

---

# **Szenenkonstruktion aus stereoskopischen Bildquellen gleicher und verschiedener Auflösungen**

---

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

Medieninformatik 4.Semester Anja Kretschmer 222222

Betreut von: Prof. Dr. Thomas Schneider

Disclaim here



Fakultät Digitale Medien der Hochschule Furtwangen  
Wintersemester 17/18 - Sommersemester 18

# Inhaltsverzeichnis

<b>1 Einleitung</b>	<b>4</b>
<b>2 Model der Bildaufnahme mit einer Kamera</b>	<b>5</b>
2.1 Lochkameramodell zur Abbildung eines Punktes auf die Bildebene . . . . .	5
2.2 Koordinatentransformation . . . . .	6
2.3 Aufname mit einer willkürlichen Kameraorientierung . . . . .	9
<b>3 Geometrische Beziehungen zwischen Punktekorrespondenzen</b>	<b>12</b>
3.1 Korrespondenzen planarer Punktmengen mit Homographien . . . . .	12
3.2 Korrespondenzanalyse für beliebige Punkte im Raum (Epipolare Geometrie) . . . . .	14
3.3 Bestimmung von Homographie und Fundamentalmatrix aus Punktekorrespondenzen . . . . .	17
<b>4 Synthetische Rekonstruktion</b>	<b>21</b>
4.1 Simulierte Bildaufnahme einer virtuellen Szene . . . . .	21
4.2 Bildanalyse . . . . .	24
4.2.1 Bestimmung der extrinsischen Kameraparameter . . . . .	24
4.2.2 Szenenrekonstruktion durch Triangulation . . . . .	26
<b>5 Auswirkungen von unterschiedlichen Kameraauflösungen</b>	<b>30</b>
5.1 Geometrie eines Sensors . . . . .	30
5.2 Auswirkungen auf die Szenenrekonstruktion . . . . .	31
<b>6 Reelle Rekonstruktion</b>	<b>36</b>
6.1 Stereoaufbau . . . . .	36
6.2 Korrespondenzanalyse . . . . .	37
6.3 Normierter acht-Punkt-Algorithmus . . . . .	38
6.3.1 Singularität der Fundamentalmatrix . . . . .	39
6.3.2 Singulärwerte der essentiellen Matrix . . . . .	41
6.4 Szenenrekonstruktion mit Sampson-Approximation . . . . .	41
6.5 Ergebnisse einer Stereoanalyse mit Kameras unterschiedlicher Auflösung . . . . .	46
<b>7 Vergleich entwickelter Rekonstruktions Algorithmus mit bereits vorhandenen (Matlab)</b>	<b>51</b>
7.1 Projektive Transformation . . . . .	55
7.2 Ähnlichkeitstransformation . . . . .	60
7.3 Scherungstransformation . . . . .	62
<b>8 Punktesortierung in Schachbrettmustern</b>	<b>66</b>
8.1 Vorläufiges Klassendiagramm . . . . .	67
8.2 Beispiele . . . . .	71
<b>9 Fazit - Conclusion</b>	<b>74</b>
<b>10 Alternativen</b>	<b>75</b>
<b>11 Abkürzungsverzeichnis - List of Abbreviations</b>	<b>76</b>

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

Die Computer Vision ist ein Fachbereich der Computer Science mit dem Fokus auf der Entwicklung von künstlicher Intelligenz, die ein visuelles Verständnis ihrer Umgebung besitzen. Folglich wird in der Computer Vision der Weg von visuellen Eindrücken oder Bildern aus der Realität in den Rechner beschrieben [2]. Der Mensch ist mit der Fähigkeit ausgestattet, gesehene Bilder zu verarbeiten und kann die ihn umgebene Welt verstehen. Maschinen, die eine ähnliche Fähigkeit besitzen, wären somit ebenfalls in der Lage Entscheidungen auf Grund von visuellen Eindrücken zu fällen. Das entwickeln solcher Maschinen und den damit verbundenen Grundprinzipien und Programme sind die Forschungsmittelpunkte von aktuellen Anwendungsbereichen wie dem Autonomen Fahren, Motion-Caturing, Bewegungserkennungen oder Service Robotern.

In dieser Masterarbeit wurde ein Algorithmus zur Rekonstruktion einer Szene aus stereoskopischen Bildquellen entwickelt. Das typische Verfahren einer Stereorekonstruktion basiert auf den Grundbausteinen, Bildaufnahme und Bildanalyse[2]. In der Bildaufnahme wird eine Szene oder ein Objekt mit Hilfe von Kameras, Sensoren oder Lasern aufgenommen und als digitale zweidimensionale Bilder an den Computer weitergegeben. In der Bildanalyse, werden die aufgenommenen Bilder ausgewertet um so die dreidimensionale Szene rekonstruieren zu können. Für die Analyse ist es essentiell die Kameraparameter, wie Position und Auflösung, zu kennen. Sind diese jedoch nicht bekannt, können die Bildquellen genutzt werden um die Kameraparameter abzuschätzen. Eine solche Abschätzung wird als wird als Kamerakalibrierung[3, 4, 5, 1] bezeichnet. Die Position und Rotation einer Kamera im Raum werden als die extrinsischen Kameraparameter bezeichnet, Parameter wie die Auflösung oder Brennweiten, werden als die intrinsischen Kameraparameter bezeichnet[3, 4]. Im Zuge dieser Arbeit ist ein Algorithmus entstanden, welcher unter anderem im Stande ist die Kameras gleicher und unterschiedlicher Auflösung zu kalibrieren eine 3D-Szenenrekonstruktion durchzuführen. Der vollständige Algorithmus wurde mithilfe eines virtuellen Beispiels verifiziert und auf eine reelle Szenenaufnahme angewandt. Mit dem Entwickeln von Algorithmen für Computer Vision Applikationen, sieht man sich mit immer wieder mit komplizierten Aufgaben und Herausforderungen konfrontiert. Bei der Aufnahme von Bildern, kann es immer wieder zu unvorhersehbaren Bildfehlern wie beispielsweise Rauschen oder Verzerrungen durch die Kameralinse kommen, was auch nicht oft zum Verlust von Referenzdaten führt. Im Kapitel Reelle Rekonstruktion wird aufgeführt, wie mit solchen Fehlern umgegangen werden kann.

In der virtuellen Rekonstruktion wird zuerst eine 3D Szene in zwei voneinander unterschiedlich positionierten, simulierten Kameras projiziert um virtuelle Bilddaten zu generieren. Anhand dieser 2D-Bilddaten wird die Kamerakalibrierung getestet. In der virtuellen Rekonstruktion, werden die Werte für Auflösung und Brennweite, welche als intrinsische Parameter bezeichnet werden, selbst gesetzt. Im Test des Algorithmus mit reellen Bilddaten, wird für dessen Schätzung auf ein bereits existierendes Programm zurückgegriffen. Die intrinsischen Parameter werden mit dem hier entwickelten Algorithmus für die Schätzung der Positionen und Orientierungen der Kameras, die als extrinsische Parameter bezeichnet werden, kombiniert um die Kameras anhand der virtuellen Daten zu kalibrieren. Die durch die Schätzung erhaltenen Kameraparameter können im virtuellen Beispiel so einfach mit den zuvor definierten Parametern verglichen werden, um den Algorithmus zu verifizieren. Diese Kameraparameter werden dann entwickelten Rekonstruktionsalgorithmus dazu verwendet, in die ursprüngliche 3D-Szene wieder herzustellen und die Funktionsweise der Rekonstruktion zu analysieren.

## 2 Model der Bildaufnahme mit einer Kamera

Um einen Szenenrekonstruktionalgorithmus zu verstehen, werden in diesem Abschnitt grundlegende Bedingungen eingeführt um die Bildaufnahme mathematisch zu beschreiben. Ein Abbildendes System besteht aus einem Objekt  $M$ , einer Kamera  $C$  und einer Bildebene  $I$  wie in Abbildung 2.1 dargestellt.

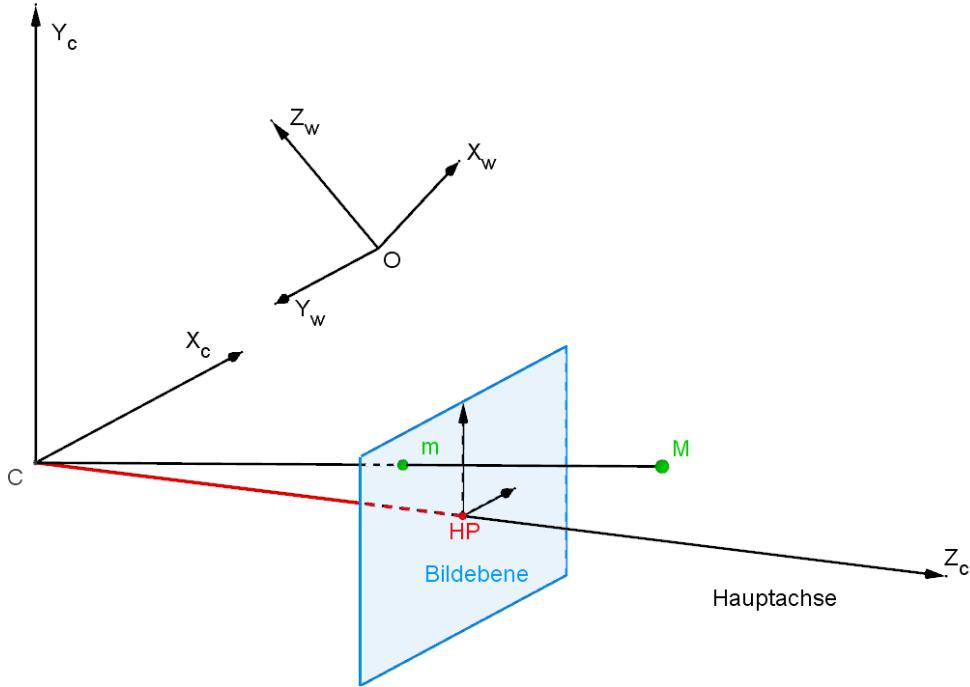


Abbildung 2.1: Schematik eines abbildenden Systems. Ein Punkt  $M$  im Weltkoordinatensystem  $O$  wird durch eine Kamera  $C$  aufgenommen. Diese Aufnahme wird durch eine Projektion, die als Verbindungslinie von  $M$  zu  $C$  zu sehen ist und  $M$  auf  $m$  abbildet, beschrieben.

Ein Punkt  $M$  in einem dreidimensionalen Weltkoordinatensystem wird mit Hilfe einer Kamera, die in einem eigenen dreidimensionalen Kamerakoordinatensystem beschrieben wird, auf die Bildebene  $I$  projiziert. Die Bildebene  $I$  ist durch ein zweidimensionales Bildkoordinatensystem beschrieben. Der projizierte Punkt  $m$  kann mit einem Sensor aufgenommen und abgespeichert werden.

Im folgenden wird zuerst ein Kameramodell eingeführt um die Projektion auf die Bildebene zu beschreiben. Daraufhin werden Koordinatentransformationen eingeführt um abschließend die Aufnahme eines Punktes mit einer willkürlichen Kameraorientierung zu berechnen.

### 2.1 Lochkameramodell zur Abbildung eines Punktes auf die Bildebene

Mit Hilfe des Lochkameramodells wird die Abbildung eines Objektes auf eine Bildebene beschrieben. Das Modell beruht ausschließlich auf der geometrischen Optik und vernachlässigt physikalische Effekte, wie Beugung oder die Auswirkung der Linse[6]. Das Lochkameramodell besteht aus einem Projektionszentrum  $C$ .  $C$  beschreibt gleichzeitig die Lage des Kamerazentrums und bildet den Ursprung des Kamerakoordinatensystems.[7, 3]. Die Blickrichtung der Kamera wird als Hauptachse bezeichnet. Die

Bildebene steht senkrecht zu Hauptachse und der Schnittpunkt der Hauptachse mit der Bildebene bildet den Hauptpunkt  $HP$ . Der Hauptpunkt ist der Ursprung des Bildebene koordinatensystems. Der Abstand vom Projektionszentrum zum Hauptpunkt wird als Brennweite  $\zeta$  beschrieben[3, 7]. Der Bildpunkt  $m$  entsteht am Schnittpunkt der Verbindungsgerade von  $C$  und  $M$  mit der der Bildebene  $I$ .

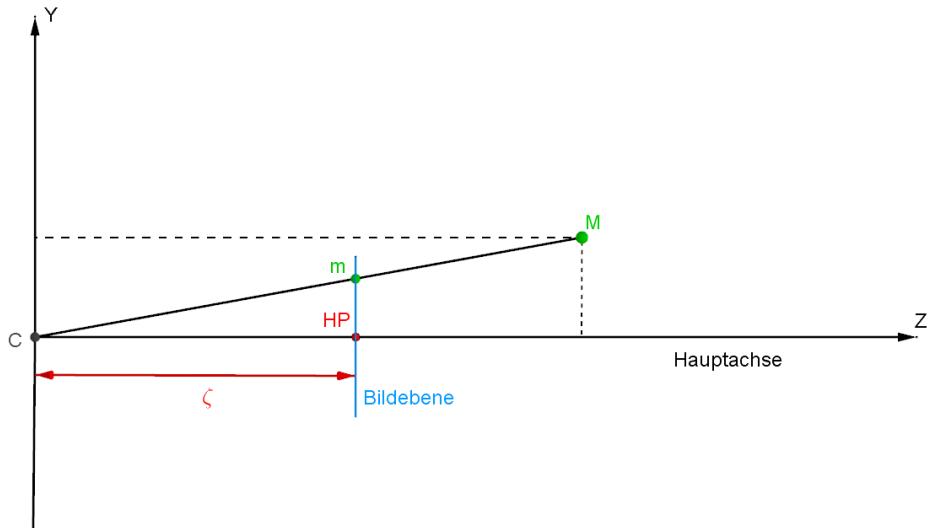


Abbildung 2.2: Die Abbildung zeigt einen Querschnitt des beschriebenen Lochkameramodells. Zu sehen ist das Projektionszentrum  $C$  der Kamera.  $C$  ist gleichzeitig das Kamerazentrum und bildet den Ursprung für das Kamerakoordinatensystem.  $\zeta$  beschreibt den Abstand des Projektionszentrums zur Bildebene. Die Hauptachse beschreibt die Blickrichtung der Kamera. Der Punkt an dem die Hauptachse die Bildebene schneidet wird Hauptpunkt genannt und ist gleichzeitig der Ursprung für das Bildebene koordinatensystem. Der Bildpunkt  $m$  entsteht am Schnittpunkt der Verbindungsgerade von  $C$  und  $M$  mit der der Bildebene  $I$

Die Projektion eines dreidimensionalen Punktes auf eine zweidimensionale Bildebene, wird durch eine  $3 \times 3$  Kameramatrix  $K_0$  beschrieben.

$$K_0 \cdot M = \begin{bmatrix} \zeta & 0 & 0 \\ 0 & \zeta & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{pmatrix} \zeta X \\ \zeta Y \\ Z \end{pmatrix} \mapsto \begin{pmatrix} \zeta \frac{X}{Z} \\ \zeta \frac{Y}{Z} \\ 1 \end{pmatrix} \quad (2.1)$$

Die Koordinaten auf der zweidimensionalen Bildebene werden häufig als homogene Koordinaten angegeben. Dazu werden die Koordinaten mit  $Z$  normiert und somit die Koordinaten auf die Ebene  $(x, y, 1)^T$  projiziert wird. Zur Vereinfachung wird zuletzt nur die x,y Koordinaten des entstandenen Bildes angegeben. Gleichung 2.1 beschreibt somit die Abbildung eines Punktes auf die Bildebene.

## 2.2 Koordinatentransformation

Um einen Punkt von einem übergeordneten Weltkoordinatensystem in ein bestimmtes zum Weltkoordinatensystem rotiertes Kamerakoordinatensystem zu überführen ist eine Transformation notwendig. Im folgenden wird der mathematische Weg einer Transformation eines Weltkoordinatensystem  $(O, \delta)$  mit  $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O)$  in ein Kamerakoordinatensystem  $(C, \beta)$  mit  $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3, C)$  beschrieben.

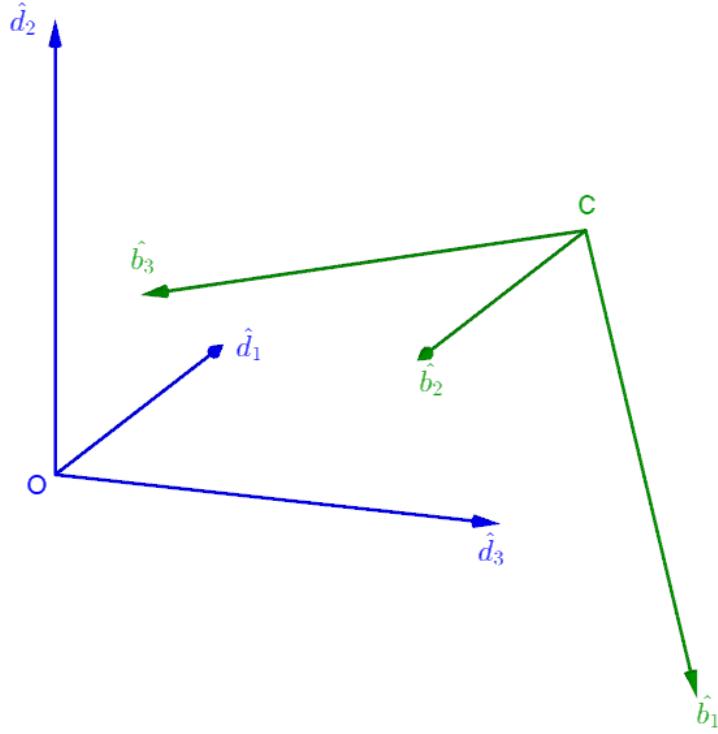


Abbildung 2.3: Ein Weltkoordinatensystem  $(O, \delta)$  mit  $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O)$  wird zu einem dazu verschobenen und rotiertem Kamerakoordinatensystem  $(C, \beta)$  mit  $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3, C)$  transformiert

Zunächst wird eine Koordinatisierung von Punkten im Weltkoordinatensystem vorgenommen. Ein Punkt  $P_\delta$  bezüglich des Weltkoordinatensystems wird wie folgt beschrieben:

$$P_\delta = O + p_{1\delta}\hat{d}_1 + p_{2\delta}\hat{d}_2 + p_{3\delta}\hat{d}_3 \quad (2.2)$$

$$\rightsquigarrow P_\delta = (p_{1\delta}, p_{2\delta}, p_{3\delta})^T = \begin{pmatrix} p_{1\delta} \\ p_{2\delta} \\ p_{3\delta} \end{pmatrix}. \quad (2.3)$$

Zwischen den beiden Koordinatensystemen  $(O, \delta)$  und  $(C, \beta)$  gelten die folgenden Beziehungen:

$$C_\beta = O_\delta + C_{\beta,1}\hat{d}_1 + C_{\beta,2}\hat{d}_2 + C_{\beta,3}\hat{d}_3 \quad (2.4)$$

$$\hat{b}_1 = b_{11}\hat{d}_1 + b_{12}\hat{d}_2 + b_{13}\hat{d}_3 \quad (2.5)$$

$$\hat{b}_2 = b_{21}\hat{d}_1 + b_{22}\hat{d}_2 + b_{23}\hat{d}_3 \quad (2.6)$$

$$\hat{b}_3 = b_{31}\hat{d}_1 + b_{32}\hat{d}_2 + b_{33}\hat{d}_3. \quad (2.7)$$

Diese Beziehungsgleichungen werden in Gleichung 2.2 eingesetzt.

$$\begin{aligned} P_\delta &= O + (C_{\beta,1} + p_{1\beta}b_{11} + p_{2\beta}b_{21} + p_{3\beta}b_{31}) \cdot \hat{d}_1 \\ &\quad + (C_{\beta,2} + p_{1\beta}b_{12} + p_{2\beta}b_{22} + p_{3\beta}b_{32}) \cdot \hat{d}_2 \\ &\quad + (C_{\beta,3} + p_{1\beta}b_{13} + p_{2\beta}b_{23} + p_{3\beta}b_{33}) \cdot \hat{d}_3 \end{aligned} \quad (2.8)$$

Aus Gleichung 2.8 wird ein Gleichungssystem in der Form von Gleichung 2.9 aufgestellt und gelöst.

$$\begin{aligned} p_{1\delta} &= C_{\beta,1} + (C_{\beta,1} + p_{1\beta}b_{11} + p_{2\beta}b_{21} + p_{3\beta}b_{31}) \\ \rightsquigarrow p_{1\delta} - C_{\beta,1} &= (C_{\beta,1} + p_{1\beta}b_{11} + p_{2\beta}b_{21} + p_{3\beta}b_{31}) \end{aligned} \quad (2.9)$$

Das Gleichungssystem lässt sich in Matrixform darstellen als

$$\begin{bmatrix} b_{11} & b_{21} & b_{31} \\ b_{12} & b_{22} & b_{32} \\ b_{13} & b_{23} & b_{33} \end{bmatrix} \begin{pmatrix} p_{1\beta} \\ p_{2\beta} \\ p_{3\beta} \end{pmatrix} = \begin{pmatrix} p_{1\delta} - C_{\beta,1} \\ p_{2\delta} - C_{\beta,2} \\ p_{3\delta} - C_{\beta,3} \end{pmatrix} \quad (2.10)$$

Wenn  $P_\beta$  gegeben ist, erhält man auf diese Weise direkt  $P_\delta$ . Die inverse Matrix  $D_\beta^{-1}$  kann verwendet werden um  $P_\beta$  aus  $P_\delta$  zu berechnen.

$$D_\beta^{-1} = \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix} \quad (2.11)$$

$$\rightsquigarrow \begin{pmatrix} p_{1\beta} \\ p_{2\beta} \\ p_{3\beta} \end{pmatrix} = D_\beta^{-1} \begin{pmatrix} p_{1\delta} - C_{\beta,1} \\ p_{2\delta} - C_{\beta,2} \\ p_{3\delta} - C_{\beta,3} \end{pmatrix} \quad (2.12)$$

Handelt es sich um ein kartesisches Koordinatensystem, so gilt  $D_\beta^{-1} = D_\beta^T$  und die transponierte Matrix kann für die Koordinatentransformation benutzt werden. Für zwei normierte, kartesische Koordinatensysteme ist  $D$  und  $D^T$  eine Rotationsmatrix  $R$ , weshalb im folgenden, analog zur Literatur [3, 5, 4],  $D^T = R$  angenommen wird. Um Gleichung 2.12 in einer kompakten Schreibweise zu formulieren, wird  $\vec{p}_\beta = (p_{1\beta}, p_{2\beta}, p_{3\beta})^T$  zu einem vierdimensionalen Vektor mit 1 zu  $\vec{p}_4\beta = (p_{1\beta}, p_{2\beta}, p_{3\beta}, 1)^T = (\vec{p}_\beta, 1)^T$  erweitert. Damit lässt sich Gleichung 2.12 als eine Matrixmultiplikation ausdrücken

$$\begin{pmatrix} p_{1\beta} \\ p_{2\beta} \\ p_{3\beta} \end{pmatrix} = D_\beta^T \begin{bmatrix} 1 & 0 & 0 & -C_{\beta,1} \\ 0 & 1 & 0 & -C_{\beta,2} \\ 0 & 0 & 1 & -C_{\beta,3} \end{bmatrix} \begin{pmatrix} p_{1\delta} \\ p_{2\delta} \\ p_{3\delta} \\ 1 \end{pmatrix} = R[I - C] \begin{pmatrix} p_{1\delta} \\ p_{2\delta} \\ p_{3\delta} \\ 1 \end{pmatrix} = T \begin{pmatrix} \vec{p}_\delta \\ 1 \end{pmatrix}. \quad (2.13)$$

Die Transformationsmatrix  $T$  setzt sich aus der Rotationsmatrix  $R$  und der Translationsmatrix  $[I| - C]$  zusammen und wirkt auf den neu definierten vierdimensionalen Vektor. Wichtig dabei ist, dass  $[I| - C]$  eine symbolische Schreibweise für eine  $3 \times 4$  Matrix ist.

