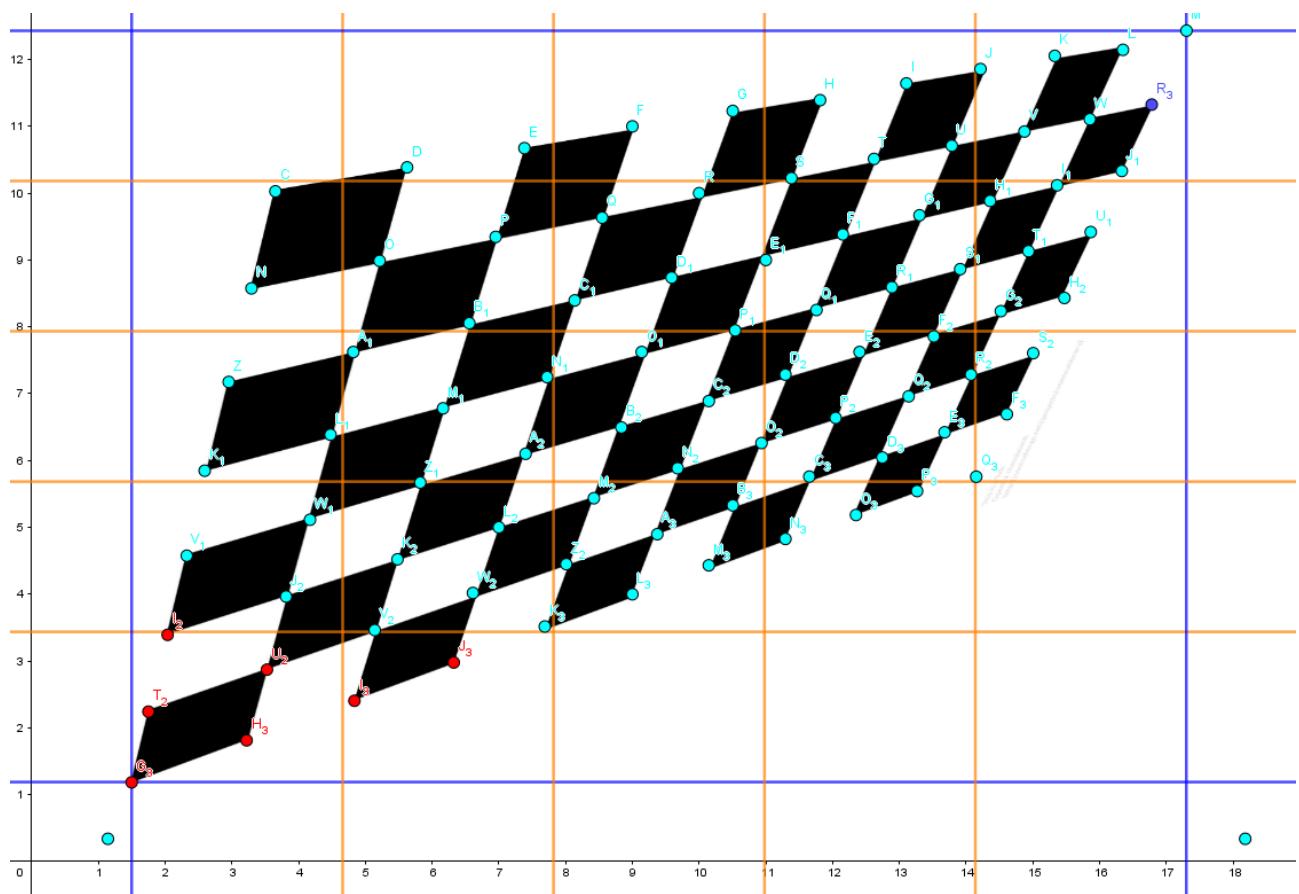


Abbildung 8.8: Bild eines perspektivisch verzerrtem Schachbretts

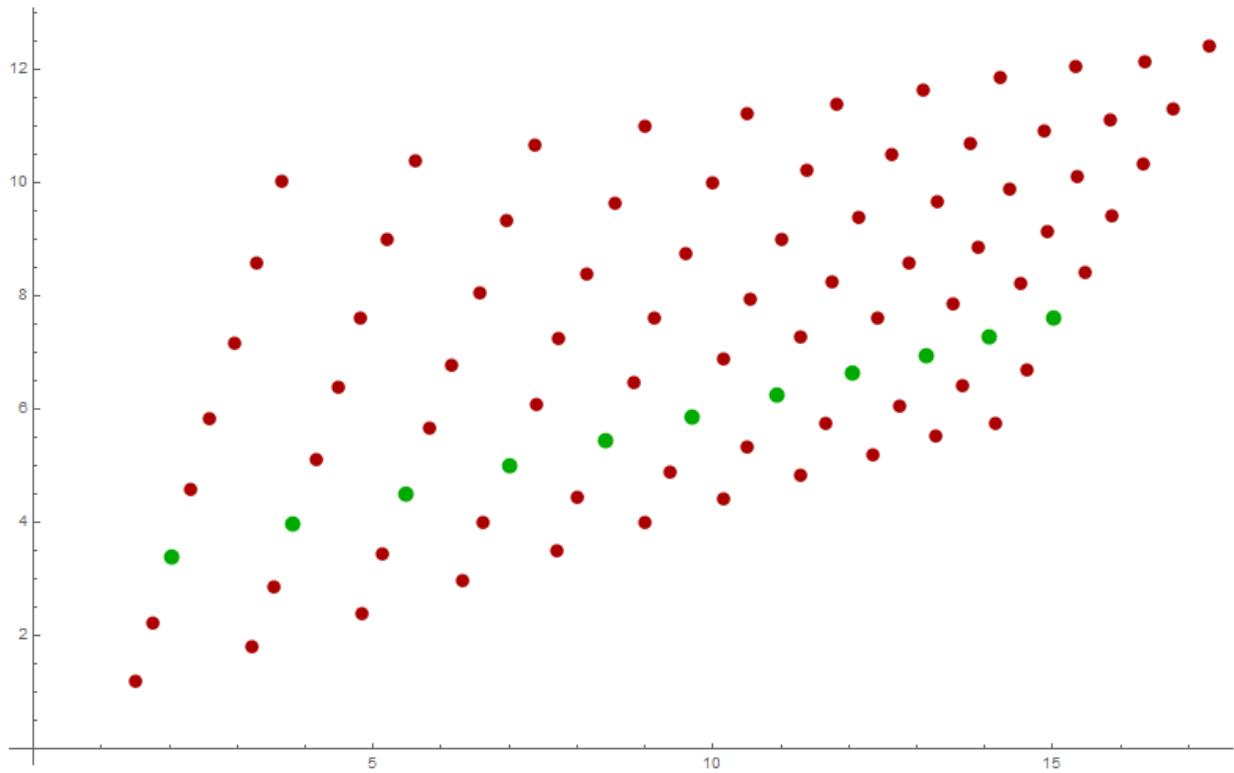


Abbildung 8.9: Algorithmisch