## 2.3 Aufname mit einer willkürlichen Kameraorientierung

Ein beliebiger Punkt im Weltkoordinatensystem kann mit der eingeführten Operation auf die Bildecke und schließlich auch auf den Sensor projiziert werden. Es werden insgesamt vier verschiedene Koordinatensysteme definiert. Das Weltkoordinatensystem  $(O, \delta)$  mit  $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3)$ , das Kamerakoordinatensystem  $(C, \beta)$  mit  $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3)$ , das Bildeckenkoordinatensystem  $(I, \tau)$  mit  $\tau = (\hat{t}_1, \hat{t}_2)$  und als letztes das Sensorkoordinatensystem mit  $(S, \sigma)$  mit  $\sigma = (\hat{u}, \hat{v})$ . Abbildung 2.4 zeigt die Koordinatensysteme schematisch im Überblick.

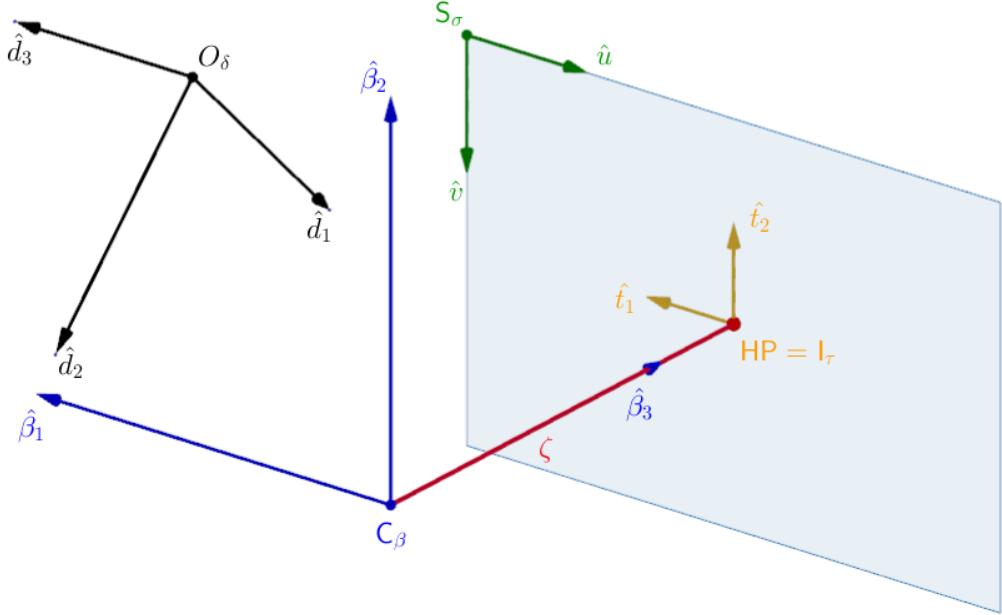


Abbildung 2.4: Das Schaubild zeigt die einzelnen Koordinatensysteme in einem Lochkameramodell. Das Weltkoordinatensystem  $(O, \delta)$  mit  $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3)$ , das Kamerakoordinatensystem  $(C, \beta)$  mit  $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3)$ , das Bildeckenkoordinatensystem  $(I, \tau)$  mit  $\tau = (\hat{t}_1, \hat{t}_2)$  und das Sensorkoordinatensystem  $(S, \sigma)$  mit  $\sigma = (\hat{u}, \hat{v})$ .

Für die Projektion eines Punktes  $M_\delta = (M_{x\delta} M_{y\delta} M_{z\delta})^T$  bezüglich des Weltkoordinatensystems in einen Punkt  $m_\tau = (m_{x\tau} m_{y\tau} m_{z\tau})^T$  bezüglich des Bildeckenkoordinatensystems kann eine Projektionsmatrix  $P$  definiert werden.

Zuerst muss der Punkt im Weltkoordinatensystem in das Kamerakoordinatensystem transformiert werden. Für die Transformation der Weltkoordinaten in Kamerakoordinaten gilt Gleichung 2.13:

$$\vec{M}_\beta = T \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = R \begin{bmatrix} 1 & 0 & 0 & -C_{\beta,1} \\ 0 & 1 & 0 & -C_{\beta,2} \\ 0 & 0 & 1 & -C_{\beta,3} \end{bmatrix} \begin{pmatrix} M_{x\delta} \\ M_{y\delta} \\ M_{z\delta} \\ 1 \end{pmatrix} = R[I - C] \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} \quad (2.14)$$

Nach der Transformation eines Punkts  $\vec{M}_\delta$  zu  $\vec{M}_\beta$  in das Kamerakoordinatensystem, erfolgt die Kameraprojektion von  $\vec{M}_\beta$  auf  $m_\beta$  wie in Gleichung 2.1 beschrieben.

$$\begin{bmatrix} m_{x\beta} \\ m_{y\beta} \\ m_{z\beta} \end{bmatrix} = \begin{bmatrix} \zeta & 0 & 0 \\ 0 & \zeta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_{x\beta} \\ M_{y\beta} \\ M_{z\beta} \end{bmatrix} = K_0 \vec{M}_\beta \quad (2.15)$$

Die Projektion eines Punktes  $\vec{M}_\delta$  auf den Bildpunkt  $\vec{m}_\beta$  kann durch Gleichung 2.14 und 2.15 zusammengefasst werden

$$\vec{m}_\beta = K_0 R [I - C] \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} \quad (2.16)$$

Um den Bildpunkt  $\vec{m}_\beta$  bezüglich eines zweidimensionalen Bildebene koordinatensystems anzugeben, wird die Bildebene mit der Tiefkomponente  $m_{z\beta}$  normiert, sodass  $m_{z\beta}$  auf den zweidimensionalen Raum der Bildebene gemappt wird. Diese Projektion wird durch folgende Gleichung beschrieben:

$$\vec{m}_\tau = \begin{bmatrix} m_{x\beta}/m_{z\beta} \\ m_{y\beta}/m_{z\beta} \\ 1 \end{bmatrix} \quad (2.17)$$

Zuletzt folgt die Transformation der Bildebene koordinaten auf den Sensorchip. Der Sensorchip besteht aus einer Ansammlung von Sensorelementen. Diese Sensorelemente können verschiedene Formen annehmen. Die meisten Sensorchips bestehen aus rechtwinkligen, rechteckigen Sensorelementen. Aus diesem Grund wird ein rechtwinkliges Sensorelement mit einer Größe  $lx ly$  angenommen. Diese Sensorelementgröße  $lx ly$  definiert auch die Pixelgröße und bildet die Längenskalierung des Sensorkoordinatensystems. Neben der unterschiedlichen Skalierung, wird der Ursprung des Sensorkoordinatensystems in der Regel an einer Ecke des Sensorchips definiert, sodass die Transformation von Bildebene koordinaten in Sensorkoordinaten auch eine Translation  $(V_{x\sigma}, V_{y\sigma})$  aufweist[3, 8]. Für einen Punkt  $m_\sigma = (u, v, 1)$  auf dem Sensorkoordinatensystem lassen sich die folgenden Bedingungen herleiten.

$$u = m_{\tau x} k_x - V_{x\sigma} \quad (2.18)$$

$$v = m_{\tau y} k_y - V_{y\sigma} \quad (2.19)$$

$$1 = 1 \quad (2.20)$$

$k_x = 1/lx$  und  $k_y = 1/ly$  ist die Pixeldichte in  $\frac{\text{pixel}}{m}$ . Es wird angemerkt, dass in der Bildebene und der Sensorebene die Punkte ausschließlich im zweidimensionalen Raum definiert sind. Aus den normierten Bildkoordinaten lässt sich somit folgende Sensormatrix bilden:

$$\vec{m}_\sigma = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} k_x & 0 & V_{\sigma,x} \\ 0 & k_y & V_{\sigma,y} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} m_{x\tau} \\ m_{y\tau} \\ 1 \end{bmatrix} = R_\sigma \vec{m}_\tau \quad (2.21)$$

Mit Hilfe der Transformationsmatrix  $R_\sigma$  kann ein Punkt von der Bildebene auf das Sensorelement projiziert werden.

Der hier skizzierte Lösungsweg beschreibt die Bildaufnahme eines Punktes im Lochkameramodell. Die hier eingeführte Projektionsmatrix  $P = K_0 R [I - C]$  gilt für die Abbildung eines Punktes auf den Bildpunkt. Der Bildpunkt wird normiert und in das Sensorkoordinatensystem umgerechnet wird. In der Literatur wird häufig die Transformation in das Sensorkoordinatensystem bereits in der Kameramatrix zusammengefasst. Damit bildet sich die erweiterte Kameramatrix  $K$

$$K = R_\sigma K_0 = \begin{bmatrix} k_x \zeta & 0 & V_{\sigma,x} \\ 0 & k_y \zeta & V_{\sigma,y} \\ 0 & 0 & 1 \end{bmatrix} \quad (2.22)$$

Mit dieser Kameramatrix wird eine neue Projektionsmatrix mit  $P = KR[I - C]$  gebildet, die einen Objektpunkt auf einen Bildpunkt im Sensorkoordinatensystem abbildet. Durch die Normierung dieses Punktes kann auf den direkten Sensorpunkt geschlossen werden. Ein weiterer Vorteil der erweiterten Kameramatrix  $K$  ist, dass sie die Pixeldichte  $k_x, k_y$  und die Brennweite  $\zeta$  beinhaltet. Diese Parameter

werden im folgenden als intrinsische Kameraparameter bezeichnet. Die Koordinatentransformationsmatrix  $R$  wird im Gegensatz aus den sogenannten extrinsischen Kameraparameter, der Kameraposition und Orientierung, definiert. Sind sowohl die intrinsischen wie auch extrinsischen Kameraparameter vorbestimmt kann somit  $P$  bestimmt werden und mit dem hier beschriebenen Lösungsweg das Bild konstruiert werden.

# 3 Geometrische Beziehungen zwischen Punktekorrespondenzen

Das Ziel dieser Masterarbeit ist es Punkte im dreidimensionalen Raum aus einer stereoskopischen Aufnahme zweier Kameras zu rekonstruieren. Bissher wurde die Bildaufnahme einer einzigen Kamera betrachtet. Jedoch kann eine Kamera allein nicht räumlich sehen. Um dreidimensionale Szenen aus Bildern zu rekonstruieren, müssen mindestens zwei Aufnahmen der gleichen Szene aus unterschiedlichen Blickwinkeln aufgenommen werden. Innerhalb dieser Aufnahmen müssen Punktekorrespondenzen gesucht werden. Korrespondierende Punkte zeichnen sich dadurch aus, dass sie die Abbildungen desselben Ursprungspunktes im Raum sind. Für diese Punkte muss eine gemeinsame Abbildungsvorschrift aufgestellt werden. Die Abbildungsvorschrift wird in einer  $3 \times 3$ -Matrix  $H$  ausgedrückt, welche die Transformationsmatrizen, sowie die Kameramatrizen zusammenfasst. Die  $3 \times 3$ -Matrix, kann aus gegebenen Punktekorrespondenzen abgeleitet werden. Aus  $H$  können Rückschlüsse auf die Kamera-parameter der beiden Kameras gezogen werden.

Es seien  $m_\tau = (m_{x\tau}, m_{y\tau}, m_{z\tau})^T$  die homogenen Koordinaten eines Punktes auf der Bildebene  $(I, \tau)$  und  $m'_{\tau'} = (m'_{x\tau'}, m'_{y\tau'}, m'_{z\tau'})^T$  der dazu korrespondierende Punkt der Bildebene  $(I', \tau')$ . Gesucht wird eine Abbildungsvorschrift welche als Matrix  $H$  ausgedrückt wird:

$$m'_{\tau'} = Hm_\tau \quad (3.1)$$

$$Hm_\tau = \begin{bmatrix} h_1^T \cdot m_{x\tau} \\ h_2^T \cdot m_{y\tau} \\ h_3^T \cdot m_{z\tau} \end{bmatrix} \quad (3.2)$$

$$\rightsquigarrow m'_{\tau'} = Hm_\tau = \begin{bmatrix} h_{11}m_{x\tau} + h_{12}m_{y\tau} + h_{13}m_{z\tau} \\ h_{21}m_{x\tau} + h_{22}m_{y\tau} + h_{23}m_{z\tau} \\ h_{31}m_{x\tau} + h_{32}m_{y\tau} + h_{33}m_{z\tau} \end{bmatrix} \quad (3.3)$$

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

In den ersten beiden Abschnitten werden zwei Herleitungen der Abbildungsvorschriften zwei unterschiedlicher Fälle gesucht. Im ersten Fall wird vorausgesetzt, dass die 3D-Punkte im Raum auf einer Ebene liegen und auf die Bildebenen von zwei zueinander verschobenen und rotierten Kameras abgebildet werden. Im zweiten Fall werden die Punkte eines komplexeren 3D-Objektes auf die beiden Bildebenen abgebildet. Im letzten Abschnitt wird die Herleitungen der entstehenden Matrizen beider Fälle anhand von Punktekorrespondenzen aufgezeigt.

## 3.1 Korrespondenzen planarer Punktmengen mit Homographien

Eine Abbildungsvorschrift kann in bestimmten Fällen eindeutig bestimmt werden. In diesen Fällen nennt man die Abbildung zwischen beiden zweidimensionalen Bildern Homographie[3, 5, 9]. In diesem Kapitel wird der beispielhafte Fall behandelt, dass Punkte auf der  $x, y$ -Ebene im Weltkoordinatensystem auf zwei unterschiedlichen Kameras  $C$  und  $C'$  abgebildet wird. Dies ist in Abbildung 3.1 dargestellt.

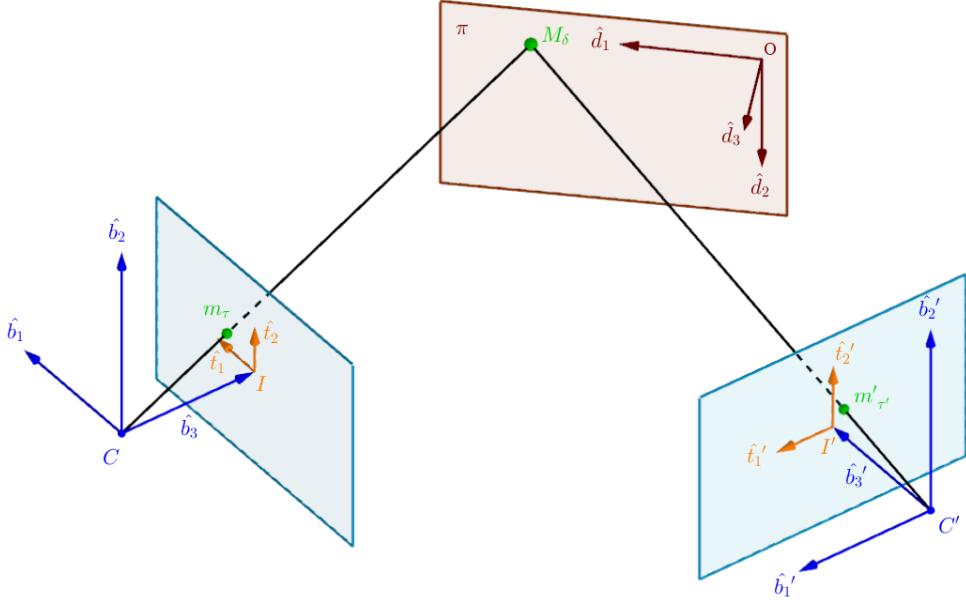


Abbildung 3.1: In der Abbildung sind die beiden Kameras  $C$  und  $C'$  mit ihren Bildebenen  $I$  und  $I'$  zu sehen. Ein Objektpunkt  $M_\delta$  bezüglich eines Weltkoordinatensystems  $(O, \delta)$  befindet sich auf der Ebene  $\pi$ , welche durch die Achsen  $\hat{d}_1$  und  $\hat{d}_2$  aufgespannt wird.  $M_\delta$  wird auf  $I$  und  $I'$  projiziert. Es entstehen die Bildpunkte  $m_\tau$  und  $m'_\tau$ .

Um die Abbildungsvorschrift herzuleiten beginnen wir bei dem Bildpunkt  $m_\tau = (m_{\tau,1}, m_{\tau,2}, m_{\tau,3})^T$  mit  $m_{\tau,3} = 1$ . Während in der Bildaufnahme ein Punkt  $m_\beta$  durch Division mit der  $m_{\beta,3}$ -Komponenten eindeutig zu  $m_\tau$  wird ist die Rückrichtung nicht eindeutig. Alle Punkte auf den Geraden von  $C$  und  $m_\tau$ , siehe Abbildung 3.2, werden auf denselben Bildpunkt projiziert. Die Projektion von  $m_\tau$  auf  $m_\beta$  ist demnach nicht eindeutig. Die Gerade mit allen möglichen Punkten  $m_\beta$  kann als  $\gamma m_\tau$ , mit der freien Variablen  $\gamma > 0$  ausgedrückt werden.

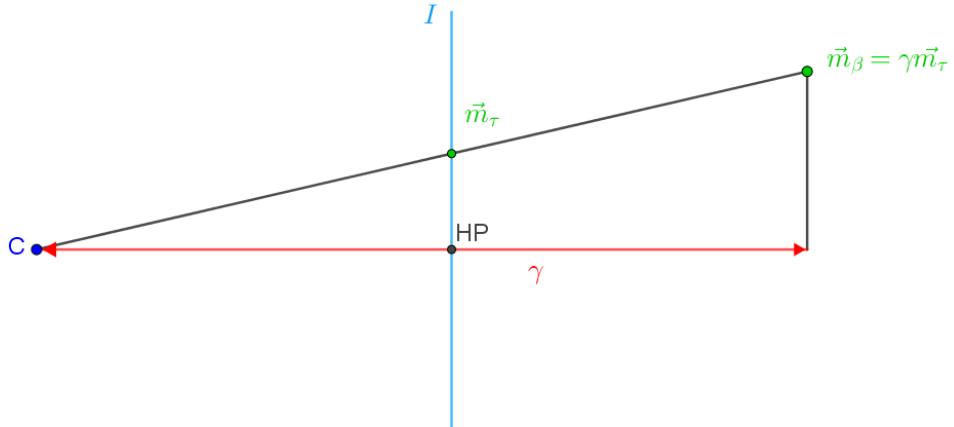


Abbildung 3.2: Auf der Geraden durch  $C$  und  $\vec{m}_\tau$ , befinden sich alle möglichen Punkte für  $m_\beta$ .  $m_\beta$  wird auf Grund seiner Unbestimmtheit als  $\gamma m_\tau$  bezeichnet

Für zwei korrespondierende Bildpunkte  $m_\tau$  und  $m'_\tau$ , kann für alle möglichen Punkte  $m_\beta$  und  $m'_\beta$  die folgende Projektionsvorschrift hergeleitet werden [5].

$$\gamma \vec{m}_\tau = P \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = [KR| - KR\vec{C}_\delta] \cdot \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = KR(\vec{M}_\delta - \vec{C}_\delta) \quad (3.5)$$

$$\gamma' \vec{m}'_\tau = P' \begin{bmatrix} \vec{M}'_\delta \\ 1 \end{bmatrix} = [K'R'| - K'R'\vec{C}'_\delta] \cdot \begin{bmatrix} \vec{M}'_\delta \\ 1 \end{bmatrix} = K'R'(\vec{M}'_\delta - \vec{C}'_\delta) \quad (3.6)$$

Mit einem Ursprungspunkt  $M_\delta = (x_\delta, y_\delta, 0)^T$  auf der x,y Ebene im Weltkoordinatensystem und einer unbekannten Projektionsmatrix P mit

$$P = \begin{bmatrix} p_{11} & p_{21} & p_{31} & p_{41} \\ p_{12} & p_{22} & p_{32} & p_{42} \\ p_{13} & p_{23} & p_{33} & p_{43} \end{bmatrix} = [p_1 \ p_2 \ p_3 \ p_4] \quad (3.7)$$

kann die folgende Gleichung aufgestellt werden [5]

$$\gamma m_\tau = P \cdot \begin{bmatrix} M_\delta \\ 1 \end{bmatrix} = [p_1 \ p_2 \ p_3 \ p_4] \cdot \begin{bmatrix} x_\delta \\ y_\delta \\ 0 \\ 1 \end{bmatrix} = [p_1 \ p_2 \ p_4] \cdot \begin{bmatrix} x_\delta \\ y_\delta \\ 1 \end{bmatrix} = G \vec{m}_\delta \quad (3.8)$$

$$\gamma' m'_{\tau'} = P \cdot \begin{bmatrix} M_\delta \\ 1 \end{bmatrix} = [p'_1 \ p'_2 \ p'_3 \ p'_4] \cdot \begin{bmatrix} x_\delta \\ y_\delta \\ 0 \\ 1 \end{bmatrix} = [p'_1 \ p'_2 \ p'_4] \cdot \begin{bmatrix} x_\delta \\ y_\delta \\ 1 \end{bmatrix} = G' \vec{m}_\delta. \quad (3.9)$$

Aus  $\gamma m_\tau = G \cdot m_\delta$  und  $\gamma' m'_{\tau'} = G' \cdot m_\delta$  kann dann folgendes abgeleitet werden[5].

$$\gamma' m'_{\tau'} = G' G^{-1} \gamma m_\tau \quad (3.10)$$

Mit  $\lambda = \frac{\gamma'}{\gamma}$ , kann Gleichung 3.10 dann wieder umformuliert werden und in die Bedienungsgleichung der Homographie mit  $H = G' G^{-1}$  umgeformt werden:

$$\lambda m'_{\tau'} = H m_\tau \quad (3.11)$$

Die entstandene Homographie  $H$  ist somit eine Abbildungsvorschrift welche zwei korrespondierende Punkte in Verbindung setzt. Diese Homographiebedingung stellt ein Gleichungssystem mit 9 unbekannten, welche in Kapitel 3.2 gelöst wird[3].

### 3.2 Korrespondenzanalyse für beliebige Punkte im Raum (Epipolare Geometrie)

Für Bilder von komplexeren dreidimensionalen Objekten, bei denen die Punkte auf verschiedenen Ebenen liegen können, kann keine Homographiebedingung hergestellt werden um die Kameraparameter zu bestimmen. Jedoch kann auf geometrische Bedingungen zurückgegriffen werden, um die Abbildungsvorschrift zwischen den Bildern auszunutzen, um die Kameraparameter beider Kameras zu bestimmen. Ein Ursprungspunkt  $M_\delta$  wird wieder mit zwei Kameras  $C$  und  $C'$  aufgenommen. In Abbildung 3.3 ist das stereoskopische System dargestellt .

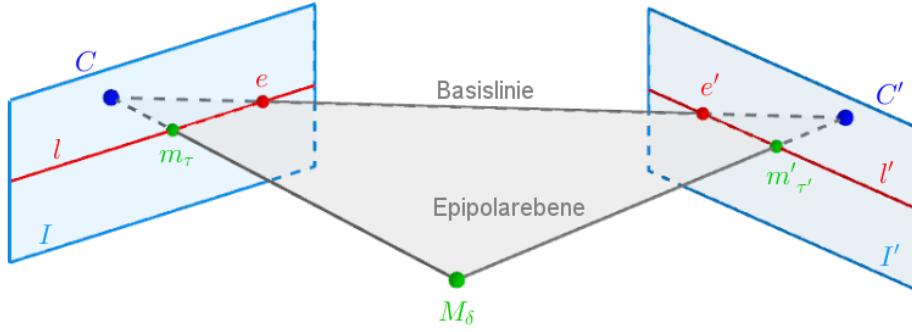


Abbildung 3.3:  $C$  und  $C'$  sind die Projektionszentren zweier Kameras. Beide Kameras besitzen jeweils eine Bildebene. Die Basislinie verbindet die Projektionszentren der Kameras. Die Punkte an welchen die Basislinie die Bildebene schneidet, werden als Epipole  $e$  und  $e'$  bezeichnet. Durch einen Epipol verlaufen alle Epipolarlinien des Bildes.  $M_\delta$  ist der Objektpunkt im 3D-Raum und  $m_\tau$  und  $m'_{\tau'}$  sind die jeweiligen Abbildungen dieses Punktes auf den Bildebene. Die Verbindungsvektoren zwischen  $C, C'$  und  $M_\delta$  bilden die sogenannte Epipolarebene[10, 11, 3, 1].

Es werden hier einige geometrische Definitionen eingeführt um die danach folgende mathematische Herleitung genauer zu verstehen. Die Vektoren  $\overrightarrow{CM} = (\vec{M}_\delta - \vec{C}_\delta)$ ,  $\overrightarrow{C'M} = (\vec{M}_\delta - \vec{C}'_\delta)$  und  $\overrightarrow{CC'} = (\vec{C}'_\delta - \vec{C}_\delta)$  definieren die Epipolarebene, die durch das schwarze Dreieck in Abbildung 3.3 gekennzeichnet ist. Die Schnittpunkte der Geraden zwischen  $C$  und  $C'$  mit der jeweiligen Bildebene  $I$  und  $I'$  werden als Epipole  $e$  und  $e'$  bezeichnet. Die Schnittgerade der Epipolarebene mit  $I$  und  $I'$  bilden die sogenannten Epipolarlinien  $l$  und  $l'$ [3, 12, 13, 11].

Ein Bildpunkt  $m_i$  auf der Bildebene  $I$  wird zuerst auf die Gerade, die durch  $m_i$  und  $C$  geht abgebildet. Die Gerade stellt alle möglichen Ursprungspunkte zu  $m_i$  dar. Dies ist durch die drei möglichen Punkte  $M_1, M_2, M_3$  in Figur 3.4 dargestellt. Jeder dieser Punkte wird nun wiederum auf  $I'$  projiziert. Die so entstandenen Punkte liegen alle auf der Epipolarlinie  $l'$ [3].

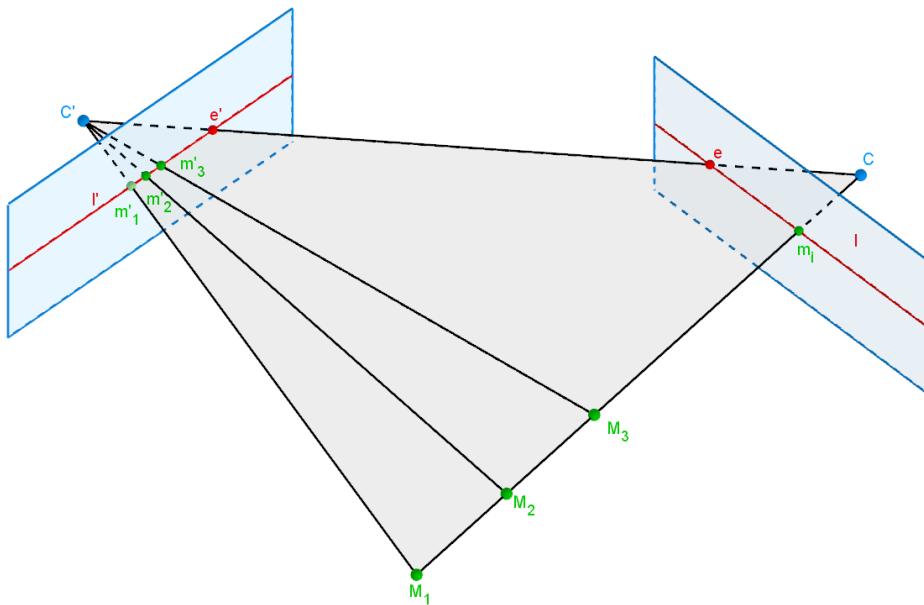


Abbildung 3.4: Die Objektpunkte  $M_1, M_2$  und  $M_3$  werden in  $I'$  als  $m'_1, m'_2$  und  $m'_3$  abgebildet, während sie in  $I$  immer den selben Bildpunkt  $m_1$  ergeben.

Die hier gezeigte Abbildung von  $m_i$  auf  $l'$  wird nun genauer betrachtet. Es werden wieder die Gleichungen für  $m_\tau$  und  $m'_{\tau'}$ , wie in Gleichung 3.6, aufgestellt

$$\gamma \vec{m}_\tau = P \begin{bmatrix} \vec{M}_\delta \\ 1 \end{bmatrix} = [KR| - KR\vec{C}_\delta] \cdot \begin{bmatrix} \vec{M}\delta \\ 1 \end{bmatrix} = KR(\vec{M}\delta - \vec{C}_\delta) \quad (3.12)$$

$$\gamma' \vec{m}'_{\tau'} = P' \begin{bmatrix} \vec{M}'_\delta \\ 1 \end{bmatrix} = [K'R'| - K'R'\vec{C}'_\delta] \cdot \begin{bmatrix} \vec{M}\delta \\ 1 \end{bmatrix} = K'R'(\vec{M}\delta - \vec{C}'_\delta) \quad (3.13)$$

Gleichungen 3.12 und 3.13 werden nach  $(\vec{M} - \vec{C}_\delta)$  und  $(\vec{M} - \vec{C}'_\delta)$  aufgelöst.

$$\gamma R^T K^{-1} \vec{m}_\tau = (\vec{M} - \vec{C}_\delta) \quad (3.14)$$

$$\gamma' R'^T K'^{-1} \vec{m}'_{\tau'} = (\vec{M} - \vec{C}'_\delta) \quad (3.15)$$

Wie in Abbildung 3.3 gezeigt bilden Vektoren  $(\vec{M}_\delta - \vec{C}_\delta)$ ,  $(\vec{M}_\delta - \vec{C}'_\delta)$  und  $(\vec{C}'_\delta - \vec{C}_\delta)$  ein Dreieck. Für dieses Dreieck kann die folgende Gleichung aufgestellt werden.

$$(\vec{C}'_\delta - \vec{C}_\delta) = (\vec{M}_\delta - \vec{C}_\delta) - (\vec{M}_\delta - \vec{C}'_\delta) \quad (3.16)$$

$(\vec{M} - \vec{C}_\delta)$  und  $(\vec{M} - \vec{C}'_\delta)$  können durch die Ausdrücke in den Gleichungen 3.14 und 3.15 ersetzt werden um folgende Gleichung zu erhalten

$$(\vec{C}'_\delta - \vec{C}_\delta) = \gamma' R^T K^{-1} \vec{m}_\tau - \gamma R'^T K'^{-1} \vec{m}'_{\tau'}. \quad (3.17)$$

Durch Vektoridentitäten können  $\gamma$  und  $\gamma'$  eliminiert werden und folgende Bedingung aus Gleichung 3.17 hergeleitet werden [14]

$$0 = \vec{m}'_\tau^T (K'^{-1})^T R' [\vec{C}'_\delta - \vec{C}_\delta] \times R^T K^{-1} \vec{m}_\tau = \vec{m}'_\tau^T F \vec{m}_\tau \quad (3.18)$$

mit

$$[\vec{C}'_\delta - \vec{C}_\delta] \times = \begin{bmatrix} 0 & -(C'_{z\delta} - C_{z\delta}) & C'_{y\delta} - C_{y\delta} \\ C'_{z\delta} - C_{z\delta} & 0 & -(C'_{x\delta} - C_{x\delta}) \\ -(C'_{y\delta} - C_{y\delta}) & C'_{x\delta} - C_{x\delta} & 0 \end{bmatrix}. \quad (3.19)$$

In Gleichung 3.18 wurde die Bedingungsgleichung für die sogenannte Fundamentalmatrix  $F$  definiert [10]. Sind die Kameraparameter und dadurch die Kameramatrix  $K$  bekannt, so wird die essentielle Matrix  $E = R' [\vec{C}'_\delta - \vec{C}_\delta] \times R^T$  definiert. Gleichung 3.18 kann zu einer Bedingung für die  $E$  umgeformt werden.

$$\vec{m}'_\tau^T K'^{-T} E K^{-1} \vec{m}_\tau = 0 \quad (3.20)$$

Gleichung 3.18 und 3.20 definieren den sogenannten *Epipolar-Constraint*[3, 10] und können verwendet werden um die Fundamentalmatrix oder essentielle Matrix aus bekannten Korrespondierenden Punkten zu bestimmen. Für die essentielle Matrix müssen zuvor noch die Koordinaten in der Form

$$\vec{m}'_\tau = \vec{m}'_\tau^T K'^{-T} \quad (3.21)$$

$$\vec{m}_\tau = K^{-1} \vec{m}_\tau \quad (3.22)$$

umgerechnet werden[3, 15]. Der verwendete Algorithmus zur Bestimmung von  $F$  wird in Kapitel 3.3 näher beschrieben.

Wenn die Fundamentalmatrix bekannt ist, können auch die Epipole  $e$  und  $e'$  und Epipolarlinien  $l$  und  $l'$  aus Eigenschaften der Fundamentalmatrix bestimmt werden [3, 11, 16, 1, 15].

Um die Epipole  $e$  zu bekommen, wird der rechte Kern von  $F$  bestimmt und für  $e'$  muss der linke Kern von  $F$  bestimmt werden[3, 11, 16, 1, 15]. Es gilt also

$$Fe = 0 \quad (3.23)$$

$$F^T e' = 0. \quad (3.24)$$

Um die zu  $m$  oder  $m'$  korrespondierende Epipolarlinie  $l'$  oder  $l$  zu bestimmten kann die folgende Transformation verwendet werden[3, 11, 16, 1, 15]

$$l' = Fm \quad (3.25)$$

$$l = F^T m'. \quad (3.26)$$

Die Matrizen  $F$  und  $E$  sind mit diesen Eigenschaften wichtige Instrumente für die Bestimmung der extrinsischen und intrinsischen Kameraparameter und ihre Eigenschaften werden in den folgenden Kapiteln ausgenutzt um effiziente Rekonstruktionalgorithmen für die Szene zu implementieren.

### 3.3 Bestimmung von Homographie und Fundamentalmatrix aus Punktekorrespondenzen

Im folgenden wird gezeigt, wie beispielsweise eine Homographie, Fundamentalmatrix und dementsprechend auch eine essentielle Matrix aus Punktekorrespondenzen gewonnen werden können. Für essentielle Matrizen gilt das selbe Verfahren wie für die Fundamentalmatrizen nur sind hier die Punkte in der Form wie in Gleichung 3.22 gezeigt. Die Herleitung selbst wird am Beispiel der Fundamentalmatrix aufgezeigt.

Es wird davon ausgegangen, dass die Transformationsmatrizen  $R$  und  $R'$  sowie die Kameramatrizen  $K$  und  $K'$  nicht bekannt sind. Des Weiteren wird vorausgesetzt, dass zuvor mindestens acht korrespondierende Punkte aus den jeweiligen Bildpaaren detektiert wurden. Im Realfall, werden hierfür bestimmte Detektionsalgorithmen verwendet, wie Beispielsweise der SURF-Algorithmus[17], welche markante Bildpunkte in beiden Bildern suchen.

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  $A$  eingetragen, welche sich nach dem folgenden Schema aufstellen lässt[3, 5]. Ausgehend von der Abbildungsvorschrift aus Gleichung 3.11 gilt:

$$Hm_\tau = \lambda m'_{\tau'} \quad (3.27)$$

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} m_\tau \\ m_{\tau'} \end{bmatrix} = \begin{bmatrix} \lambda m'_{\tau'} \\ \lambda m'_{y\tau'} \\ \lambda m'_{z\tau'} \end{bmatrix} \quad (3.28)$$

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} m_{x\tau} \\ m_{y\tau} \\ m_{z\tau} \end{bmatrix} = \begin{bmatrix} \lambda m'_{x\tau'} \\ \lambda m'_{y\tau'} \\ \lambda m'_{z\tau'} \end{bmatrix} \quad (3.29)$$

Aus Gleichung 3.29 lässt sich das folgende Gleichungssystem aufstellen.

$$h_{11}m_{x\tau} + h_{12}m_{y\tau} + h_{13}m_{z\tau} = \lambda m'_{x\tau'} \quad (3.30)$$

$$h_{21}m_{x\tau} + h_{22}m_{y\tau} + h_{23}m_{z\tau} = \lambda m'_{y\tau'} \quad (3.31)$$

$$h_{31}m_{x\tau} + h_{32}m_{y\tau} + h_{33}m_{z\tau} = \lambda m'_{z\tau'} \quad (3.32)$$

Da mit zweidimensionalen homogenen Bildkoordinaten gearbeitet wird und somit  $m_{z\tau}$  und  $m'_{z\tau'} = 1$  ist, ergibt sich für die letzte Zeile  $h_{31}m_{x\tau} + h_{32}m_{y\tau} + h_{33}m_{z\tau} = \lambda$ . Setzt man diesen Ausdruck anstelle von  $\lambda$  in die anderen beiden Gleichungen ein, so ergeben sich:

$$h_{11}m_{x\tau} + h_{12}m_{y\tau} + h_{13}m_{z\tau} = (h_{31}m_{x\tau} + h_{32}m_{y\tau} + h_{33}m_{z\tau}) \cdot m'_{x\tau'} \quad (3.33)$$

$$h_{21}m_{x\tau} + h_{22}m_{y\tau} + h_{23}m_{z\tau} = (h_{31}m_{x\tau} + h_{32}m_{y\tau} + h_{33}m_{z\tau}) \cdot m'_{y\tau'} \quad (3.34)$$

Für den Aufbau von  $A$  werden beide Ausdrücke nach Null aufgelöst, so dass sich pro korrespondierendem Punktpaar zwei Gleichungen nach 3.35 und 3.36 ergeben.

$$h_{11}m_{x\tau} + h_{12}m_{y\tau} + h_{13}m_{z\tau} - (h_{31}m_{x\tau} + h_{32}m_{y\tau} + h_{33}m_{z\tau}) \cdot m'_{x\tau'} = 0$$

$$h_{21}m_{x\tau} + h_{22}m_{y\tau} + h_{23}m_{z\tau} - (h_{31}m_{x\tau} + h_{32}m_{y\tau} + h_{33}m_{z\tau}) \cdot m'_{y\tau'} = 0$$

$$\rightsquigarrow h_{11}m_{x\tau} + h_{12}m_{y\tau} + h_{13}m_{z\tau} - h_{31}m_{x\tau} \cdot m'_{x\tau'} - h_{32}m_{y\tau} \cdot m'_{x\tau'} - h_{33}m_{z\tau} \cdot m'_{x\tau'} = 0 \quad (3.35)$$

$$\rightsquigarrow h_{21}m_{x\tau} + h_{22}m_{y\tau} + h_{23}m_{z\tau} - h_{31}m_{x\tau} \cdot m'_{y\tau'} - h_{32}m_{y\tau} \cdot m'_{y\tau'} - h_{33}m_{z\tau} \cdot m'_{y\tau'} = 0 \quad (3.36)$$

Die entstandenen Gleichungen werden dann nach folgendem Schema in die Koeffizientenmatrix  $A$  eingetragen.[5, 3, 18, 6]

$$A \cdot x = 0 \quad (3.37)$$

$$\begin{pmatrix} m_{x\tau} & m_{y\tau} & 1 & 0 & 0 & 0 & m_{x\tau}m'_{x\tau'} & m_{y\tau}m'_{x\tau'} & 1 \cdot m'_{x\tau'} \\ 0 & 0 & 0 & m_{x\tau} & m_{y\tau} & 1 & m_{x\tau}m'_{y\tau'} & m_{y\tau}m'_{y\tau'} & 1 \cdot m'_{y\tau'} \\ & & & \ddots & & & & & \\ & & & & \ddots & & & & \\ & & & & & \ddots & & & \\ m_{i,x\tau} & m_{i,y\tau} & 1 & 0 & 0 & 0 & m_{i,x\tau}m'_{i,x\tau'} & m_{i,y\tau}m'_{i,x\tau'} & 1 \cdot m'_{i,x\tau'} \\ 0 & 0 & 0 & m_{i,x\tau} & m_{i,y\tau} & 1 & m_{i,x\tau}m'_{i,y\tau'} & m_{i,y\tau}m'_{i,y\tau'} & 1 \cdot m'_{i,y\tau'} \end{pmatrix} \cdot \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_i \end{pmatrix} = 0 \quad (3.38)$$

Gesucht wird ein Vektor  $\vec{x}$ , für den gilt das  $A \cdot x = 0$ . Besitzt Matrix  $A$  einen Rang von 8, so entspricht der gesuchte Vektor  $\vec{x}$  dem Kern der Koeffizientenmatrix und ist ein Spaltenvektor mit insgesamt neun Einträgen, welche in die 3x3-Homographiematrix eingetragen werden können[3, 18].

Das Verfahren mit welchem sowohl  $F$  als auch  $E$  geschätzt werden können, ähnelt in seinem Aufbau dem der Bestimmung der Homographiematrix. Das Verfahren wird hier allgemein als der 8-Punkte-Algorithmus bezeichnet. Der 8-Punkte-Algorithmus ist eine lineare Technik, welche angewandt wird, um die Fundamentalmatrix aus  $n \geq 8$  Punkten schätzen zu können [3, 12, 4]. Der Algorithmus benötigt  $n \geq 8$  Punkte, um ein valides Ergebnis zu liefern [3, 12, 4]. Das Ergebnis und jedes seiner Vielfachen ist eine mögliche Lösung für  $F$ [3]. Der Algorithmus wird am Beispiel für die Bestimmung von  $F$  veranschaulicht. Zunächst wird eine Koeffizientenmatrix  $A$  aus Punktekorrespondenzen gebildet. Hierzu wird sich auf die für  $F$  hergeleitete Gleichung 3.18 bezogen.

$$\begin{aligned}
m'^T_{\tau'} \cdot F \cdot m_{\tau} &= 0 \\
F &= \begin{bmatrix} f_{11} & f_{122} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \\
\begin{bmatrix} m'_{x\tau'} & m'_{y\tau'} & 1 \end{bmatrix} \cdot \begin{bmatrix} f_{11} & f_{122} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \cdot \begin{bmatrix} m_{x\tau} \\ m_{y\tau} \\ 1 \end{bmatrix} &= 0 \\
f_{11}m_{x\tau}m'_{x\tau'} + f_{12}m_{y\tau}m'_{x\tau'} + f_{13}m'_{x\tau'} + f_{21}m_{x\tau}m'_{y\tau'} \\
+ f_{22}m_{y\tau}m'_{y\tau'} + f_{23}m'_{y\tau'} + f_{31}m_{x\tau} + f_{32}m_{y\tau} + f_{33} &= 0 \quad (3.39) \\
(m_{x\tau}m'_{x\tau'}, m_{y\tau}m'_{x\tau'}, m'_{x\tau'}, m_{x\tau}m'_{y\tau'}, m_{y\tau}m'_{x\tau'}, m'_{x\tau'}, m_{x\tau}, m_{y\tau}, 1) \cdot f &= 0 \\
\begin{bmatrix} m_{x\tau}m'_{x\tau'} & m_{y\tau}m'_{x\tau'} & m'_{x\tau'} & m_{x\tau}m'_{y\tau'} & m_{y\tau}m'_{y\tau'} & m'_{y\tau'} & m_{x\tau} & m_{y\tau} & 1 \\ \vdots & \vdots \\ m_{i,x\tau}m'_{i,x\tau'} & m_{i,y\tau}m'_{i,x\tau'} & m'_{i,x\tau'} & m_{i,x\tau}m'_{i,y\tau'} & m_{i,y\tau}m'_{i,y\tau'} & m'_{i,y\tau'} & m_{i,x\tau} & m_{i,y\tau} & 1 \end{bmatrix} \cdot \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} &= 0 \\
A \cdot f &= 0
\end{aligned}$$

Gesucht wird nun ein Vektor  $\vec{f}$ , für den gilt das  $A \cdot f = 0$ . Besitzt Matrix  $A$  einen Rang von 8, so entspricht der gesuchte Vektor  $\vec{x}$  auch hier dem Kern von  $A$  und ist ein Spaltenvektor mit insgesamt neun Einträgen, welche in die 3x3-Fundamentalmatrix eingetragen werden können[3, 1].