detektierte Linie der dritten i-Reihe

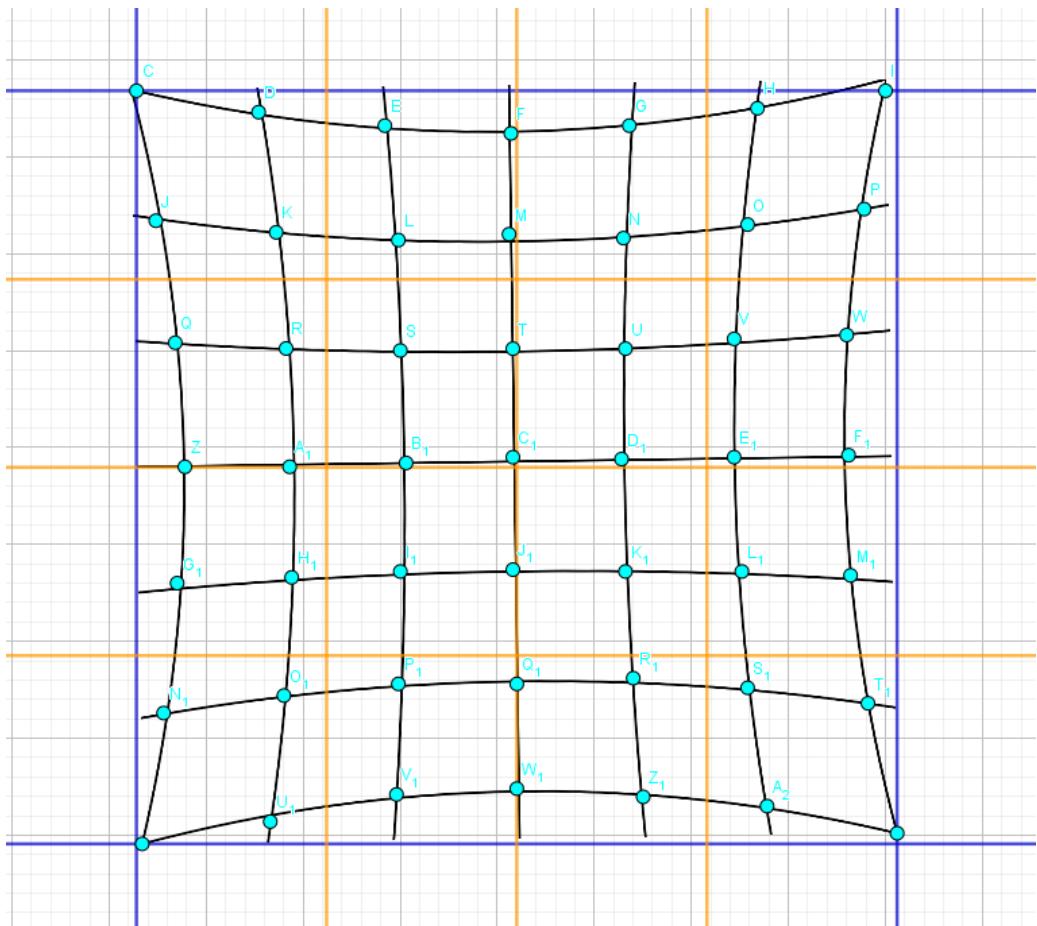


Abbildung 8.10: Bild eines Kissenförmig verzeichnetem Schachbretts

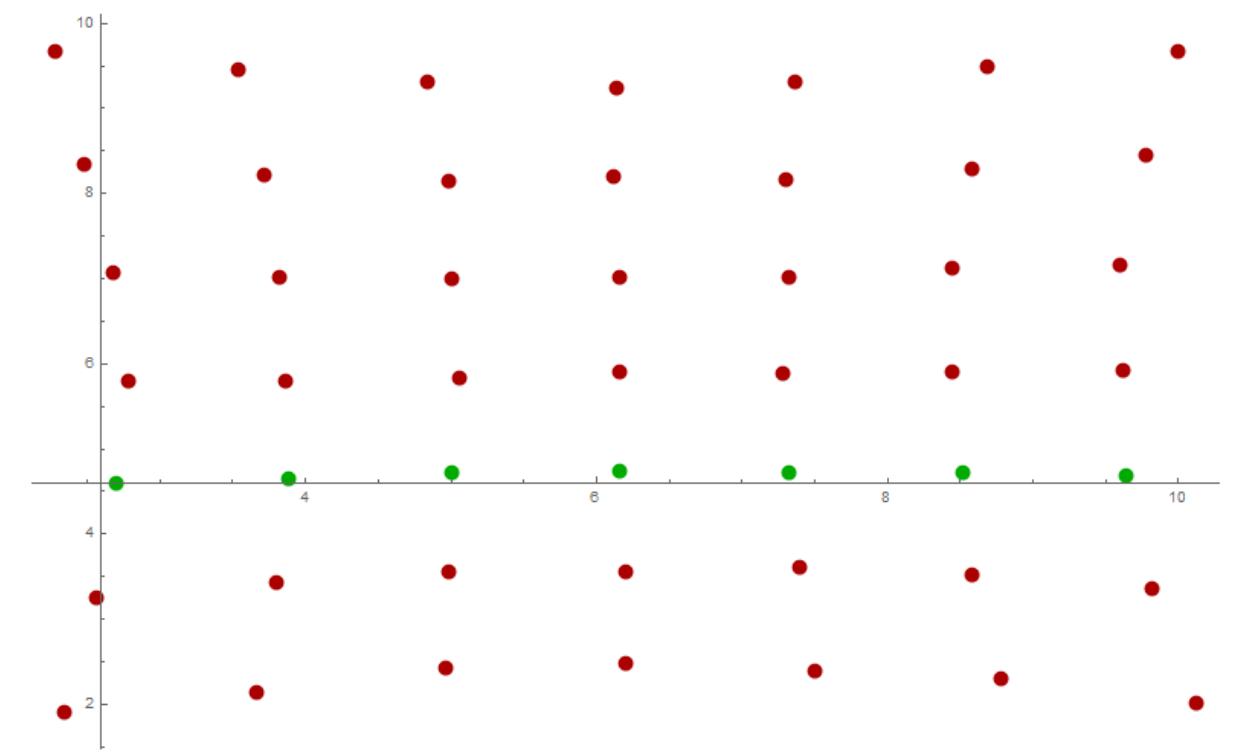