Bei der Homographie wie auch bei der Fundamentalmatrix, kann es zu überbestimmten Systemen kommen. Ein System gilt als überbestimmt, wenn es durch mehr Gleichungen als Unbekannte beschrieben wird[18, 19]. Für die Koeffizientenmatrix für  $F$  und  $H$  hätte das zur Folge, dass sie in ihrem Rang steigt. Die Bestimmung des Kerns würde in beiden Fällen kein eindeutiges Ergebnis mehr liefern[3, 18].

Für die Lösung überbestimmter Systeme wird durch ein *least-square-* Verfahren, mit Hilfe der Singulärwertszerlegung einer Matrix  $A$  eine Lösung für einen Vektor  $\vec{x}$  gesucht, so dass  $\| A \cdot x \|$  minimal wird [3, 19, 18]. Die Singulärwertzerlegung von  $A$  ist eine Faktorisierung der Matrix  $A \in \mathbb{R}^{m \times n}$  der Form  $A = U \cdot S \cdot V^T$  mit orthogonalen Matrizen  $U \in \mathbb{R}^{m \times n}$  und  $V \in \mathbb{R}^{m \times n}$  sowie mit einer Diagonalmatrix  $S$ .

$$S = \begin{pmatrix} s_1 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \ddots & \ddots & \ddots & & \ddots \\ \vdots & \ddots & \ddots & \ddots & & \ddots \\ \vdots & \ddots & \ddots & \ddots & & \ddots \\ 0 & \dots & s_r & 0 & \dots & 0 \\ 0 & \dots & 0 & 0 & \dots & 0 \\ \vdots & & \ddots & \ddots & & \ddots \\ \vdots & & \ddots & \ddots & & \ddots \\ 0 & \dots & 0 & 0 & \dots & 0 \end{pmatrix} \quad (3.40)$$

Die Diagonalmatrix  $S$  beinhaltet die Singulärwerte der Matrix. Dabei soll für die diagonalen Singulärwerte in  $S$  mit  $s_1$  bis  $s_r$  gelten, dass  $s_1 \geq s_2 \geq \dots \geq s_r \geq 0$ [19]. Die Spalte der Matrix  $V^T$ , welche mit dem kleinsten Singulärwert von  $S$  korrespondiert, ergibt den Vektor  $\vec{x}$ , für den  $\| A \cdot x \|$  minimal wird.

# 4 Synthetische Rekonstruktion

Anhand der erarbeiteten mathematischen Grundlagen ist ein Algorithmus für Rekonstruktion einer Szenen aus einer Stereobildaufnahme entstanden. Der Algorithmus wurde mit dem Ziel der Kamerakalibrierung und der Szenenrekonstruktion aus Bildquellen unterschiedlicher Auflösungen entwickelt, da Stereokalibrierungsverfahren einiger Computer Vision Applikationen kein unterschiedlichen Auflösungen von Kameras berücksichtigen. Der entwickelte Algorithmus ist sowohl in der Lage aus einem Stereobildpaar extrinsische Kameraparameter zu bestimmen und anhand dessen die 3D-Szene zu rekonstruieren, jedoch unter der Voraussetzung, dass die intrinsischen Kameraparameter beider Kameras bekannt sind.

Im Folgenden soll der Algorithmus anhand eines virtuellen Beispiels erklärt werden. Dabei werden die einzelnen Schritte des Aufbaus der virtuellen Szene, der Bestimmung der extrinsischen Kameraparameter und der Rekonstruktion der virtuellen 3D-Szene beschrieben. Abbildung 4.1 fasst den Arbeitsprozess des Szenenrekonstruktionsalgorithmus für das virtuelle Beispiel zusammen.

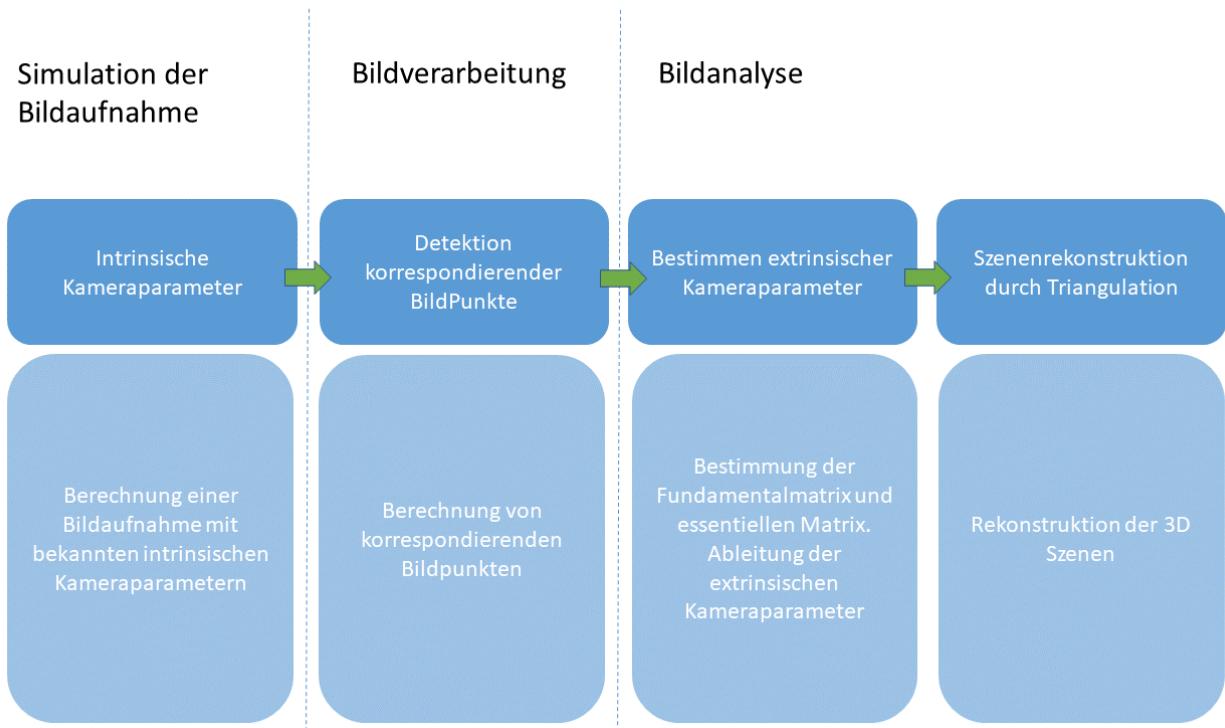


Abbildung 4.1: Ablaufdiagramm für das synthetische Beispiel

## 4.1 Simulierte Bildaufnahme einer virtuellen Szene

Als 3D-Objekt wurde ein Quader, in ein Weltkoordinatensystem  $(O, \delta)$  mit  $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3)$  positioniert. Es werden zwei Kameras  $(C, \beta)$  mit  $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3)$  und  $(C', \beta')$  mit  $\beta' = (\hat{b}'_1, \hat{b}'_2, \hat{b}'_3)$  in  $(O, \delta)$  platziert. Das Weltkoordinatensystem  $(O, \delta)$  und Das Kamerakoordinatensystem  $(C, \beta)$  sind deckungsgleich.  $C'$  ist relativ zu  $C$  verschoben und rotiert. Die zwei Bildebenen  $(I, \tau)$  mit  $\tau = (\hat{t}_1, \hat{t}_2, \hat{t}_3)$  und  $(I', \tau')$  mit  $\tau' = (\hat{t}'_1, \hat{t}'_2, \hat{t}'_3)$  sind vor  $C$  und  $C'$  positioniert. Die Sensorkoordinatensysteme  $(S, \sigma)$  und  $(S', \sigma')$  wurden gleich den Bildebenenkoordinatensystemen  $(I, \tau)$  und  $(I', \tau')$  gesetzt. Es wird von zwei

identischen Kameras ausgegangen und somit werden für den hier diskutierten Fall ausschließlich mit dem vereinfachten Kameramatrizen  $K_0$  gerechnet. Der schematische Aufbau der Szenen ist in den Abbildungen 4.2, 4.3 und 4.4 dargestellt.

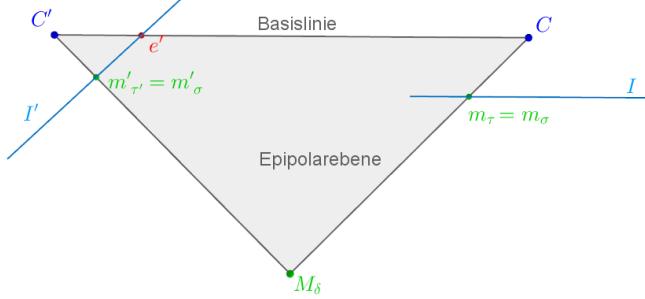


Abbildung 4.2: In der Abbildung ist der vereinfachte Stereoaufbau in einer Top-Down-Ansicht zu sehen

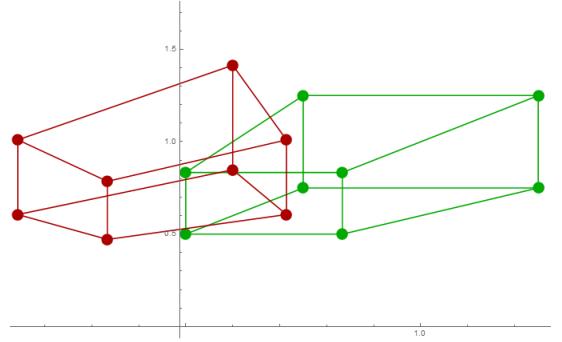


Abbildung 4.3: Simulierte Abbildung des Quaders auf die Kamera  $C$  in Grün und auf  $C'$  in rot

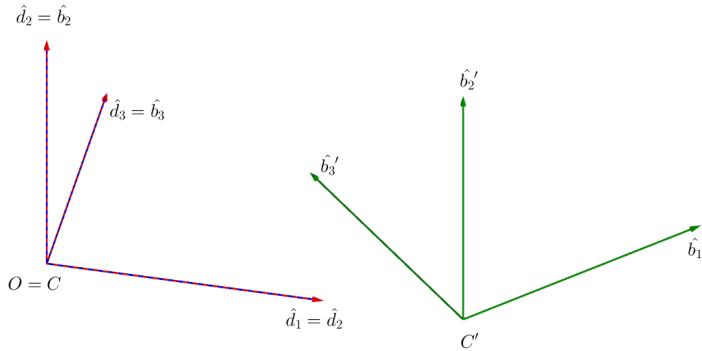


Abbildung 4.4: Abbildung der verschiedenen Kamerakoordinatensysteme, das Weltkoordinatensystem  $O$  ist zum Kamerakoordinatensystem  $C$  deckungsgleich und  $C'$  verschoben und rotiert.

Um die Eckpunkte des Quaders auf die Bildebenen von  $C$  und  $C'$  abbilden zu können, werden zunächst die Projektionsmatrizen  $P$  und  $P'$  aufgestellt.  $C$  ist Deckungsgleich mit dem Weltkoordinatensystem  $(O, \delta)$ . es gilt also:

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

$$C = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (4.2)$$

$$T = R[I] - C \quad (4.3)$$

$$T = \begin{bmatrix} 1 & 0 & 0 & C_1 \\ 0 & 1 & 0 & C_2 \\ 0 & 0 & 1 & C_3 \end{bmatrix} \quad (4.4)$$

$C'$  dagegen ist gegenüber  $C$  verschoben und rotiert. Als Beispiel wird eine Rotation um die  $\hat{b}_2'$  Achse für  $C'$  bestimmt. Somit gilt für  $P'$ :

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

$$\vec{C}' = \begin{pmatrix} 1 & 0 & 0 & -C'_1 \\ 0 & 1 & 0 & -C'_2 \\ 0 & 0 & 1 & -C'_3 \end{pmatrix} \quad (4.6)$$

$$T' = R'[I] - C \quad (4.7)$$

$$T' = \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -C'_1 \\ 0 & 1 & 0 & -C'_2 \\ 0 & 0 & 1 & -C'_3 \end{pmatrix} \quad (4.8)$$

$$T' = \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) & -C'_1 \cos(\alpha) + C'_3 \sin(\alpha) \\ 0 & 1 & 0 & -C'_2 \\ \sin(\alpha) & 0 & \cos(\alpha) & -C'_1 \sin(\alpha) - C'_3 \cos(\alpha) \end{pmatrix} \quad (4.9)$$

Der Quader hat insgesamt acht Punkte, welche auf die Bildebenen der Kameras projiziert werden. Neben den Punkten des Quaders, wird noch ein weiterer Punkt außerhalb des Quaders platziert und ebenfalls auf die Bildebenen projiziert. Mit insgesamt neun Punkten bei der Bestimmung der Fundamentalmatrix, wird die Wahrscheinlichkeit ein unterbestimmtes System aus der Koeffizientenmatrix  $A$  zu bekommen minimiert. Ist  $A$  unterbestimmt so besitzt sie ein Rang 7 und  $F$  kann nicht eindeutig durch einen Sieben-Punkte-Algorithmus bestimmt werden[3, 20]. In dem hier berechneten Beispiel werden deswegen neun Punkte benutzt um sicherzugehen, dass die  $A$  einen Rang 8 besitzt und somit eindeutig bestimmt werden kann. Zur Bestimmung von  $F$ , wird der in Kapitel 3 aufgeführte acht-Punkte Algorithmus angewandt.

Um die neun Punkte auf die Bildebenen  $(I, \tau)$  und  $(I', \tau')$  zu projizieren, müssen neben den Transformationsmatrizen  $R$  und  $R'$  noch die Kameramatrizen  $K_0$  und  $K'_0$  festgelegt werden.

$$K_0 = \begin{bmatrix} \zeta_C & 0 & 0 \\ 0 & \zeta_C & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.10)$$

$$K'_0 = \begin{bmatrix} \zeta_{C'} & 0 & 0 \\ 0 & \zeta_{C'} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (4.11)$$

Da zunächst von gleichen Kameraauflösungen ausgegangen wird, gilt  $\zeta = \zeta'$ . Sind  $R, R', K_0$  und  $K'_0$  bekannt, können die Projektionsmatrizen  $P$  und  $P'$  gebildet werden und anschließend werden die Punkte mit  $P$  und  $P'$  auf die Bildebenen projiziert.

$$P = K_0 \cdot R \quad (4.12)$$

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

$$P' = K'_0 \cdot R' \quad (4.14)$$

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

Durch die Projektionsmatrix können die Punkte des Quaders auf die Bildebenen projiziert werden. Die entstandenen Bilder sind in Abbildung 4.3 zu sehen.

## 4.2 Bildanalyse

Der Szenenrekonstruktionsalgorithmus für das synthetische Beispiel ist in drei Abschnitte unterteilt. Zuerst wird aus den Punktekorrespondenzen die Fundamentalmatrix und die essentielle Matrix geschätzt. Mit Hilfe der essentiellen Matrix werden die extrinsischen Kameraparameter bestimmt um so im letzten Schritt die Szenenpunkte durch Rückprojektion der Bildpunkte mit Hilfe der Kameraparameter rekonstruiert.

### 4.2.1 Bestimmung der extrinsischen Kameraparameter

Zur Bestimmung der extrinsischen Kameraparameter werden in dem hier berechneten Beispiel neun korrespondierende Punkte bestimmt. Mittels des *Epipolar-Constraint* aus Gleichung 3.18 wird der 8-Punkt-Algorithmus wie in Kapitel 3.3 beschrieben angewandt, um die Fundamentalmatrix zu bestimmen.

$$0 = \vec{m}'_\tau^T K'^{-T} R' [\vec{C}'_\delta - \vec{C}_\delta]_\times R^T K^{-1} \vec{m}_\tau \quad (4.16)$$

$$= \vec{m}'_\tau F \vec{m}_\tau \quad (4.17)$$

Wie in Kapitel 3.2 hergeleitet, bildet sich die essentielle Matrix aus:

$$\vec{m}'_\tau^T K'^{-T} R' [\vec{C}'_\delta - \vec{C}_\delta]_\times R^T K^{-1} \vec{m}_\tau = 0 \quad (4.18)$$

$$\rightsquigarrow E = R' [\vec{C}'_\delta - \vec{C}_\delta]_\times R^T \quad (4.19)$$

Um also  $E$  aus  $F$  zu bestimmen, gilt:

$$E = K'^T F K \quad (4.20)$$

$$E = K'^T (K'^{-T} R' [\vec{C}'_\delta - \vec{C}_\delta]_\times R^T K^{-1}) K \quad (4.21)$$

$$E = R' [\vec{C}'_\delta - \vec{C}_\delta]_\times R^T \quad (4.22)$$

Es wird davon ausgegangen, dass für  $T = [R] - RC$  von  $C$  gilt, dass  $T = [I] - 0$  ist. Die aus  $E$  zu ermittelnde Matrix  $T'$  beschreibt dann die Transformation von  $C'$  relativ zu  $C[3, 4]$ . Somit kann  $E$  umformuliert werden zu:

$$E = R' [\vec{C}'_\delta - \vec{C}_\delta]_\times R^T \quad (4.23)$$

$$E = R' [\vec{C}'_\delta - 0]_\times I^T \quad (4.24)$$

$$E = R' [\vec{C}'_\delta]_\times \quad (4.25)$$

Um  $R'$  und  $[\vec{C}'_\delta]_\times$  zu bestimmen, wird zunächst die essentielle Matrix  $E$ , mit Hilfe der Singulärwertszerlegung, in drei Matrizen zerlegt.

$$E = U \Sigma V^T \quad (4.26)$$

Die Singulärwerte  $\text{diag}(\sigma_1, \sigma_2, \sigma_3)$  der Matrix  $\Sigma$  müssen die Bedingung erfüllen, dass  $\Sigma = \text{diag}(1, 1, 0)[3, 4]$ . Wenn diese Bedingung nicht erfüllt ist, so wird sie erzwungen. Dazu wird Matrix  $\Sigma$  aus der Singulärwertszerlegung aus  $E$  modifiziert[3, 4].

$$E' = U \text{diag}(1, 1, 0) V^T \quad (4.27)$$

$[C'_\delta]_\times$  ist schiefsymmetrisch und kann in  $UZU^T$  zerlegt werden, wobei  $U$  eine orthogonale Matrix ist und  $Z$  eine block-diagonale Matrix[3].  $R'$  wird in  $UWV^T$  und  $UW^TV^T$  zerlegt, wobei  $W$  eine schiefsymmetrische Matrix ist[4, 3, 15].

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (4.28)$$

Somit lassen sich die folgenden Lösungsmöglichkeiten für  $[C'_\delta]_\times$  und  $R_1$  und  $R_2$  aufstellen[3, 4].

$$[C'_\delta]_\times = \pm UZU^T \quad R'_1 = UW^TV^T \quad (4.29)$$

$$S_2 = UZU^T \quad R'_2 = UWV^T \quad (4.30)$$

$[C'_\delta]_\times$  ist eine schiefsymmetrische Matrix, welche die Information für den noch gesuchten Translationsanteil  $v = -R'C'$  beinhaltet. Ohne zusätzliche Informationen kann  $v$ , nur bis zu einer Skaleninvarianz genau bestimmt werden[3, 4, 15]. Durch die Modulation der Singulärwerte von  $E$  gilt für  $\|v\| = 1$ [3, 4]. Um  $v$  aus  $[C'_\delta]_\times$  zu extrahieren, wird der Kern von  $[C'_\delta]_\times$  bestimmt

$$[C'_\delta]_\times \cdot v = v \times v = 0 \quad (4.31)$$

Die Skaleninvarianz bewirkt, dass es bei der Rekonstruktion die Größe der Objekte von ihrer Originalgröße abweichen. Die Abbildungen 4.11, 4.12 und 4.13, zeigen die Auswirkungen von Skaleninvarianz auf die später rekonstruierte Szene.

Letztendlich können, für die Rekonstruktion der extrinsischen Kameraparameter vier mögliche Lösungen für  $T$  in Form von  $T = R[I] - C$ , wie in Gleichung 2.13 in Kapitel 2 definiert, gefunden werden[3, 4, 15].  $\lambda v$  heißt dabei, dass sowohl  $v$  also auch alle Vielfache von  $v$ , Lösungen sein können, was durch die Skaleninvarianz der Resultate bedingt ist[3, 4, 15].

$$T' = [UWV^T] + \lambda v \quad \text{oder} \quad [UW^TV^T] + \lambda v \quad (4.32)$$

$$\text{oder} \quad [UWV^T] - \lambda v \quad \text{oder} \quad [UW^TV^T] - \lambda v \quad (4.33)$$

Die Abbildungen 4.5, 4.6, 4.7 und 4.8 stellen schematisch die vier verschiedenen Transformationsmöglichkeiten von  $T'$  dar.

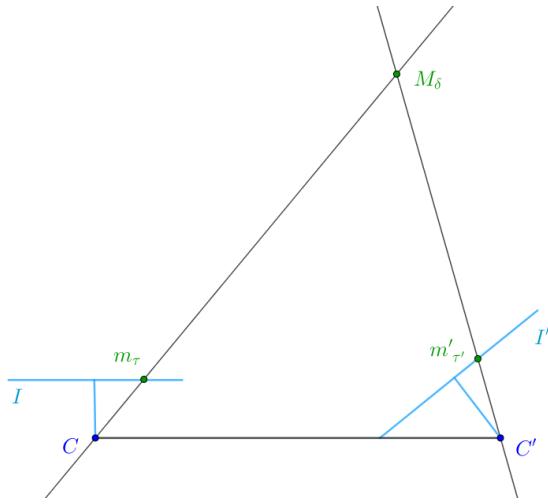


Abbildung 4.5: a)

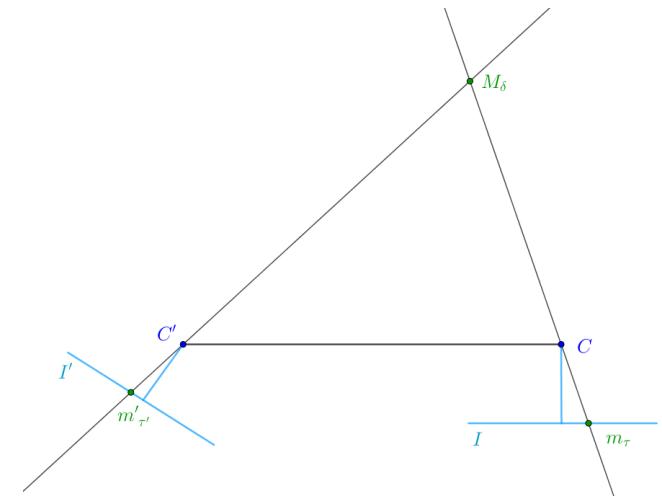


Abbildung 4.6: b)

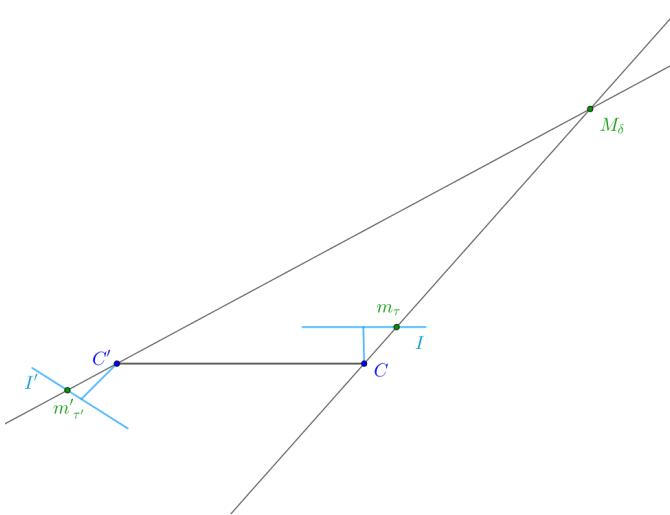


Abbildung 4.7: c)

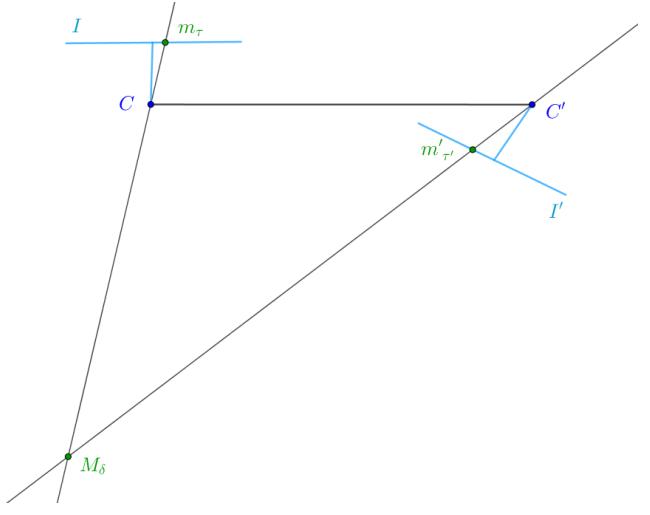


Abbildung 4.8: d)

Abbildung 4.9: Die Abbildungen a, b, c und d veranschaulichen, welche Bilder aus den vier Lösungen entstehen. In den Abbildungen a und b kommt es zu einer Umkehrung der Basisline. In den Abbildungen c und d wird  $C'$  um  $180^\circ$  gedreht

#### 4.2.2 Szenenrekonstruktion durch Triangulation

Als Triangulierung wird in der Computer Vision die Bestimmung eines 3D-Objektpunktes aus korrespondierenden Bildpunkten genannt. Als Voraussetzung für die Rekonstruktion müssen die jeweiligen korrespondierenden Bildpunkte und die Kameraparameter der einzelnen Kameras bekannt sein. Die Triangulierung funktioniert wie eine umgekehrte Projektion der Bildpunkte auf der Bildebene in einen Objektpunktes im Raum. Zwei Geraden, welche jeweils durch die Projektionszentren und den zu rekonstruierenden Bildpunkten gehen, treffen sich im Raum. Der Schnittpunkt beider Geraden, bildet den zu den Bildpunkten gehörenden Ursprungspunkt, wie in Abbildung 4.10 schematisch dargestellt ist.

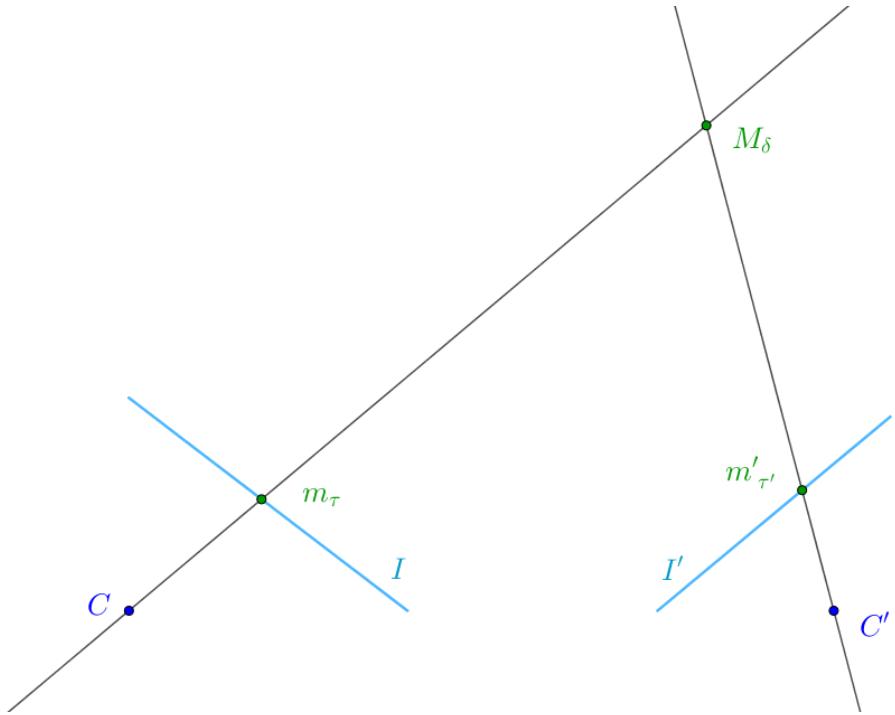


Abbildung 4.10: Optimale Triangulierung: Beide Geraden Treffen sich in einem Punkt im 3D-Raum

Im synthetischen Beispiel wird mit reinen Daten gearbeitet. Das heißt, die Bildpunkte wurden mathematisch von ihrem Ursprungspunkt im Raum berechnet und sind somit frei von Verfälschungen durch äußere Einflüsse. Ein zum Bildpunkt  $m_\tau$  korrespondierender Bildpunkt  $m'_{\tau'}$  liegt genau auf der zu  $m_\tau$  korrespondierenden Epipolarlinie  $l'$ . Somit ist garantiert, dass sich die Bildpunkte  $m_\tau$  und  $m'_{\tau'}$  bei einer Rückprojektion in einem Punkt  $M_{0,\delta}$  im Raum treffen. Durch die zuvor erwähnte Skaleninvarianz der extrinsischen Kameraparameter, handelt es sich bei  $M_{0,\delta}$  jedoch noch nicht um den eigentlichen Ursprungspunkt  $M_\delta$ . Um die Rekonstruktion im synthetischen Beispiel so nah wie möglich an einem realen Beispiel zu halten, wird davon ausgegangen, dass nur die Kameraparameter und die jeweiligen Bildkoordinaten bekannt sind.

Vor der Bestimmung der extrinsischen Kameraparameter wurde festgesetzt, dass  $T = [I| - 0]$  die Translationsmatrix von  $C$  ist. Somit gilt für die Projektionsmatrix von  $C$ , dass  $P = K_0T = K_0[I| - 0]$  ist. Die Projektionsmatrix  $P'$  für  $C'$  setzt sich aus einer der Lösungen von  $T'$  und der Kameramatrix  $K'_0$  zusammen, so dass gilt  $P' = K'_0T' = K[R'| - RC]$ .

Um eine Gerade von den Projektionszentren  $C$  und  $C'$  durch die jeweiligen Bildpunkte bilden zu können, müssen die Positionen von  $C$  und  $C'$  bekannt sein. Da die Koordinatensysteme  $(C, \beta)$  und  $(O, \delta)$  deckungsgleich sind, und  $P = K_0[I| - 0]$  ist, gilt  $C = (0, 0, 0)^T$ . Um  $C'$  aus  $T' = R'[R'| - R'C']$  zu bestimmen wird der Translationsvektor  $-R'C'$  aus  $T'$  mit dem Transponierten Rotationsmatrix  $R'^T$  aus  $T'$  multipliziert.

$$C' = R'^T \cdot -R'C' \quad (4.34)$$

Die zweidimensionalen Bildpunkte werden mit den bekannten Brennweiten  $\zeta$  und  $\zeta'$  aus  $K_0$  und  $K'_0$  zu einer dreidimensionalen Koordinate erweitert.

$$\begin{pmatrix} m_{\tau x} \\ m_{\tau y} \end{pmatrix} \rightsquigarrow \begin{pmatrix} m_{\tau x} \\ m_{\tau y} \\ \zeta \end{pmatrix} \quad (4.35)$$

$$\begin{pmatrix} m'_{\tau' x} \\ m'_{\tau' y} \end{pmatrix} \rightsquigarrow \begin{pmatrix} m'_{\tau' x} \\ m'_{\tau' y} \\ \zeta' \end{pmatrix} \quad (4.36)$$

Danach werden für die Rückprojektion zwei Geradengleichungen aufgestellt. Eine Gerade geht durch  $C$  und  $\begin{pmatrix} m_{\tau x} \\ m_{\tau y} \\ \zeta \end{pmatrix}$  die zweite Gerade geht durch die Punkte  $C'$  und  $\begin{pmatrix} m'_{\tau' x} \\ m'_{\tau' y} \\ \zeta' \end{pmatrix}$ . Danach wird aus den zwei Geraden der Schnittpunkt  $M_{\delta,0}$  im Raum bestimmt.  $C$  und  $C'$  sind aus Sicht des Weltkoordinatensystems  $(O, \delta)$  definiert.  $m'_{\tau'}$  wird noch in Koordinaten bezüglich des Kamerakoordinatesystem  $(C, \beta)$  transformiert mit  $m'_{\beta} = [R[I] - C']^{-1} \cdot m'_{\tau'}$ .

$$g := \vec{C} + t \cdot \left( \vec{C} - \begin{pmatrix} m_{\beta x} \\ m_{\beta y} \\ \zeta \end{pmatrix} \right) \quad (4.37)$$

$$g' := \vec{C}' + t \cdot \left( \vec{C}' - \begin{pmatrix} m'_{\beta x} \\ m'_{\beta y} \\ \zeta' \end{pmatrix} \right) \quad (4.38)$$

Bei den zuvor ermittelten extrinsischen Kameraparametern, ist der Translationsvektor skaleninvariant. Dies führt dazu, dass der rekonstruierte Objektpunkt  $M_{\delta,0}$  nach der Szenenrekonstruktion noch nicht dem Ursprünglichen  $M_{\delta}$  entsprechen muss. Dementsprechend wird als letzter Schritt die rekonstruierte Szene anhand einer bekannten Referenzgröße skaliert. Als Referenzgröße kann beispielsweise ein zuvor abgemessener Abstand zwischen zwei Punkten in der Szene dienen. Im synthetischen Aufbau sind beispielsweise die Abstände zwischen den Originalbildpunkten bekannt. Die Abbildungen 4.11, 4.12 und 4.13 zeigen die Szene des Quader mit unterschiedlichen Skalierungen. Die Abbildungen 4.14 und 4.15 zeigen die rekonstruierte Szene des synthetischen Beispiels.

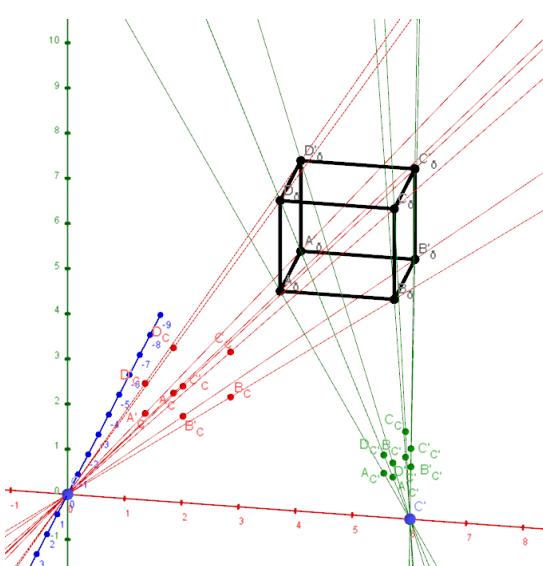


Abbildung 4.11

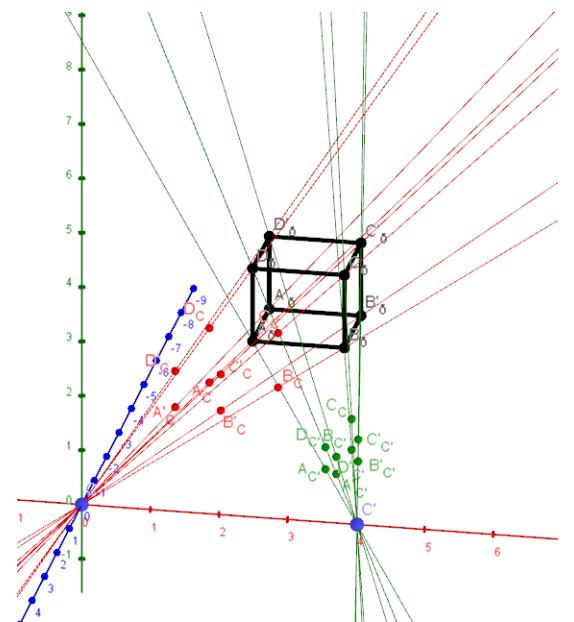


Abbildung 4.12

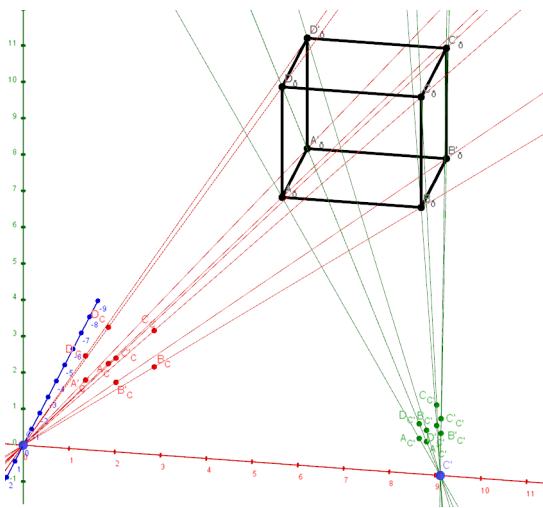


Abbildung 4.13: Veranschaulichung der Skaleninvarianz und dessen Auswirkung auf die geometrische Form und Größe der Objekte

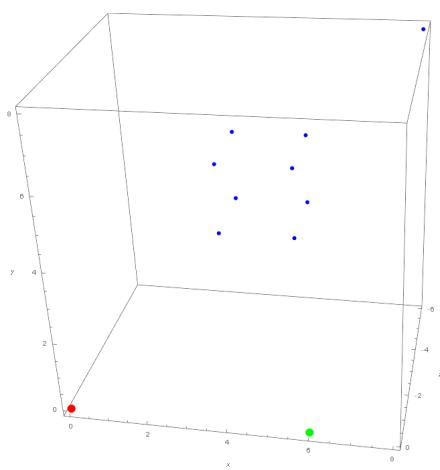


Abbildung 4.14

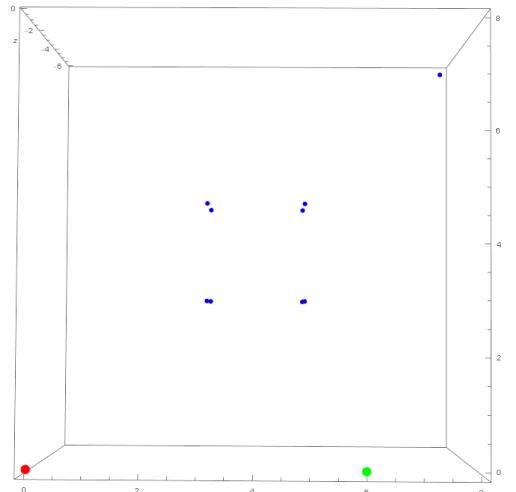


Abbildung 4.15

Abbildung 4.16: Der rote Punkt stellt Die Postion von  $C$  dar, der grüne steht für die Position von  $C'$  relativ zu  $C$ . Die blauen Punkte stellen den rekonstruierten Quader und den extern platzierten neunten Punkt da. Das Abbild entstand aus dem in *Mathematica* implementierten Algorithmus.

# 5 Auswirkungen von unterschiedlichen Kameraauflösungen

Aufbauend auf dem entstandenen Szenenrekonstruktionsalgorithmus für Kameras mit gleicher Auflösung, wurde der Algorithmus d.hingehend getestet, ob eine Szenenrekonstruktion auch möglich ist, wenn gilt, dass  $\zeta \neq \zeta'$ . Zunächst wird beschrieben, was genau bei der Umstellung der Auflösung auf einem Sensor sich verändert. Anschließend wird analysiert, wie sich die Auflösungsänderung auf das in Kapitel 2 beschriebene Kameramodell ändert und welche Einflüsse sie auf die in Kapitel 3 hergeleiteten Epipolaren Bedingungen hat. Zum Schluss werden die Ergebnisse des synthetischen Beispiels mit unterschiedlichen Kameraauflösungen präsentiert und validiert.

## 5.1 Geometrie eines Sensors

Die Geometrie eines Sensors kann als eine  $M \times N$ -Matrix, bestehend aus  $M \times N$  Sensorelementen dargestellt werden[8]. Die Auflösung eines Sensors hängt von den horizontalen und vertikalen Abständen der Sensorelemente ab. Abbildung 5.1 zeigt den schematischen Aufbau eines Sensors (CMOS).

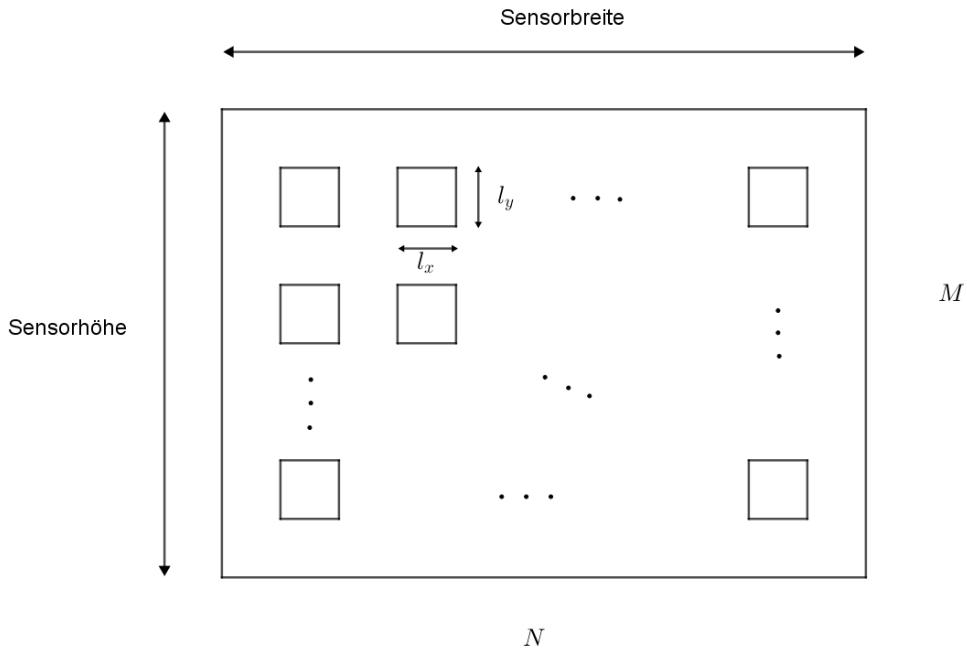


Abbildung 5.1: Rechteckiger Bildsensor mit darauf sich befindenden quadratischen Sensorelementen. Vergleiche [8]

Ein Sensor hat eine maximale Auflösung. Bei maximaler Auflösung definiert genau ein Sensorelement einen Pixel. Ein Pixel wiederum entspricht einem Bildpunkt[8]. Die Bildqualität ist abhängig von der Größe des Sensorchips und der Menge der sich darauf befindenden Sensorelemente. Wird eine Auflösung kleiner der maximal möglichen Auflösung eingestellt, desto geringer wird die Anzahl der Pixel. Der Prozess, welcher hier stattfindet, gehört zu den Nachbarschaftsoperationen[8]. Ein neuer Pixel wird aus den benachbarten Pixeln berechnet. Dabei nehmen alle Pixel, die sich zum neuen Pixel zusammengeschlossen haben, die gleichen Farbwerte an.

Eine Veränderung der Auflösung auch eine Änderung der Seitenverhältnisse mit einschließen. Ändert sich das Seitenverhältnis so wird der Bereich der lichtempfindlichen Fläche auf dem Sensor beschränkt[8]. Abbildung 5.2 stellt schematisch da wie sich die lichtempfindlichen Bereiche auf dem Sensor bei unterschiedlichen Auflösungen ändert.

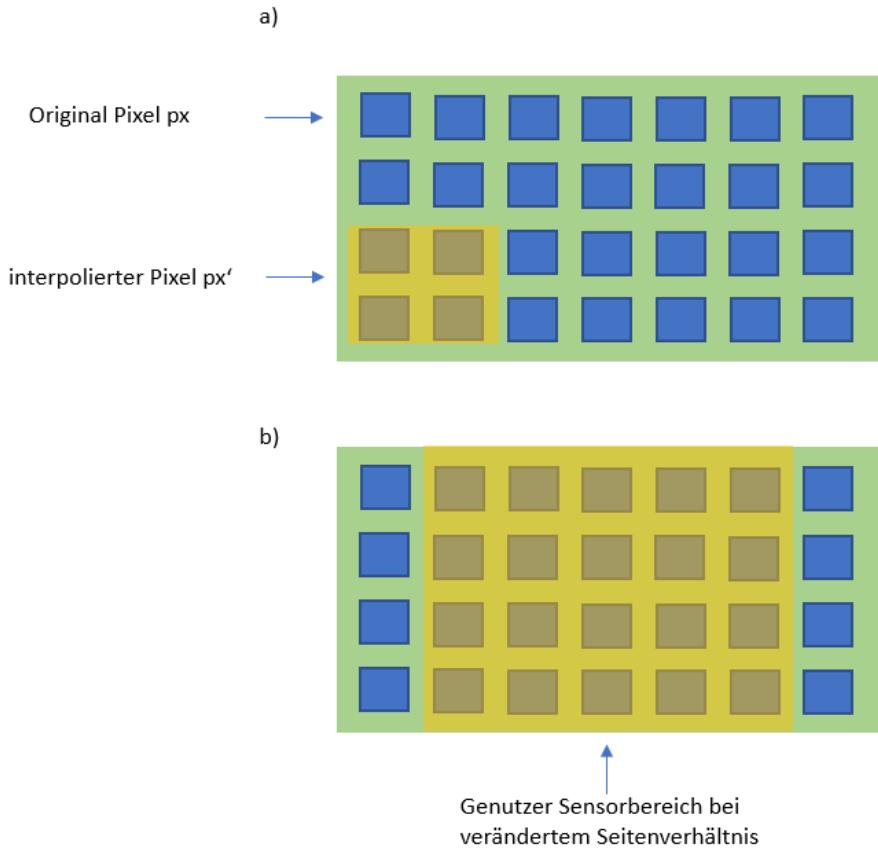


Abbildung 5.2: Bild a) zeigt die den Zusammenschluss mehrerer benachbarter Pixel zu einem neuen Pixel. Bild b) zeigt in gelb markiert, den aktiven lichtempfindlichen Bereich des Sensors, wenn sich das Seitenverhältnis geändert wird und nicht mehr der komplette Sensor genutzt wird.