Abbildung 8.11: Algorithmisch detektierte Linie der dritten i-Reihe

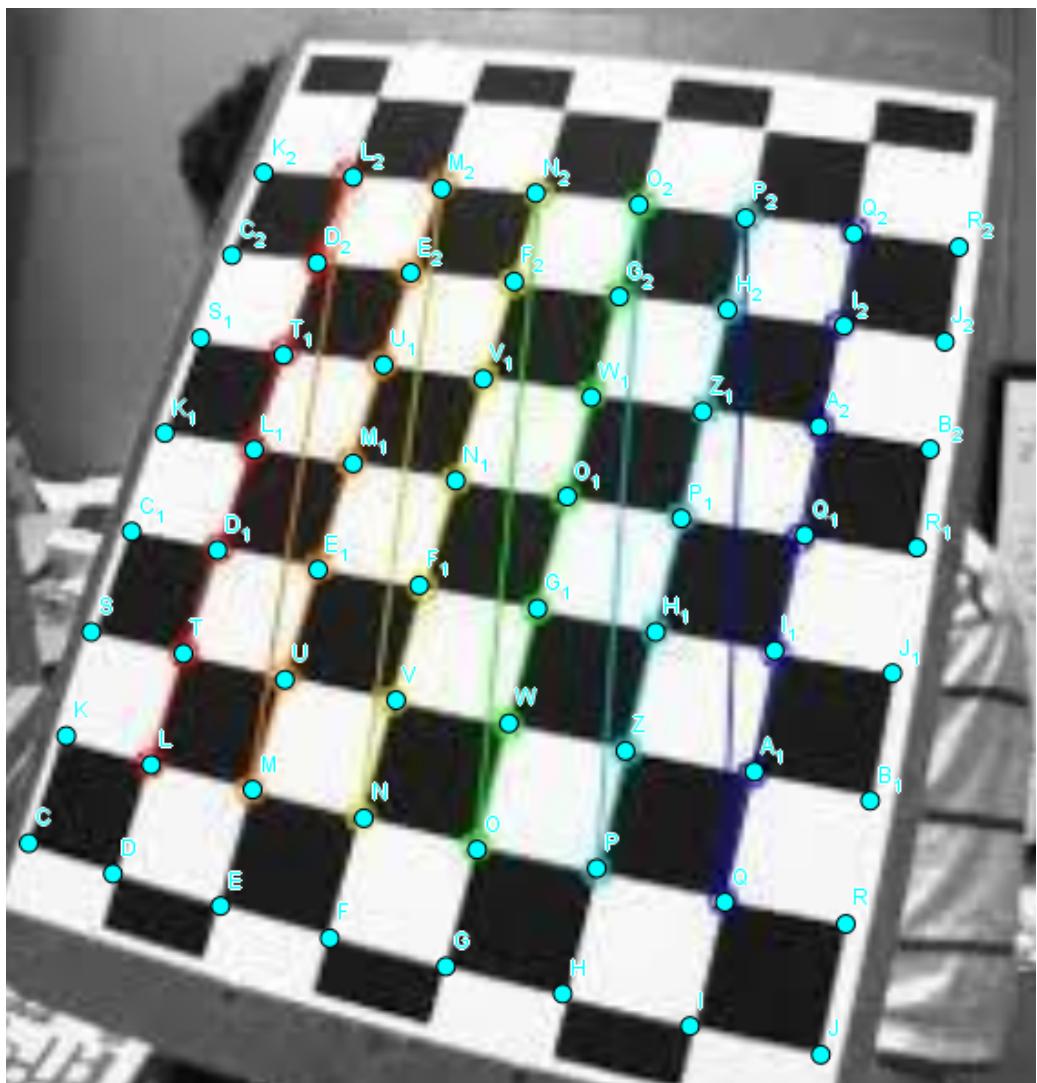


Abbildung 8.12: Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts

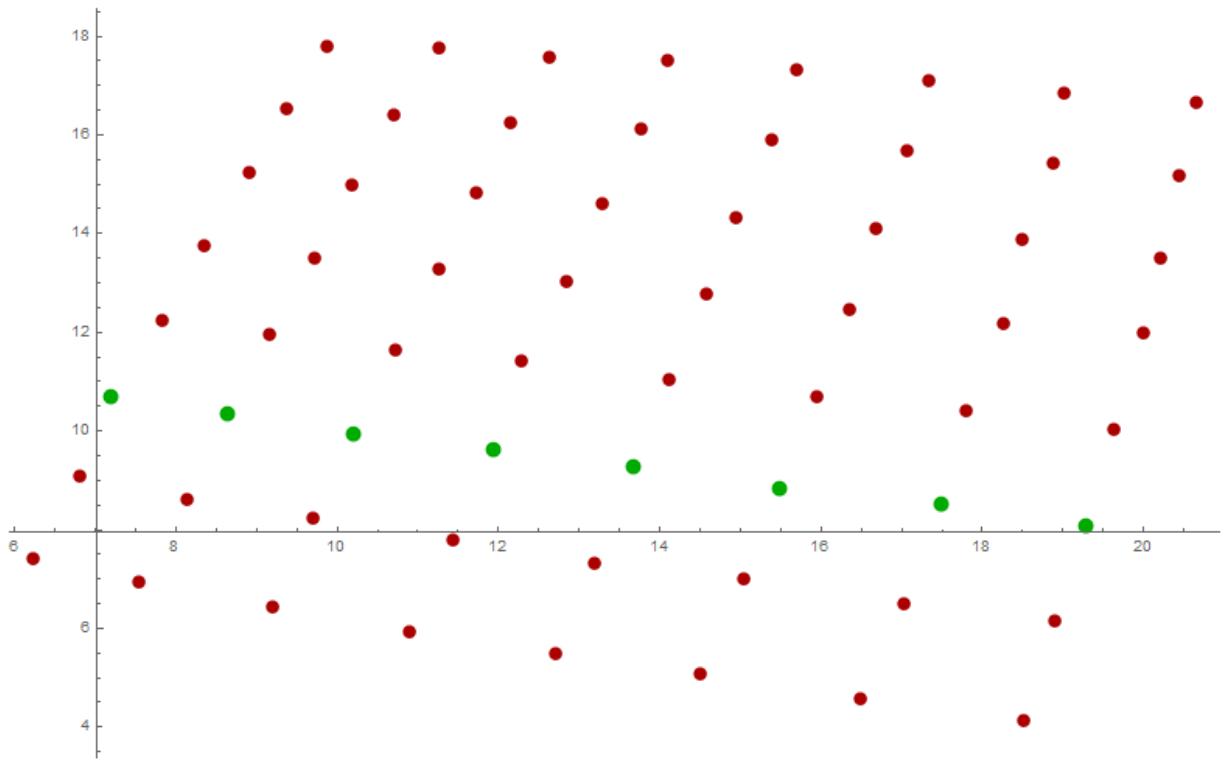


Abbildung 8.13: Algorithmisch detektierte Linie der dritten i-Reihe

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.