## 5.2 Auswirkungen auf die Szenenrekonstruktion

Im folgenden soll analysiert werden, welche Änderungen sich im Lochkameramodell bei veränderte Auflösung ergeben und was für Auswirkungen diese auf den, *Epipolar-Constraints* haben, gerade in Bezug auf die Bestimmung der extrinsischen Kameraparameter.

Im Lochkameramodell verändert sich mit ändernder Auflösung der Kamera lediglich die Skalierung der Sensorkoordinatenachsen. Diese orientieren sich wie in Kapitel 2 beschrieben an längenskalierung der Pixelkanten  $l_x$  und  $l_y$ . Wird die Auflösung einer Kamera verändert, so ändern sich die Anzahl und die Größe der Pixel. Folglich kommt es zu einer Skalierung des Sensorkoordinatensystems. Für alle anderen Koordinatensysteme ändert sich nichts. Während sich für dein Punkt im Sensorkoordinatensystem neue skalierte Koordinaten ergeben, so bleibt der Bildpunkt in Bildebenenkoordinaten unverändert. Durch Nachbarschaftsoperationen werden aus mehreren benachbarten Pixel ein neuer, jedoch bleibt der Ort des Pixel der gleiche[21]. Die Abbildungen 5.3 und 5.4 zeigen, dass sich zwar das Mapping von Bildebenenkoordinatensystem auf das Sensorsystem ändert, Jedoch wird der Bildpunkt  $m_{\tau'}$  an der selben Position des Sensors wie zuvor auch abgebildet.

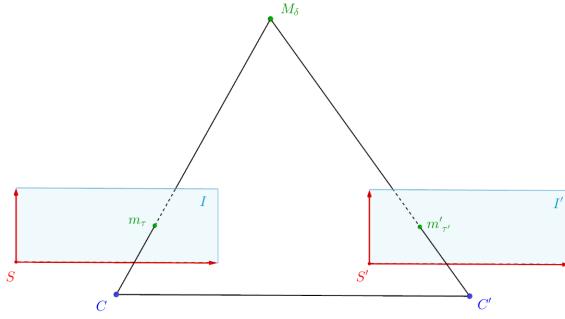


Abbildung 5.3:  $C$  und  $C'$  haben die selbe Auflösung eingestellt

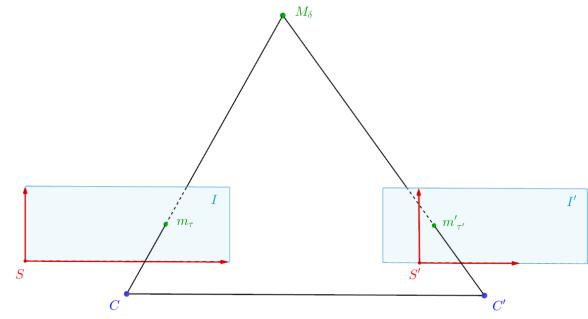


Abbildung 5.4:  $C$  und  $C'$  haben unterschiedliche Auflösungen eingestellt

Eine Skalierung der Sensorkoordinaten führt eine Skalierung der Kameramatrix mit sich. Wie in Kapitel 2 gezeigt, setzt sich die erweiterte Kameramatrix  $K$  wie folgt zusammen:

$$K = R_\sigma K_0 = \begin{bmatrix} k_x \zeta & 0 & V_{\sigma,x} \\ 0 & k_y \zeta & V_{\sigma,y} \\ 0 & 0 & 1 \end{bmatrix} \quad (5.1)$$

Wird die Kameraauflösung verändert so bleibt die Brennweite  $\zeta_x$  und  $\zeta_y$  effektiv die gleiche, da sich aber die Skalierung der Pixel mit  $k_x$  und  $k_y$  ändert, ist die Brennweite in Pixeln ausgedrückt verändert. Im synthetischen Beispiel wurde die Auflösung von  $K'$  verändert indem, die Werte  $\zeta'_x$  und  $\zeta'_y$  durch Multiplikation mit einem Skalierungsfaktor verändert wurden.

Um die aufgestellte Theorie zu überprüfen, wurde im synthetischen Beispiel die Kameramatrix von einer der beiden Kameras modifiziert. Für  $C$  wurde  $\zeta_x = \zeta_y = \zeta$  definiert, so dass für Kameramatrix  $K$  gilt:

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

Für  $C'$  wurden drei verschiedene Kameramatrizen  $K'$  definiert. Die resultierenden Abbildungen des Quaders sind in den Abbildungen 5.5, 5.6, 5.7 und 5.8 zu sehen.

Während für die Kameramatrix von  $C$  der Wert  $\zeta_x = \zeta_y = 1$  in der Kameramatrix  $K$  ist, wird das  $\zeta$  in  $C'$  verändert, so dass drei verschiedene neue Kameramatrizen  $K'_1, K'_2$  und  $K'_3$  ergeben.

$$K = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.3)$$

$$K'_1 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.4)$$

$$K'_2 = \begin{bmatrix} 3.2 & 0 & 0 & 0 \\ 0 & 1.2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.5)$$

$$K'_3 = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 4.3 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5.6)$$

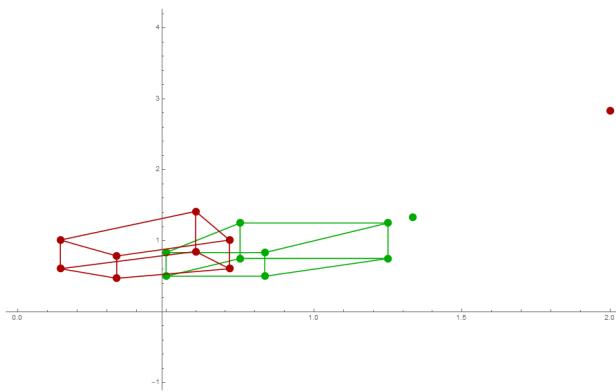


Abbildung 5.5:  $C$  und  $C'$  haben die selbe Auflösung eingestellt

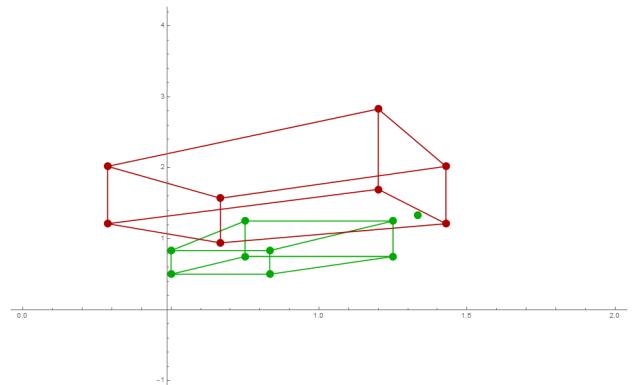


Abbildung 5.6:  $C$  und  $C'$  haben unterschiedliche Auflösungen eingestellt.  $C$  mit  $K$  und  $C'$  mit  $K'_1$

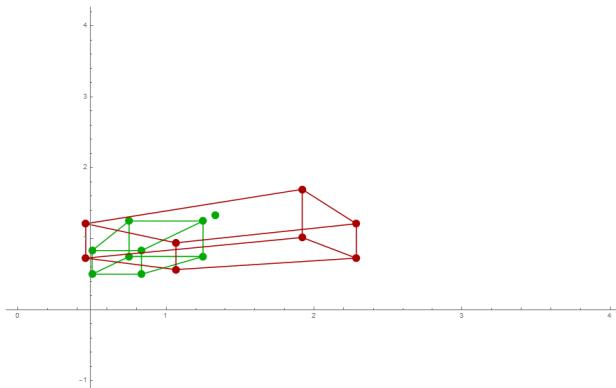


Abbildung 5.7:  $C$  mit  $K$  und  $C'$  mit  $K'_2$

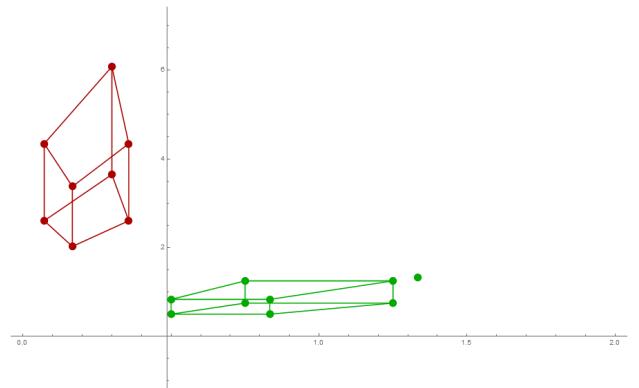


Abbildung 5.8:  $C$  mit  $K$  und  $C'$  mit  $K'_3$

Berchnung Für F und E mit unterschiedlichen Auflösungen, was verändert sich für F ? E erklären, dass dadurch das K auf die Punkte berechnet wird E unverändert bleibt

Jetzt noch irgendwie drauf verweisen dass bei der herleitung eh von K und K' ausgegangen wird also zwei unterschiedlichen Kameramatrizen....

Für die Fundamentalmatrix und die essentielle Matrix ergeben sich verglichen mit denen aus dem ersten Beispiel mit Kameras gleicher Abbildung folgende Matrizen.

Für die Fundamentalmatrix und die essentielle Matrix ergeben sich verglichen mit denen aus dem ersten Beispiel mit Kameras gleicher Abbildung folgende Matrizen.

$$\begin{aligned}
\zeta_x = 1, \zeta_y = 1 : \quad F &= \begin{pmatrix} 0 & -0.5 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & -0.5 & 0 \end{pmatrix} | : -0.5 \rightsquigarrow F = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix} \\
\zeta_x = 2, \zeta_y = 2 : \quad F &= \begin{pmatrix} 0 & 0.378 & 0 \\ 0 & 0 & -0.534 \\ 0 & 0.756 & 0 \end{pmatrix} | : 0.756 \rightsquigarrow F = \begin{pmatrix} 0 & 0.5 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \end{pmatrix} \\
&\quad \frac{1}{\zeta_x} = 0.5, \quad \frac{\sqrt{2}}{\zeta_y} = \frac{1}{\sqrt{2}} \\
\zeta_x = 3.2, \zeta_y = 1.2 : \quad F &= \begin{pmatrix} 0 & 0.198 & 0 \\ 0 & 0 & -0.747 \\ 0 & 0.634 & 0 \end{pmatrix} | : 0.634 \rightsquigarrow F = \begin{pmatrix} 0 & 0.312 & 0 \\ 0 & 0 & -1.178 \\ 0 & 1 & 0 \end{pmatrix} \\
&\quad \frac{1}{\zeta_x} = 0.312, \quad \frac{\sqrt{2}}{\zeta_y} = -1.178 \\
\zeta_x = 0.5, \zeta_y = 4.3 : \quad F &= \begin{pmatrix} 0 & 0.885 & 0 \\ 0 & 0 & -0.145 \\ 0 & 0.442 & 0 \end{pmatrix} | : 0.442 \rightsquigarrow F = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & -0.328 \\ 0 & 1 & 0 \end{pmatrix} \\
&\quad \frac{1}{\zeta_x} = 2, \quad \frac{\sqrt{2}}{\zeta_y} = -0.328
\end{aligned}$$

Ich weiß nicht wie ich das formuliere was man hier beobachtet....

$$\begin{aligned}
\zeta_x = 1, \zeta_y = 1 : \quad E &= \begin{pmatrix} 0 & -0.5 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0.5 & 0 \end{pmatrix} | : 0.5 \rightsquigarrow E = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix} \\
\zeta_x = 2, \zeta_y = 2 : \quad E &= \begin{pmatrix} 0 & 0.756 & 0 \\ 0 & 0 & 1.069 \\ 0 & -0.756 & 0 \end{pmatrix} | : -0.756 \rightsquigarrow E = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix} \\
\zeta_x = 3.2, \zeta_y = 1.2 : \quad E &= \begin{pmatrix} 0 & 0.634 & 0 \\ 0 & 0 & 1.069 \\ 0 & -0.634 & 0 \end{pmatrix} | : -0.634 \rightsquigarrow E = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix} \\
\zeta_x = 0.5, \zeta_y = 4.3 : \quad E &= \begin{pmatrix} 0 & 0.442 & 0 \\ 0 & 0 & 1.069 \\ 0 & -0.442 & 0 \end{pmatrix} | : -0.442 \rightsquigarrow E = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}
\end{aligned}$$

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[3]. 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 =  $\begin{pmatrix} 3. & 3. & -4. \\ 5. & 3. & -4. \\ 5. & 5. & -4. \\ 3. & 5. & -4. \\ 3. & 3. & -6. \\ 5. & 3. & -6. \\ 5. & 5. & -6. \\ 3. & 5. & -6. \\ 8. & 8. & -6. \end{pmatrix}$

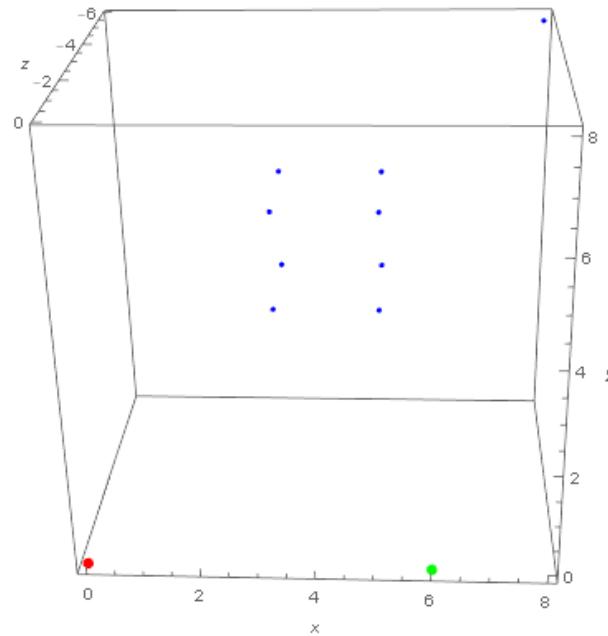


Abbildung 5.9: Die rekonstruierten Szenenpunkte und Kamerapositionen bleibt auch bei unterschiedlichen Auflösungen die selben

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 Reelle Rekonstruktion

Der entwickelte Szenenrekonstruktionsalgoritmus wird im folgenden anhand eines realen Stereoaufbau getestet. Da das Arbeiten mit realen Bilddaten bestimmte Fehleranfälligkeit aufweist, wird in diesem Kapitel Hauptsächlich drauf eingegangen, um welche Fehler es sich handelt, was ihre Auswirkungen sind und wie man ihnen entgegenwirken kann. Hierzu wurde der Ursprüngliche Algorithmus um bestimmte Funktionen erweitert, welche im Verlauf des Kapitels genau erläutert werden. Abbildung 6.1 zeigt den Ablauf für die Reelle Rekonstruktion.

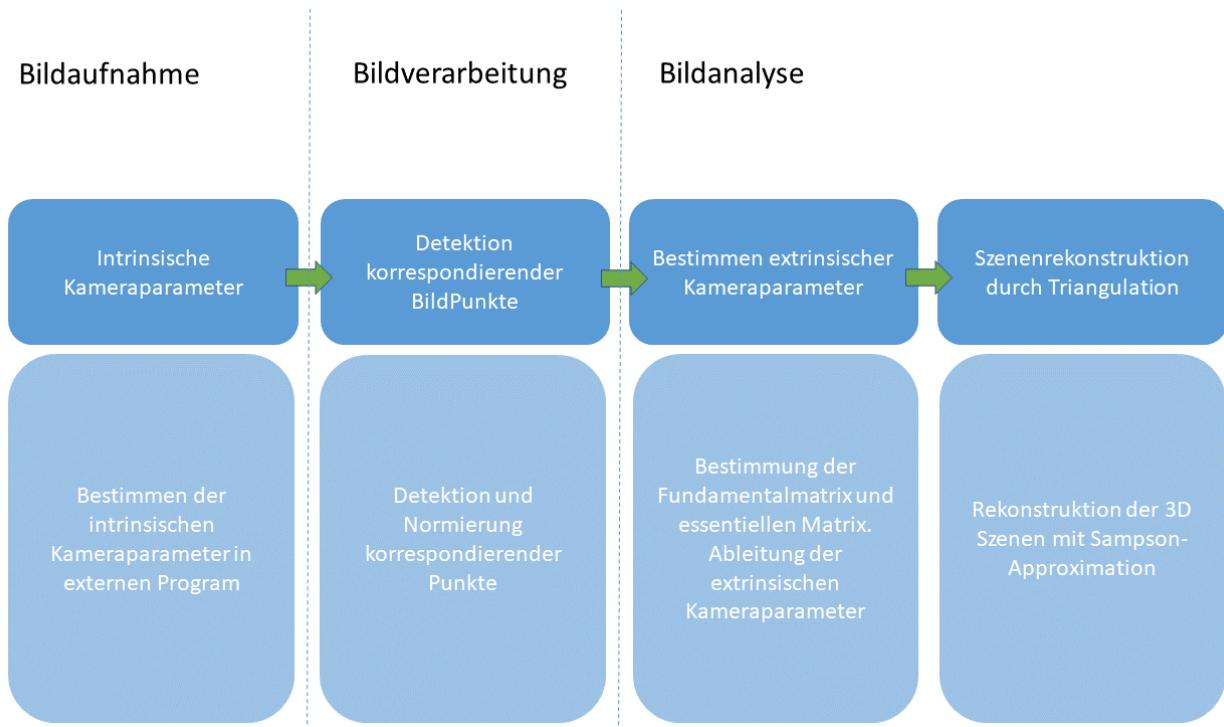


Abbildung 6.1: Ablaufdiagramm für die reelle Rekonstruktion

Zu aller erst wird der Stereoaufbau vorgestellt. Danach folgt die Korrespondenzanalyse, in welcher zwei Möglichkeiten aufgeführt werden, wie die Korrespondierenden Punkte aus den Bildern gewonnen werden. Der normierte-acht-Punkt-Algorithmus, stellt eine für reale Bilddaten leicht veränderte Fassung des bereits bekannten acht-Punkte-Algorithmus vor. Danach werden die Singulärwerte der aus realen Daten bestimmten Fundamentalmatrix und essentiellen Matrix analysiert und deren Auswirkung auf die Epipolargeometrie aufgezeigt. Als letztes wird das Triangulationsverfahren anhand abweichender Punktekorrespondenzen vorgeführt.

## 6.1 Stereoaufbau

Für die Stereobildaufnahme, wurde eine Szene vor zwei nebeneinander platzierten Kameras platziert. Beide Kameras wurden zur Szene hin rotiert. Abbildung 6.2 zeigt den Stereoaufbau.



Abbildung 6.2: Szenenaufbau: Die Canon 60D befindet sich in dieser Abbildung auf der linken Seite, die Canon 60 D auf der rechten. Auf dem Tisch zwischen den Kamera ist die in den Abbildungen 6.1 und 6.1 abgebildete Szene zu sehen. Beide Kamera sind zu Szene hin gedreht.

Die auf Abbildung 6.2 zu sehende linke Kamera wurde als Kamera eins definiert. Für Kamerakoordinatensystem  $(C, \beta)$  wurde gleich dem Weltkoordinatensystem  $(O, \delta)$  gesetzt. Kamera zwei mit  $(C', \beta')$  befindet sich auf der Abbildung rechts. Für beide Kamera wurde in einem externen Programm die intrinsischen Kameraparameter  $K$  und  $K'$  bestimmt. An den Stereoaufnahmen der Szene wurde dann eine Korrespondenzanalyse gemacht.

## 6.2 Korrespondenzanalyse

Für die Analyse der Korrespondierenden Punkte bei Stereoaufnahmen einer dreidimensionalen Szene wurde eine existierende Funktion von Mathematica genutzt[22]. Die Funktion basiert auf dem Prinzip eines SURF-Algoruthmus. SURF ist die Kurzform für *Speeded Up Robust Features* und ist ein Rotations- und Skaleninvariante Punkte Detektor und Deskriptor[17, 23]. Es werden Punkte an markanten Stellen in beiden Bildern detektiert, wie beispielsweise Eckpunkte oder Kanten. Die Umgebung eines jeden gefundenen Punktes wird durch einen Merkmalsvektor, dem Deskriptor, beschrieben. Die Deskriptoren beider Bilder werden abgeglichen und gleiche Punkte werden als korrespondierende Punkte gekennzeichnet[17, 23]. Die Abbildungen 6.4 und 6.3 zeigen die Ergebnisse nach der Anwendung des SURF-Algoruthmus auf das Stereobildpaar. Eine eigens implementierte alternative für die Korrespondenzanalyse zwischen Stereoaufnahmen eines zweidimensionalen Schachbretts wird in Kapitel 8 vorgestellt.

Die mit dem SURF-Algoruthmus gefundenen Punkte sind mit den gelben Ziffern im Bild gekennzeichnet. Abbildung a zeigt das Bild von  $C$ , die Abbildung b zeigt das Bild von  $C'$ .

Die Detektion von korrespondierenden Punkten mit Detektionsalgorithmen, wie Beispielsweise dem angewendeten SURF-Algoruthmus, können immer Fehler und Abweichungen mit sich bringen. Die Ursprünge der Fehler können sowohl durch den Algoruthmus als auch durch Fehler, wie Bildrauschen, in den Aufnahmen selbst. Diese Fehler wirken sich sowohl auf die bestimmung der Abbildungsvorschriften  $F$  und  $E$  aus und somit auch auf die Genauigkeit der Szenenrekonstruktion[3]. Im folgenden werden sowohl die Fehler als auch Methoden für deren Minimierung vorgestellt.