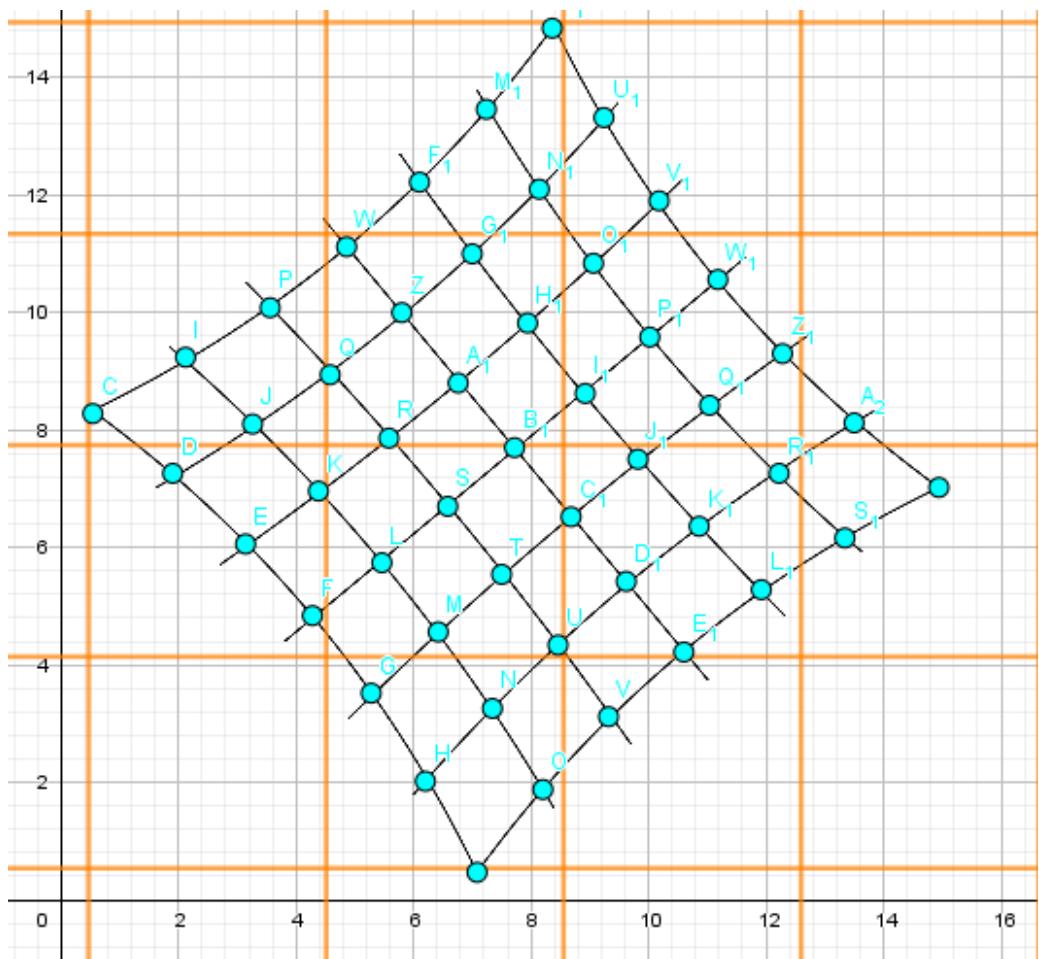


Abbildung 8.14: Kissenverzeichnung start rotiert

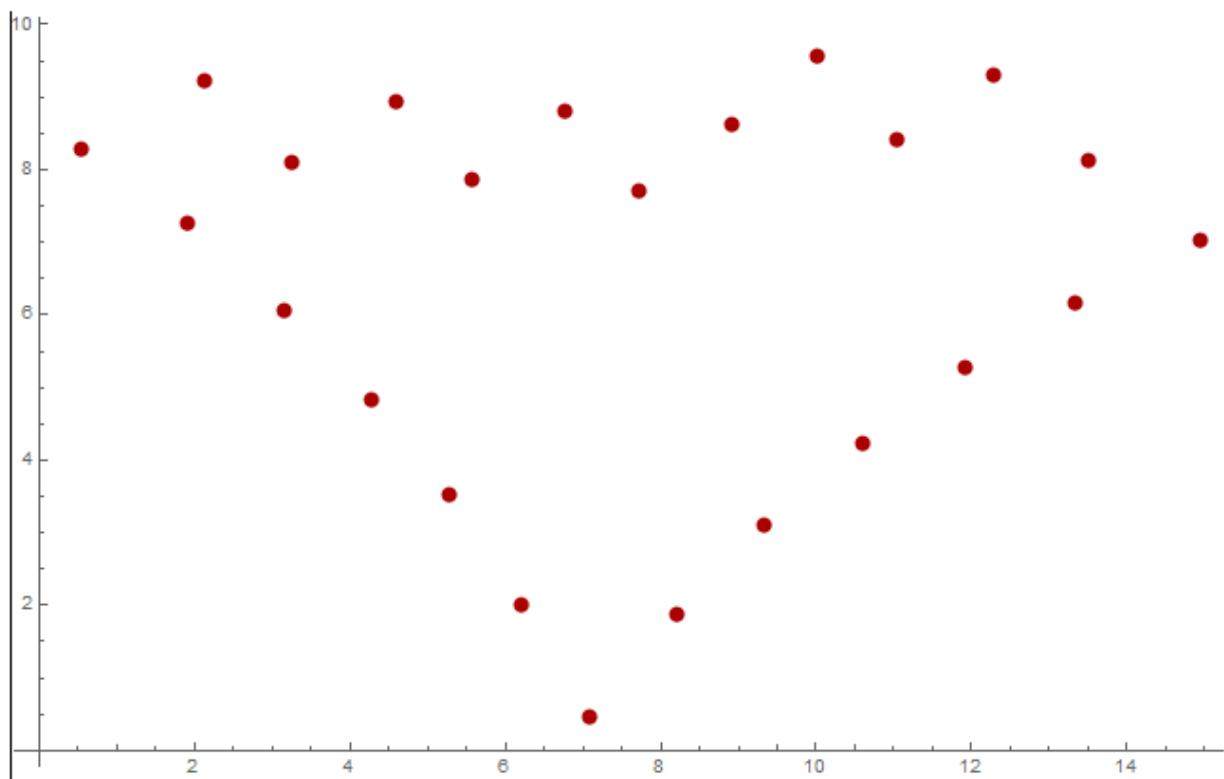


Abbildung 8.15: Kissenverzeichnung start rotiert bisher gefundene Punkte

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 (O, δ) mit $\delta = (d_1, d_2, d_3)$ und Kamerakoordinatensystem (C, β) mit $\beta = (b_1, b_2, b_3)$	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 Projektionszentrum liegt hinter der Bildebene und somit auch hinter dem Sensor. n ist die Normale der Bildebene	12
3.1	Veranschaulichung von Homographie mit nur einer rotierten Kamera	18
3.2	Veranschaulichung von Homographie mit nur einer rotierten Kamera	19
3.3	Weltkoordinatensystem (O, δ) mit $\delta = [\vec{d}_1, \vec{d}_1, \vec{d}_1, O]$ und Kamerakoordinatensysteme $\beta = [\vec{b}_1, \vec{b}_2, \vec{b}_3, C]$ und (C', β') mit $\beta' = [\vec{b}'_1, \vec{b}'_2, \vec{b}'_3, C']$	21
3.4	in blau ist die Abbildung des Quaders von Kamera eins und in rot die Abbildung des selben Quaders in Kamera zwei	25
3.5	Objekt im Raum	28
3.6	Drehung um das Projektionszentrum	28
3.7	Drehung um einen Drehpunkt. In diesem Beispiel wurde der rote Punkt als Drehpunkt verwendet	29
3.8	Strahlengang durch das Projektionszentrum. Auf der Grafik ist erkennbar, dass der grüne Punkt auch nach der Drehung der Kamera um das Projektionszentrum vom roten Punkt verdeckt bleibt	29
3.9	Strahlengang durch das Projektionszentrum. Auf der Grafik ist erkennbar, dass der grüne Punkt nach der Drehung der Kamera um einen Drehpunkt, welcher in diesem Fall der rote Punkt darstellt, sichtbar wird.	30
3.10	Veranschaulichung der Homographie bei zwei verschieden translatierten und rotierten Kameras. (\vec{CO}) und $(\vec{C'O})$ sind jeweils die Vektoren welche von den Abbildungspunkten m beziehungsweise m' ausgehen und sich im Objektpunkt M treffen.	31