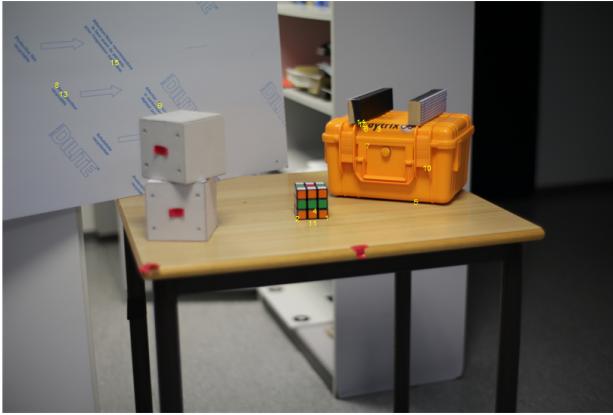


Abbildung 6.3: a

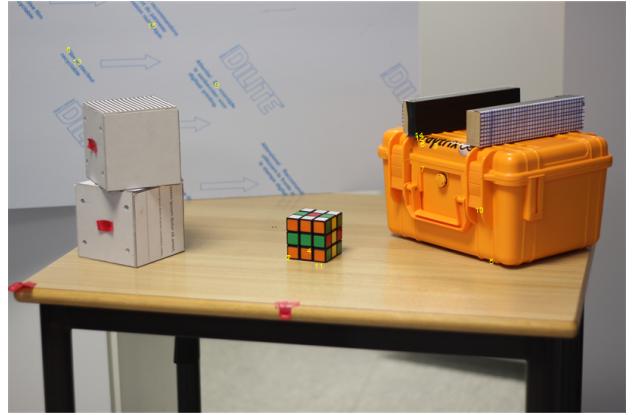


Abbildung 6.4: b

### 6.3 Normierter acht-Punkt-Algorithmus

Trotz das der acht-Punkt-Algorithmus eine einfache Methode zur Bestimmung der Fundamentalmatrix bietet, ist er sehr unstabil sobald Fehler wie Ungenauigkeiten in Punktekorrespondenzen oder Rauschen in Bilder auftreten[3, 24].

Der Fehler lässt sich anhand der Kondition der Koeffizientenmatrix  $A$  genauer beschreiben. Als Kondition wird die Abhängigkeit der Lösung eines Problems von der Störung der Eingangsdaten beschrieben[16, 25]. Die Kondition lässt sich durch Bestimmung des kleinsten Eigenvektors der Matrixmultiplikation der Koeffizientenmatrix  $A$  mit ihrer Transponierten  $A^T$  herausfinden. Die Matrix  $AA^T$  wird in die Matrizen  $UDU^T$ , wobei  $U$  eine orthogonale und  $D$  eine diagonale Matrix ist, zerlegt. Die Diagonaleinträge von  $D$  sind in eine nicht ansteigenden Reihenfolge, woraus resultiert, dass der kleinste Singulärwert von  $D$  mit der letzten Spalte von  $U$  korrespondiert und somit ist die letzte Spalte von  $U$  gleich dem kleinsten Eigenvektor von  $AA^T$ [16, 25]. Wird angenommen, dass  $AA^T$  eine  $9 \times 9$ -Matrix ist, so ergeben die Diagonaleinträge  $d_1/d_9$  den Wert der Kondition. Je größer die Kondition ist, desto größer wirken sich schon kleinste Abweichungen der einkommenden Bilddaten, auf die aus  $A$  bestimmten Matrix  $F$  aus.

Um die Kondition möglichst klein zu halten, werden die Bildkoordinaten beider Bilder normiert. Die in Literaturquellen, vorgeschlagene Normierung beinhaltet pro Bild eine Translation und Skalierung, so dass der Schwerpunkt aller Punktekorrespondenzen auf einem Bild im Ursprung des Sensorsystems liegt und der durchschnittliche Abstand der Punkte zum Ursprung  $\sqrt{2}$  beträgt[3, 4, 24].

Für die Normierung wird pro Bild eine Transformationsmatrix  $T$  und  $T'$  definiert. Die Matrizen beinhalten sowohl eine Skalierung als auch eine Translation. Die Bestimmung der Matrix  $T$  wird hier aufgezeigt. Zuerst wird der Schwerpunkt  $s$  mit  $s = \begin{pmatrix} s_x \\ s_y \end{pmatrix}$  der Punktemenge  $p_n$  mit  $p_n = \begin{pmatrix} p_{nx} \\ p_{ny} \end{pmatrix}$  berechnet, indem der Mittelwert aller Punkte  $p_n$  berechnet wird.

$$\begin{pmatrix} s_x \\ s_y \end{pmatrix} = \frac{1}{n} \sum_{i=1}^n \begin{pmatrix} p_{ix} \\ p_{iy} \end{pmatrix} \quad (6.1)$$

Danach wird  $s$  in den Ursprung verschoben. Die Punkte  $x_n$  werden um den Wert von  $s$  verschoben  $x'_n = x_n - s$ . Der Mittelwert aus den um  $s$  verschobenen Punkten  $x'_n$  ergibt den neuen Schwerpunkt  $s_0$  im Koordinatenursprung. Als nächstes wird die Distanz jedes Punktes von  $x'_n$  zu  $s_0$  berechnet und der Mittelwert aller Distanzen, hier mit  $d$  bezeichnet, berechnet. Die Matrix  $T$  und  $T'$  haben dann die folgende Form:

$$T = \begin{pmatrix} \frac{\sqrt{2}}{d} & 0 & -s_x \\ 0 & \frac{\sqrt{2}}{d} & -s_y \\ 0 & 0 & 1 \end{pmatrix} \quad (6.2)$$

$$T' = \begin{pmatrix} \frac{\sqrt{2}}{d'} & 0 & -s'_x \\ 0 & \frac{\sqrt{2}}{d'} & -s'_y \\ 0 & 0 & 1 \end{pmatrix} \quad (6.3)$$

Die originalen Bildpunkte des Stereobildpaars, werden mit den Matrizen  $T$  und  $T'$  verrechnet. Mit den Normierten Bildkoordinaten wird dann wieder nach dem in Kapitel 3.3 beschriebenen acht-Punkte-Algorithmus eine Fundamentalmatrix  $\hat{F}$  bestimmt[3, 16, 4, 24]. Nachdem  $\hat{F}$  aus den normierten Koordinaten bestimmt wurde, wird sie mit  $T$  und  $T'$  wieder denormalisiert.

$$F = T'^T \hat{F} T \quad (6.4)$$

### 6.3.1 Singularität der Fundamentalmatrix

Die Fundamentalmatrix ist eine singuläre-Matrix und ist somit eine Matrix von Rang zwei. Die Singularität der Fundamentalmatrix sorgt zum einen dafür das ihr rechter und linker Kern jeweils den Epipol des jeweiligen Bildes ergibt und die Epipolarlinien auch alle durch eben diese Epipole verlaufen[3]. wird die Fundamentalmatrix durch eine Singulärwertszerlegung von  $A$  geschätzt, ist die Chance sehr hoch, dass das Ergebnis für  $\hat{F}$  eine Matrix von Rang 3 ist. Sollte dies der Fall sein gehen die Epipolarlinien der Bilder nicht mehr durch genau einen Punkt, wie man in den Abbildungen 6.5 und 6.6 erkennen kann. Diese bilden Epipolarlinien in einem Stereobildpaar ab.

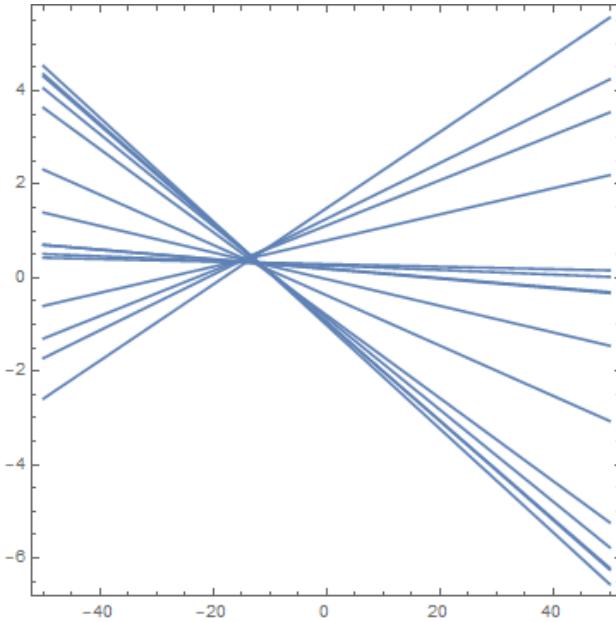


Abbildung 6.5: Epipolarlinien ohne *Epipolar-constraint* im Bild der Canon 6D

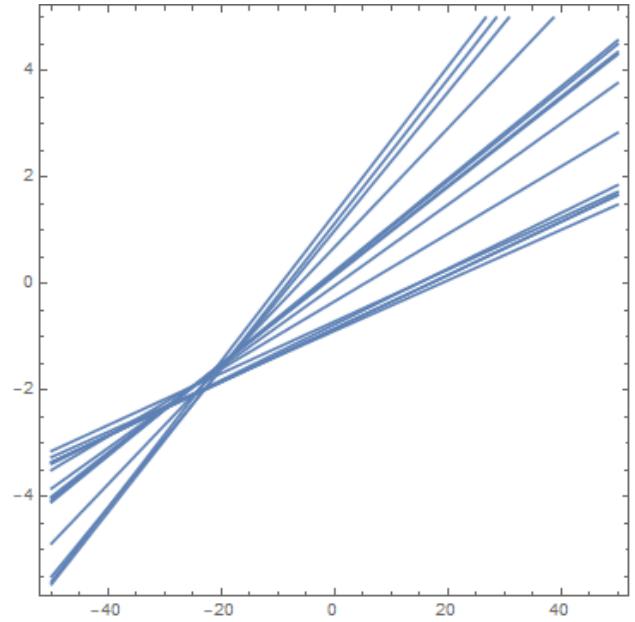


Abbildung 6.6: Epipolarlinien ohne *Epipolar-constraint* im Bild der Canon 60D

Um eine gültige Fundamentalmatrix für den weiteren Arbeitsprozess zu generieren, kommt hier ein sogenannter *singularity constraint* zum Einsatz. Dieser erzwingt in  $\hat{F}$  eine Singularität. Zu aller erst wird eine Singulärwertszerlegung an  $F$  durchgeführt, so dass  $\hat{F}$  in  $\hat{F} = UDV^T$  zerlegt wird.  $D$  beinhaltet in einer diagonalen Matrix die Singulärwerte  $D = \text{diag}(r, s, t)$ , welche die Bedingung  $r \leq s \leq t$

erfüllen. Um nun den *singularity-constraint* in  $\hat{F}$  zu erzwingen, wird der letzte Singulärwert  $t = 0$  gesetzt, so dass am Ende dasteht  $D = \text{diag}(r, s, 0)$ . Danach werden die Matrizen  $UDV^T$ , wobei  $D$  nun die modifizierten Singulärwerte beinhaltet, wieder zu  $\hat{F}$  multipliziert. Die jetzt resultierende Fundamentalmatrix  $\hat{F}$  besitzt den Rang 2. Der rechte und linke Kern ergeben wieder die Epipole und die Epipolarlinien verlaufen wieder durch eben diese Epipole. Die Abbildungen 6.7 und 6.8 zeigen die selben Epipolarlinien wie in 6.5 und 6.6 nachdem der *singularity-constraint* in  $\hat{F}$  erzwungen wurde. Die somit entstandene Matrix  $\hat{F}$ , ist die laut Frobenius norm, nächste zum ursprünglichen  $\hat{F}$ [3]. Jetzt erst erfolgt die zuvor erwähnte Denormierung von  $\hat{F}$  durch  $T$  und  $T'$ . Die Abbildungen 6.9 und 6.10 zeigen die Epipolarlinien im Originalbild mit denormalisierten Koordinaten.

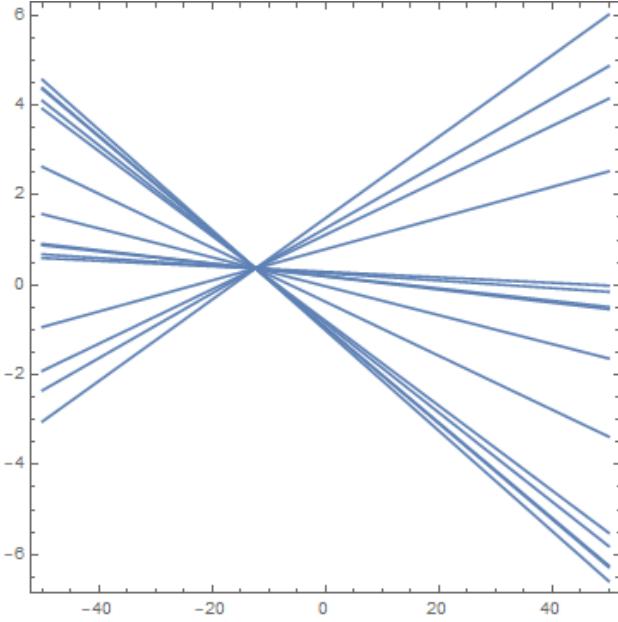


Abbildung 6.7: Epipolarlinien mit *Epipolar constraint* im Bild der Canon 6D

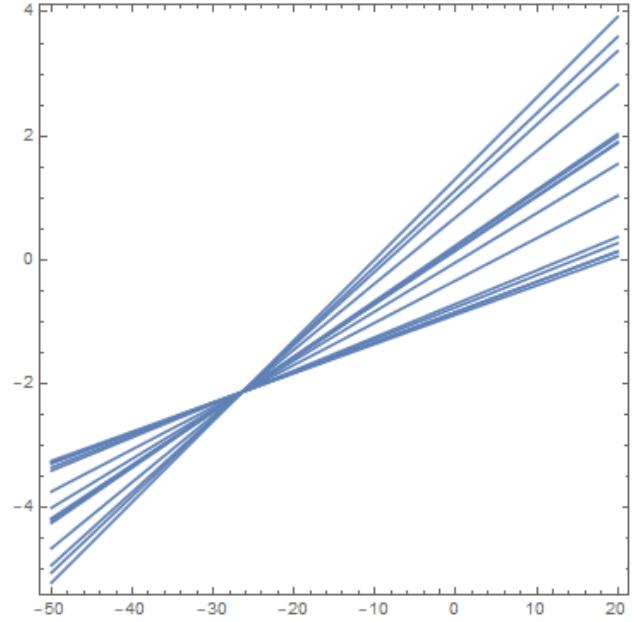


Abbildung 6.8: Epipolarlinien ohne *Epipolar constraint* im Bild der Canon 60D

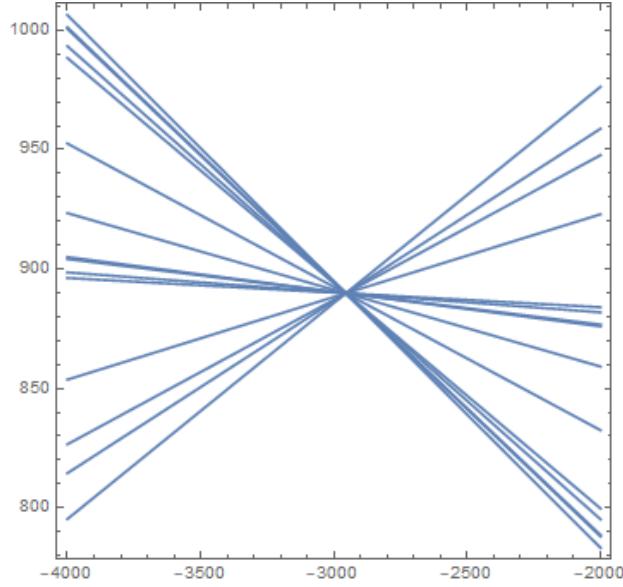


Abbildung 6.9: Epipolarlinien mit *Epipolar constraint* im Bild der Canon 6D, nach der denormalisierung von  $F$

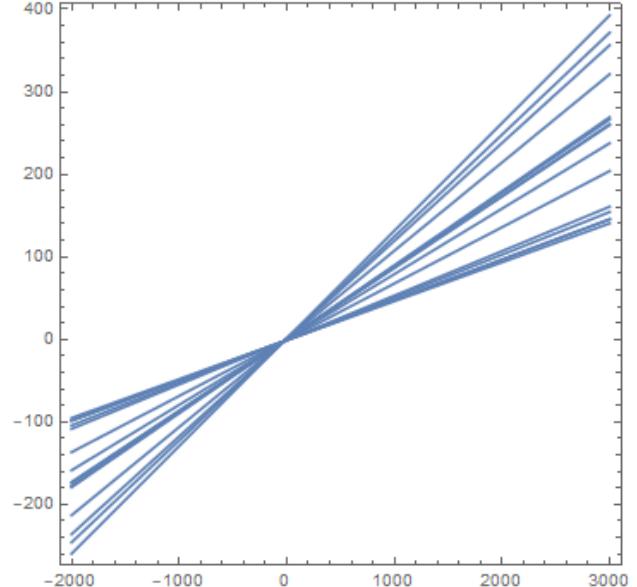


Abbildung 6.10: Epipolarlinien ohne *Epipolar constraint* im Bild der Canon 60D, nach der denormalisierung von  $F$

### 6.3.2 Singulärwerte der essentiellen Matrix

Die essentielle Matrix  $E$  kann wenn sie über den *eight-point-algorithm* ermittelt wird, auch eine Rang 3 Matrix anstelle einer Rang 2 Matrix sein. Eine essentielle Matrix wird darüber definiert, dass sie eine Matrix mit Rang 2 ist und ihre Singulärwerte in  $D$  von  $E = UDV^T$  die Eigenschaft besitzen, dass  $D = \text{diag}(a, b, c)$  mit  $a = b$  und  $c = 0$ . Sind die Singulärwerte nicht in der gezeigten Form vorhanden und  $E$  hat den Rang 3, so ist sie keine gültige essentielle Matrix[3]. Im implementierten Algorithmus, welcher in dieser Arbeit vorgestellt wird, wird die essentielle Matrix über die Fundamentalmatrix  $F$  und den intrinsischen Kameraparameter  $K$  und  $K'$  gewonnen. Da im vorherigen Schritt für die Matrix  $F$  schon der *singularity-constraint* erwirkt wurde, ist dadurch dass  $F$  nun eine Matrix von Ran 2 ist auch versichert, dass  $E$  ebenfalls von Rang 2 ist. Jedoch bedeutet das nicht gleichzeitig, dass auch die Bedingungen für die Singulärwerte von  $E$  erfüllt sind. Wird  $E$  in  $UDV^T$  zerlegt und die Singulärwerte in  $D$  haben beispielsweise die Form  $D = \text{diag}(a, b, c)$  mit  $a \geq b \geq c$ , so muss auch hier die für  $E$  typische Singularität erzwungen werden. Die laut Frobenuis Norm nächste Matrix  $E$  zur momentanen  $E$  kann durch modifizieren der Singulärwerte von  $D$  mit  $D = \text{diag}(\frac{a+b}{2}, \frac{a+b}{2}, 0)$  erzwungen werden[3]. Mit der neuen essentiellen Matrix können dann genau wie im Kapitel Synthetische Rekonstruktion auch wieder die vier möglichen Lösungen der externen Kameraparameter ermittelt werden.

## 6.4 Szenenrekonstruktion mit Sampson-Approximation

Im letzten Schritt des Arbeitsprozesses, wird nun noch die Szenen mit Hilfe eines Triangulationsverfahrens rekonstruiert. Wie bereits im Kapitel Synthetische Rekonstruktion erwähnt wurde, ist es bei den Fehlerhaften Bildkoordinaten nicht möglich die 3D-Szenenpunkte durch eine einfache Rückprojektion der Bildpunkte zu einem Punkt im 3D-Raum zu erhalten. liegt nur einer der beiden Bildpunkte  $m$  oder  $m'$  nicht hundert prozentig auf der jeweiligen korrespondierenden Epipolarlinie, so liegen die rückprojizierten Strahlen windschief im Raum. Das liegt daran, dass die Bildpunkte  $m$  und  $m'$  nicht den *Epipolar-Constraint*  $m'^T F m = 0$  erfüllen. Sprich die Gleichungen  $m = PM$  und  $m' = P'M$  können nicht erfüllt werden, da es kein  $M$  gibt, dass für beide Gleichungen mit den momentanen  $m$  und  $m'$  gilt. Abbildung 6.11 veranschaulicht die Rückprojektion der Kamerazentren durch zwei Fehlerhafte Bildpunkte.

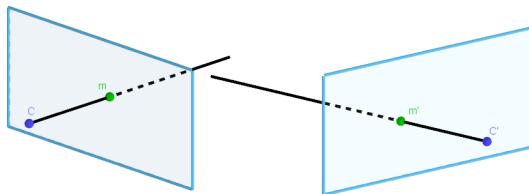


Abbildung 6.11: a)

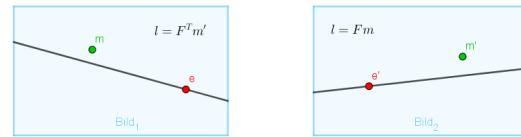


Abbildung 6.12: b)

Abbildung 6.13: a) Die rückprojizierten Strahlen der ungenauen korrespondierenden Punkte  $m$  und  $m'$  sind schief und treffen sich nicht in einem Punkt im 3D-Raum. b) The epipolar geometry for  $m, m'$ . The measured points do not satisfy the epipolar constraint. The epipolar line  $l' = Fm$  is the image of the ray through  $n$ , and  $l = F^T m'$  is the image of the ray through  $m'$ . Since the rays do not intersect,  $m'$  does not lie on  $l'$ , and  $m$  does not lie on  $l$ .