3.11	Weltkoordinatensystem (O, δ) mit $\delta = [\vec{d}_1, \vec{d}_1, \vec{d}_1, O]$ und Kamerakoordinatensysteme $\beta = [\vec{b}_1, \vec{b}_2, \vec{b}_3, C]$ und (C', β') mit $\beta' = [\vec{b}'_1, \vec{b}'_2, \vec{b}'_3, C']$	32
3.12	Grafik zu den geometrischen Eigenschaften der Epipolargeometrie zwischen zweik Bildern. C und C' sind die Projektionszentren zweier Kameras. Beide Kameras besitzen jeweils eine Bildebene. Die Basislinien verbindet die Projektionszentren der Kameras. Der Punkt an welchem die Basislinie die Bildebenen schneidet, wird als Epipol bezeichnet. Durch den Epipol verlaufen alle Epipolarlinien des Bildes. M ist der Objektpunkt im 3D-Raum und m_1 und m_2 sind die jeweiligen Abbildungen dieses Punktes auf den Bildebenen. Die Verbindungsvektoren zwischen C, C' und M bilden die sogenannte Epipolarebene[3, 4, 1].	35
3.13	Die Objektpunkte M_1, M_2 und M_3 werden in I' als m'_1, m'_2 und m'_3 abgebildet, während sie in I immer den selben Bildpunkt m_1 ergeben.	36
4.1	vereinfachte Top-Down-Ansicht des Szenenaufbaus des Minimalbeispiel	40
4.2	In Grün ist die Abbildung auf der Bildebenen I von C und in rot ist die Abbildung auf der Bildebenen I' von C'	40
4.3	In Blau und Rot sind jeweils das Welt- und Kamerakoordinatensystem von Kamera eins zu sehen. In grün ist das gedrehte Koordiantensystem von Kamera 2 zu sehen.	40
4.4	Grün zeigt den Quader welcher auf I von C abgebildet wird. Das größere Quadrat sind die vorderen Punkte A, B, C, D , das kleinere Quadrat sind die hinteren Punkte A', B', C', D' . Der Punkt E ist weiter weg von den Abbildungen und deshalb auf dieser Abbildung momentan nicht zu sehen. Rot zeigt denselben Quader auf I' von C' abgebildet.	43

4.5	Die blauen Geraden zeigen die jeweiligen Epipolargeraden. Die vom roten Quader schneiden sich bei -1 im Epipol e' . Die Epipolargeraden vom grünen Quader schneiden sich im Epipol e' im Unendlichen, weshalb die Epipolarlinien Parallel zueinander verlaufen.	46
4.6	Anordnung der Kameras bei den vier verschiedenen Lösungen für P	50
4.7	Optimale Triangulierung: Beide Geraden Treffen sich in einem Punkt im 3D-Raum	51
4.8	Veranschaulichung der Skaleninvarianz und dessen Auswirkung auf die geometrische Form der Objekte	53
4.9	auf Originalgröße skalierte rekonstruierte Szene	54
4.10	Beispiel eines rektifizierten Bildes. Quelle: [17]	55
4.11	Beispiel einer einfachen Tiefenkarte eines Stereobildpaars nach der Rektifizierung. Quelle: [18]	55
4.12	Aufnahmen zweier Kameras mit den selben Auflösungen, Kamera eins(Grün) und Kamera(rot) zwei gelten jeweils $\zeta = 1$	62
4.13	Epipole für Kamera eins und Kamera zwei vor der Rektifizierung	62
4.14	Abbildung beider Bilder nach anwenden der Matrizen H_p und H'_p . Die Epipole beider Bilder sind nun im unendlichen. Das die entstehenden parallelen Epipolarlinien auch hier schon horizontal ausgerichtet sind ist Zufall. Die Epipolarlinien sind immer parallel nach dieser Transformation aber die Richtung ist nicht immer automatisch bereits $i = [1,0,0]$	63
4.15	Abbildung beider Bilder nach anwenden der Matrizen $H_r \cdot H_p$ und $H'_r \cdot H'_p$. Die Epipolarlinien sind nun horizontal zueinander ausgerichtet	65
4.16	Abbildung beider Bilder nach anwenden der Matrizen $H_s \cdot H_r \cdot H_p$ und $H'_s \cdot H'_r \cdot H'_p$. Die horizontale Verzerrung wurde reduziert.	65
4.17	In dieser Abbildung wurden die Epipolarlinien noch in den Grafikplot mit eingebaut	66
5.1	Rechteckiger Bildsensor mit darauf sich befindenden quadratischen Pixeln	68
5.2	Bild a) zeigt die Interpolation von Pixeln, wenn bei gleichbleibenden Seitenverhältnissen weniger Pixel für das Bild verwendet werden sollen. Die interpolierten Pixel leiten dann alle das selbe Signal weiter. Bild b) zeigt in gelb markiert, den verwendeten Bereich des Sensors, wenn sich die Seitenverhältnisse ändern und nicht mehr der volle Sensor genutzt wird.	69
5.3	C und C' haben die selbe Auflösung eingestellt	70
5.4	C und C' haben unterschiedliche Auflösungen eingestellt	70
5.5	C und C' haben die selbe Auflösung eingestellt	71
5.6	C und C' haben unterschiedliche Auflösungen eingestellt	71
5.7	C und C' haben die selbe Auflösung eingestellt	71
5.8	C und C' haben unterschiedliche Auflösungen eingestellt	71
5.9	Die Fundamentalmatrizen sind bei jeder Auflösung, die gewählt wurden immer Vielfache voneinander	72
5.10	Die essentiellen Matrizen sind bei jeder Auflösung, die gewählt wurden immer Vielfache voneinander	73
5.11	Die rekonstruierten Szenenpunkte und Kamerapositionen bleibt auch bei unterschiedlichen Auflösungen die selben	74
6.1	Rekonstruktion des Gier-Winkels der Kameras zueinander	78
6.2	Kamera 6D	79
6.3	Kamera 60D	79
6.4	Stereosetup	80
6.5	Szenenbild der primären Kamera 6D	80
6.6	Szenenbild der sekundären 60D	81
6.7	Beispiel einer resultierenden Koeffizientenmatrix	82
6.8	Resultierende Epipolarlinien des Bildes der Canon 6D	83
6.9	Resultierende Epipolarlinien des Bildes der Canon 60D	84

6.10	Resultierende Epipolarlinien des Bildes der Canon 6D nach singularity-constraint	85
6.11	Resultierende Epipolarlinien des Bildes der Canon 60D nach singularity-constraint	86
6.12	Ergebnis der Prüfung der Punktekorrespondenzen mit der essentiellen Matrix	87
6.13	Linker Nullspace von E und S	88
6.14	Übergebene Essentielle Matrix	89
6.15	Test $E = [t]_x R$	89
8.1	Klassendiagramm	92
8.2	Klassendiagramm	92
8.3	Klassendiagramm	93
8.4	Klassendiagramm	93
8.5	Klassendiagramm	93
8.6	Bild eines Tonnenförmig verzeichneten Schachbretts	94
8.7	Algorithmisch detektierte Linie der dritten i-Reihe	95
8.8	Bild eines perspektivisch verzerrtem Schachbretts	95
8.9	Algorithmisch detektierte Linie der dritten i-Reihe	96
8.10	Bild eines Kissenförmig verzeichnetem Schachbretts	96
8.11	Algorithmisch detektierte Linie der dritten i-Reihe	97
8.12	Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts .	98
8.13	Algorithmisch detektierte Linie der dritten i-Reihe	99
8.14	Kissenverzeichnung start rotiert	100
8.15	Kissenverzeichnung start rotiert bisher gefundene Punkte	100

Tabellenverzeichnis

5.1 Auflösungen Canon EOS 6D	68
--	----

Literaturverzeichnis

- [1] Zhengyou Zhang Gang Xu. *Epipolar Geometry in Stereo, Motion and Object Recognition: A Unified Approach*. Springer-Science and Business Media, 1996.
- [2] Jianzhong Lu. *Ein dreidimensionales Bildverarbeitungssystem für die Automatisierung visueller Prüfvorgänge*.
- [3] Tomas Pajdla. *Elements of Geometry for Computer Vision*. "<http://people.ciirc.cvut.cz/pajdla/>", 2013, überarbeitet am 27.2.2017.
- [4] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge, 2004, Second Edition.
- [5] Zhengyou Zhang. *Epipolar Geometry*, pages 247–258. Springer US, Boston, MA, 2014.
- [6] Christian Heipke. *Photogrammetrie und Fernerkundung*. 2017 Springer, 1. Auflage.
- [7] Rolf Martin Ekbert Hering. *Photonik, Grundlagen, Technologien und Anwendung*. Springer-Verlag Berlin Heidelberg, 2006.
- [8] Dipl.-Ing. Martin Roser. *Modellbasierte und positionsgenaue Erkennung von Regentropfen in Bildfolgen zur Verbesserung von viedeoisierten Fahrerassistenzfunktionen*. 1986, 1994 Springer Basel AG, KIT Scientific Publishing.
- [9] Jeanne Peiffer and Amy Dahan-Dalmat. *Wege und Irrwege - Eine Geschichte der Mathematik*. 1986, 1994 Springer Basel AG, Editions du Seuil, Springer Basel AG, aus dem französischen von Klaus Volkert.
- [10] Norbert Köckler Hans Rudolf Schwarz. *Numerische Mathematik*. 2011, Springer Verlag, 8. Auflage.
- [11] Daniel Scholz. *Numerik interaktiv, Grundlagen verstehen, Modelle erforschen und Verfahren anwenden mit taramath*. 2016, Springer Verlag.
- [12] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. Received July 16, 1996; Accepted February 13, 1997.
- [13] Richard I. Hartley. In defence of the 8-point algorithm. GE-Corporate Research and Development, Schenectady, NY, 12309.
- [14] Ferid Bajrmovic. *Self-Calibration of Multi- Camera Systems*. Logos Verlag Berlin GmbH, 2010.
- [15] Cui Guodong Yu Ming Zhang Mandun, Qi Lichao. A triangulation method in 3d reconstruction from image sequences. College of Computer Science and Software, Hebei University of Technology Tianjin, China, 2009, Second International Conference on Intelligent Networks and Intelligent Systems.
- [16] MathWorks. Mathworks documentation, rectify stereo images. "<https://de.mathworks.com/help/vision/ref/rectifystereoimages.html>".
- [17] Charles Loop and Zhengyou Zhang. Computing rectifying homographies for stereo vision. Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, Vol.1, pages 125–131, June 23-25, 1999. Fort Collins, Colorado, USA, 1999 Errors corrected on June 6, 2001.

- [18] Carlos VILLAGRÁ ARNEDOr Antonio Javier GALLEGÓ SÁNCHEZ, Rafael MOLINA CARMONA. *Scene reconstruction and geometrical rectification from stereo images*. Januar 2005, uploaded by Antonio Javier Gallego Sánchez on 21 May 2014, ResearchGate.
- [19] Luca Irsara Andrea Fusiello. Euclidean epipolar rectification of uncalibrated images. Eurac researc, IT.
- [20] Richard I. Hartley. Euclidean reconstruction from uncalibrated views. G.E. CRD, Schenectady, NY, 12301.
- [21] Andrea Fusiello. Euclidean epipolar rectification. "http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO2/rectif_cvol.html".
- [22] Dongqing Li, editor. *Encyclopedia of Microfluidics and Nanofluidics*, pages 999–999. Springer US, Boston, MA, 2008.
- [23] S. Margulies. Fitting experimental data using the method of least squares. Department of Physics, University of Illinois at Chicago Circle, Chicago, Illinois 60680, 1967.
- [24] William T. Vetterling Brian P. Flannery William H. Press, Saul A. Teukolsky. *Numerical Recipes in Fortran 77 The Art Of Scientific Computing*. Copyright Numerical Recipes Software 1986, 1992, 1997 All Rights Reserved., Reprinted with corrections 1997, Volume 1 of Fortran Numerical Recipes.
- [25] MathWorks. Mathworks documentation,estimate camera parameters. "<https://de.mathworks.com/help/vision/ref/estimatecameraparameters.html>".
- [26] MathWorks. Mathworks documentation, disparity. "<https://de.mathworks.com/help/vision/ref/disparity.html>".
- [27] MathWorks. Mathworks documentation, stereo camera calibration app. "<https://de.mathworks.com/help/vision/stereo-camera-calibration.html>".
- [28] Canon. Eos 6d, eos 6d (wg), eos 6d (n), instruction manual, pages 188. "http://gdlp01.c-wss.com/gds/6/0300009626/01/EOS_6D_Instruction_Manual_DE.pdf".