Um eine Triangulation zu ermöglichen, muss eine Methode gefunden werden, welche diesen Fehler so weit minimiert, dass es zu einer erfolgreichen Rückprojektion kommt. Die verwendete Methode zur Rekonstruktion der Szene wurde nach der Vorlage von Hartley & Zisserman[3] implementiert und wird im folgenden Schritt für Schritt beschrieben. Voraussetzung ist, dass die Projektionsmatrizen  $P$  und  $P'$ , sowie die Fundamentalmatrix  $F$  bekannt sein müssen. Sind die Projektionsmatrizen  $P$  und  $P'$  bis auf eine projektive oder affine Komponenten bekannt, so ist es wünschenswert, wenn die

Triangulierung auf einem affinen und projektiv invarianten Verfahren funktioniert[3]. Die hier verwendeten Projektionsmatrizen sind bis auf eine Skaleninvarianz genau bestimmt, was unter den Fall der affinen Invarianz Fällt. Wären die intrinsischen Kameraparameter  $K$  und  $K'$  nicht bekannt gewesen, gibt es die Möglichkeit die Projektionsmatrizen über die Fundamentalmatrix  $F$  mit dem, im Buch von *Hartley & Zisserman* beschriebenen *Stratified-Approach* bis auf eine projektive Invarianz genau zu bestimmen[3]. Die hier verwendete Triangulierung ist nur projektiv invariant, kann aber trotzdem genutzt werden. Die rekonstruierte Szene ist, dann wie im Minimalbeispiel auch, nicht auf ihre Originalmaße skaliert, was aber nach der Triangulierung noch getan werden kann. Die Triangulierung ist deshalb projektiv invariant, weil alle Rechenoperationen, wie die Minimierungen von Distanzen, sich nur auf die 2D-Bildern beziehen und sich nicht in den projektiven 3D-Raum erstreckt[3]. Der Grundgedanke der Triangulation ist, dass zwei Punkte  $\hat{m}$  und  $\hat{m}'$  gefunden werden sollen, die möglichst nah an den ursprünglichen  $m$  und  $m'$  sind und gleichzeitig den *Epipolar-Constraint*  $\hat{x}'^T F \hat{x} = 0$  erfüllen. Dies erfolgt durch die Minimierung einer Kostenfunktion  $C$ . In vielen bekannten Computer Vision Applikationen wird für diese Minimierung eine numerische Lösung gewählt, die wohl bekannteste Methode ist der *Levenberg-Marquardt Algorithmus*[3]. Jedoch hat sich gezeigt, dass ein nahezu optimales Minimum der geometrischen Kostenfunktion  $C$  auch durch eine Annäherung ersten Grades finden lässt. Die Annährung um die es sich handelt ist die sogenannten *Sampson-approximation*. Es sollen zwei Punkte  $\hat{m}$  und  $\hat{m}'$  gefunden werden, welche nahe an den Ursprünglichen  $m$  und  $m'$  liegen und gleichzeitig den *Epipolar-Constraint* erfüllen.  $\hat{m}$  und  $\hat{m}'$  sollen durch Minimierung einer Kostenfunktion  $C$  ermittelt werden, welche die Distanz  $d$  zwischen  $m$  und  $\hat{m}$  und  $m'$  und  $\hat{m}'$  minimiert.

$$C(m, m') = d(m, \hat{m})^2 + d(m', \hat{m}')^2 \quad (6.5)$$

Die projizierten Punkte  $\hat{m}$  und  $\hat{m}'$  eines 3D-Objektpunktes  $\hat{M}$  liegen auf einem paar korrespondierender Epipolarlinien. Jedes Punktpaar, welches den *Epipolar-Constraint* erfüllt, liegt auf einem paar korrespondierender Epipolarlinien. Die optimalen  $\hat{m}$  und  $\hat{m}'$  liegen am Fuße des Lots, welches von den ursprünglich projizierten Punkten  $m$  und  $m'$  ausgeht. Zusätzlich liegen  $\hat{m}$  und  $\hat{m}'$  auf korrespondierenden Epipolarlinien  $l$  und  $l'$ [3].

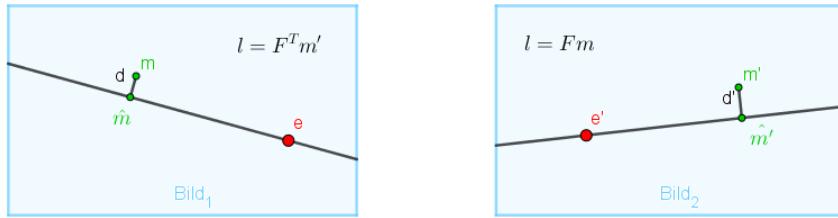


Abbildung 6.14: Grafische Darstellung der optimalen Punkte  $\hat{m}$  und  $\hat{m}'$

Jedes Punktpaar auf  $l$  und  $l'$  würde den *Epipolar-Constraint* erfüllen, jedoch minimieren nur  $x_\perp$  und  $x'_\perp$  die quadratischen Distanzen in der Kostenfunktion  $C$ . Ausgehend von dieser Aussagen wird  $C$  so umformuliert, dass gilt  $d(m, \hat{m}) = d(\hat{m}, l)$  und  $d(m', \hat{m}') = d(\hat{m}', l')$ , was jeweils den senkrechten Abstand  $m$  zu  $l$  und  $m'$  zu  $l'$  beschreibt. Werden  $l$  und  $l'$  frei aus allen möglichen Epipolarlinien gewählt, so wird immer der senkrechte Abstand von  $x$  zu dieser gewählten  $l$  berechnet, entsprechend gilt das auch für  $m'$  und irgendeine  $l'$ . Nun muss der Abstand  $d(\hat{m}, l)^2 + d(\hat{m}', l')$  minimiert werden, da natürlich nicht einfach jede beliebigen Epipolarlinien genutzt werden können. Es wird eine Strategie mit insgesamt vier Schritten für die Minimierung verfolgt[3]. Zuerst werden die Epipolarlinienbündel pro Bild so parametrisiert, dass beispielsweise eine Epipolarlinie im ersten Bild als  $l(t)$  geschrieben werden kann. Danach wird die Fundamentalmatrix  $F$  dazu benutzt, die entsprechend korrespondierende Epipolarlinie  $l'$  zu berechnen. Die Kostenfunktion  $C$  kann somit als eine Funktion von  $t$  definiert

werden. Schlussendlich muss ein Wert für  $t$  gefunden werden, welcher  $C$  minimal werden lässt.

$$C(m, m') = d(m, \hat{m})^2 + d(m', \hat{m}')^2 \quad (6.6)$$

$$\rightsquigarrow C(m, m') = d(m, l(t))^2 + d(m', l'(t))^2 \quad (6.7)$$

Es kann passieren, dass ein Bildpunkt korrespondierend zum jeweiligen Epipol des anderen Bildes ist, der Rückprojizierte Punkt im 3D-Raum würde sich dann auf der Basislinie der zwei Projektionszentren befinden und es ist somit nicht möglich ihn zu rekonstruieren. Um solche Fälle zu vermeiden, wird eine Transformation der Punkte  $m$  und  $m'$  in den Ursprung  $(0, 0, 1)^T$  zu verschieben.

$$T = \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \\ 0 & 0 & 1 \end{bmatrix} \rightsquigarrow \bar{m} = T \cdot m \quad (6.8)$$

$$T' = \begin{bmatrix} 1 & 0 & -x' \\ 0 & 1 & -y' \\ 0 & 0 & 1 \end{bmatrix} \rightsquigarrow \bar{m}' = T' \cdot m' \quad (6.9)$$

Die Fundamentalmatrix  $F$  wird dann wieder an die neu translatierten Punkte  $\bar{m}$  und  $\bar{m}'$  angepasst.

$$\bar{F} = T'^{-T} F T^{-1} \quad (6.10)$$

Als nächstes wird  $F$  mit  $T$  und  $T'$  so Transformiert, dass den *Singularity-Constraint* zwischen Des Weiteren sollen die Epipole auf die x-Achse an die Punkte  $\hat{e} = (1, 0, f)^T$  und  $\hat{e}' = (1, 0, f')^T$ , wobei  $f$  und  $f'$  nahezu null sein werden. Die Epipole lassen sich durch den rechten und linken Kern der neuen  $\bar{F}$  berechnen. Angenommen  $f$  und  $f'$  seien genau 0, so lauten die Koordinaten der Epipole  $e = (1, 0, f)^T$  und  $e' = (1, 0, f')^T$ . Ist dies der Fall so hat  $\bar{F}$  für welche dann gilt, dass  $\bar{F}(1, 0, f)^T = (1, 0, f')\bar{F} = 0$  eine spezielle Form.

$$\bar{F} = \begin{pmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{pmatrix} \quad (6.11)$$

$$\begin{pmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ f \end{pmatrix} = \begin{pmatrix} ff'd + (-ff'd) \\ -fb + fb \\ -fd + fd \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (6.12)$$

$$(1 \ 0 \ f') \cdot \begin{pmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{pmatrix} = \begin{pmatrix} ff'd + (-ff'd) \\ -f'c + f'c \\ -f'd + f'd \end{pmatrix} = (0 \ 0 \ 0) \quad (6.13)$$

Im Realfall sind die Werte der Epipole  $e$  und  $e'$  nicht so rein wie im Beispiel gezeigt. Aufgrund dessen, werden zwei Rotationsmatrizen aufgestellt, welche die Epipole  $e$  und  $e'$  auf  $Re = (1, 0, e_3)$  und  $R'e' = (1, 0, e'_3)$  rotiert.

$$R = \begin{bmatrix} e_1 & e_2 & 0 \\ -e_2 & e_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.14)$$

$$R' = \begin{bmatrix} e'_1 & e'_2 & 0 \\ -e'_2 & e'_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.15)$$

$\bar{F}$  wird dann nochmals mit  $R'FR^T$  ersetzt. Die Einträge in  $\bar{F}_{Rot}$  haben nun die Form wie in Gleichung 7.10, mit  $f = e_3$ ,  $f' = e'_3$ ,  $a = \bar{F}_{Rot,22}$ ,  $b = \bar{F}_{Rot,23}$ ,  $c = \bar{F}_{Rot,32}$  und  $d = \bar{F}_{Rot,33}$ . Angenommen eine Epipolarlinie verläuft nun durch einen Punkt  $(0, t, 1)^T$  und dem Epipol  $e = (1, 0, f)^T$ , wird diese Epipolarlinie mit  $l(t)$  bezeichnet. Das Kreuzprodukt dieser beiden Punkte beschreibt die Gerade.

$$\begin{pmatrix} 0 \\ t \\ 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ f \end{pmatrix} = \begin{pmatrix} tf \\ 1 \\ -t \end{pmatrix} \quad (6.16)$$

Die quadratische Distanz dieser Linie zum Ursprung wird dann bezeichnet mit:

$$d(m', l'(t))^2 = \frac{t^2}{1 + (tf)^2} \quad (6.17)$$

Für die Herleitung von Gleichung 7.16 wird angenommen die Gleichung einer Gerade sei zunächst in Koordinatenform Dargestellt

$$Ax + By - C = 0 \quad (6.18)$$

Die Selbe Gerade kann auch in Normalform ausgedrückt werden

$$\vec{n} \cdot (\vec{x} - \vec{p}) = 0 \quad (6.19)$$

$$\begin{pmatrix} A \\ B \end{pmatrix} \cdot (\vec{x} - \vec{p}) = 0 \quad (6.20)$$

Der Abstand eines Punktes zu einer Geraden kann folgend ausgedrückt werden.

$$\vec{v} = \frac{\vec{p} \cdot \vec{n}}{\vec{n} \cdot \vec{n}} \cdot \vec{n} \rightsquigarrow \frac{-C}{A^2 + B^2} \cdot \begin{pmatrix} A \\ B \end{pmatrix} \quad (6.21)$$

$$\|\vec{v}\| = \frac{|\vec{p} \cdot \vec{n}|}{\|\vec{n}\|^2} \cdot \|\vec{n}\| \rightsquigarrow \|\vec{v}\| = \frac{|\vec{p} \cdot \vec{n}|}{\|\vec{n}\|} \quad (6.22)$$

$$\Rightarrow |C| = |\vec{p} - \vec{n}| \quad (6.23)$$

$$\Rightarrow \sqrt{|A^2 + B^2|} = \|\vec{n}\| \quad (6.24)$$

$$\|\vec{v}\| = \frac{|C|}{\sqrt{|A^2 + B^2|}} \quad (6.25)$$

Werden nun  $A$ ,  $B$  und  $C$  mit den Werten der Geraden  $(tf, 1, -t)^T$  ersetzt, kann Gleichung 7.16 rekonstruiert werden.

$$A = tf, B = 1, C = -t, \vec{v} = d \quad (6.26)$$

$$d^2 = \frac{t^2}{\sqrt{((tf)^2 + 1^2)^2}} = \frac{t^2}{(tf)^2 + 1^2} = \frac{t^2}{1 + (tf)^2} \quad (6.27)$$

Für korrespondierende Epipolarlinie  $l'(t)$  wird der Punkt  $(0, t, 1^T)$  und die Fundamentalmatrix  $\bar{F}_{Rot}$  multipliziert.

$$l'(t) = F(0, t, 1)^T = (-f'(ct + d), at + b, ct + d)^T. \quad (6.28)$$

Für die quadratische Distanz  $d(m', l'(t))^2$  ergibt sich dann:

$$d(m', l'(t))^2 = \frac{(ct + d)^2}{(at + b)^2 + f'^2(ct + d)^2} \quad (6.29)$$

Für die ursprüngliche Kostenfunktion  $C$  kann jetzt on eine eine Funktion  $s(t)$  überestzt werden.

$$C(m, m') = d(m, \hat{m})^2 + d(m', \hat{m}')^2 \quad (6.30)$$

$$\rightsquigarrow C(m, m') = d(m, l(t))^2 + d(m', l'(t))^2 \quad (6.31)$$

$$\rightsquigarrow s(t) = \frac{t^2}{1 + (tf)^2} + \frac{(ct + d)^2}{(at + b)^2 + f'^2(ct + d)^2} \quad (6.32)$$

$s(t)$  ist dann Minimal, wenn für dessen Ableitung gilt  $s'(t) = 0$ . Werden die beiden Terme in  $s(t)$  Nennergleich gemacht und der Nenner gleich Null gesetzt, ergibt sich der folgende Ausdruck  $g(t)$

$$g(t) = t((at + b)^2 + f'^2(ct + d)^2)^2 - (ad - bc)(1 + f^2t^2)^2(at + b)(ct + d) \quad (6.33)$$

Funktion  $g(t)$  ist ein Polynom vom Grad 6. Das Minimum von  $s(t)$  ergibt sich aus einer der 6 möglichen Lösungen für  $t$  aus  $g(t)$ . Für die Ermittlung des Minimums werden nur die reellen Lösungen in betracht gezogen, die nicht-reellen Lösungen können ignoriert werden. Die reellen Lösungen für  $t$  aus  $g(t)$ , werden dann wieder in  $s(t)$  eingesetzt. Das  $t$ , welches durch einsetzte in  $s(t)$  den kleinsten Wert ergibt, ist das gesuchte Minimum. Ist  $t_{min}$  gefunden, können die Epipolarlinien  $l = (tf, 1, -t)$  und  $l'$  durch einsetzen von  $t_{min}$  berechnet werden. Nun müssen nur noch die zwei Punkte  $\hat{m}_{Rot}$  und  $\hat{m}'_{Rot}$  gefunden werden, welche dieser Epipolarlinien vom Ursprung aus am nächsten sind. Der Punkt vom Ursprung aus mit dem geringsten Abstand zu einer Linie  $(\lambda, \mu, v)$  berechnet sich durch  $(-\lambda \cdot v, -\mu \cdot v, \lambda^2 + \mu^2)$

$$l = (tf, 1, -t) \quad (6.34)$$

$$\hat{m}_{Rot} = (-(tf) \cdot v, -1 \cdot v, (tf)^2 \cdot 1^2) \quad (6.35)$$

Nachdem zu beiden Linien  $l$  und  $l'$  der jeweils nächste Punkte  $\hat{m}_{Rot}$  und  $\hat{m}'_{Rot}$  vom Ursprung aus gefunden wurden, müssen diese nun wieder mit an ihre Ausgangsposition verschoben werden.

$$\hat{m} = T^{-1}R^T\hat{m}_{Rot} \quad (6.36)$$

$$\hat{m}' = T'^{-1}R'^T\hat{m}'_{Rot} \quad (6.37)$$

Vergleicht man die Punkte  $m$  und  $\hat{m}$  und die Punkte  $m'$  und  $\hat{m}'$ , so kann die minimalen Abweichungen der Punkte voneinander sehen. Um nun noch den Punkt  $\hat{M}$  im 3D-Raum zu rekonstruieren, kann nun jegliche bekannte Methode für die Triangulierung verwendet werden. Durch die zuvorigen Rechenoperationen ist nun gewährleistet, dass sich die Gerade der Projektionszentren  $C$  und  $C'$  durch ihre jeweiligen Bildpunkte  $\hat{m}$  und  $\hat{m}'$  auf jeden Fall im Raum treffen[3]. Für Doe Rückprojektion der Punkte  $\hat{m}$  und  $\hat{m}'$  zu  $\hat{M}$  wurde ebefalls sich wieder auf ein Verfahren von *Hartley & Zisserman* berufen. Es handelt sich um eine lineare Triangulierungsmethode. Die Gleichungen  $\hat{m} = P\hat{M}$ ,  $\hat{m}' = P'\hat{M}$  werden in eine Gleichung der Form  $AX = 0$  zusammengeschrieben. Durch die Verwendung des Kreuzproduktes, wird die Homogene Komponente eliminiert.

$$\hat{m} \times (P\hat{M}) = 0 \quad (6.38)$$

$$\hat{m}' \times (P\hat{M}') = 0 \quad (6.39)$$

Was ausgeschrieben für  $\hat{m}$  und  $\hat{m}'$  zu den folgenden drei Gleichungen führt.

$$x(p^{3T}X) - (p^{1T}X) = 0 \quad (6.40)$$

$$y(p^{3T}X) - (p^{2T}X) = 0 \quad (6.41)$$

$$x(p^{3T}X) - y(p^{1T}X) = 0 \quad (6.42)$$

$p^{iT}$  bezeichnet hier jeweils die Reihen der Projektionsmatrix  $P$  beziehungsweise  $P'$ . Die Matrix  $A$  stellt sich, aufgrund der Tatsache, dass die Komponenten der Gleichungen 7.37 bis 7.39 linear zu  $\hat{M}$  sind, wie folgt zusammen.

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix} \quad (6.43)$$

Die zwei Wege eine solche Matrix zu lösen sind bereits bekannt, so kann zum einen wieder die Inhomogenen Methode angewandt werden und Kern dieser Koeffizientenmatrix berechnet werden, oder es kann das homogene Verfahren verfolgt werden. Hier wird die Singulärwertzerlegung an A durchgeführt und derjenige Vektor gesucht werden, welcher mit dem kleinsten Singulärwert korrespondiert[3]. Das Ergebnis ist jeweils  $\hat{M}$  im 3D-Raum. Da die vorherige  $P$  und  $P'$  nur bis zu einem Skalierungsfaktor genau bestimmt wurden, muss nachdem die Punkte rekonstruiert wurden noch die Skalierung auf ihre ursprüngliche Größe erfolgen. Dies ist am einfachsten, wenn eine Referenzgröße zuvor in der Originalszene gemessen wurde. Die Abbildungen 7.15 und 7.16 zeigen die Rekonstruierte Szene des Beispiels, jedoch noch nicht skaliert auf ihre Ursprungsgrößen. Abbildung 7.15 zeigt die 3D Szene. Der Rote Punkt symbolisiert die Position von  $C$  also der Canon 6D und der grüne Punkt symbolisiert die Position von  $C'$  also der Canon 60D. Die Blauen Punkte sind die durch den *SURF*-Algorithmus detektierten Punkte der Szene. Abbildung 7.16 zeigt die rekonstruierten Objektpunkte als 2D-Punkte, hierfür wurden ihre Koordinaten einfach durch ihren Tiefenwert geteilt.

## 6.5 Ergebnisse einer Stereoanalyse mit Kameras unterschiedlicher Auflösung

Für den Test, ob Szeneriekonstruktion im Realbeispiel auch mit unterschiedlichen Kameraauflösungen funktioniert, wurde eine der von Matlab ermittelten Kameramatrizen  $K'$  und auch die durch den Surf Algorithmus detektierten Punkte jeweils skaliert. In Kapitel ?? wurden die einzelnen Bauteile der Kameramatrix genau beschrieben. Die Kameramatrix  $K'$  aus Matlab für die Canon 60D gegeben.

$$K' = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.44)$$

$\alpha_x$  und  $\alpha_y$  setzen sich auch dem Abstand des Kamerazentrums zum Hauptpunkt zusammen, welcher in dieser Arbeit als mit  $\zeta$  bezeichnet wurde, und den Kantenlängen der Pixel auf dem Sensor  $m_x$  und  $m_y$ . Um die Auflösung der Kamera zu verändern, wird auf  $\alpha_x$  und  $\alpha_y$  jeweils ein beliebiger Faktor dazu

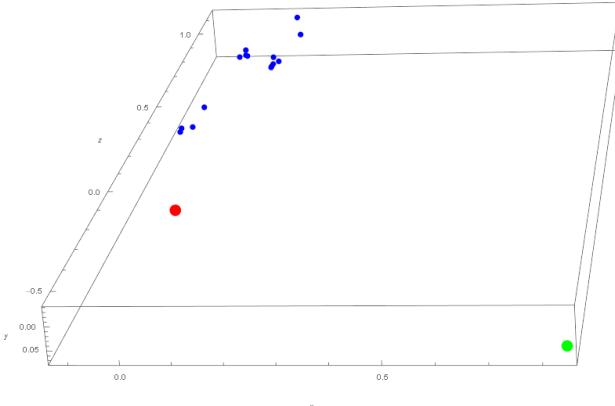


Abbildung 6.15: Rekonstruierte Szene, unskaliert in Pixeleinheiten

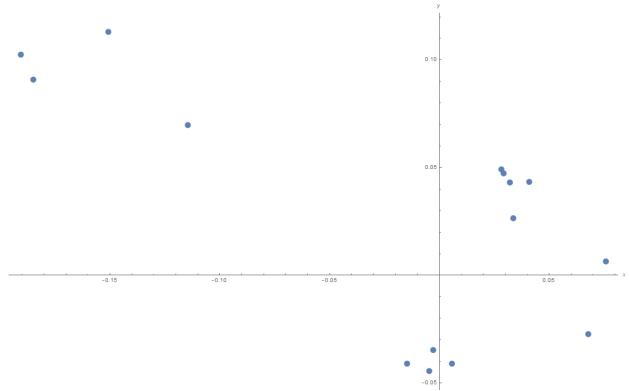


Abbildung 6.16: Rekonstruierte Szene, unskaliert, in Pixeleinheiten und in einem 2D-Plot geschrieben

multipliziert. Zum Beweis, dass die Rekonstruktion der externen Kameraparameter und die Szenekonstruktion, bei egal welcher Skalierung, die ähnlichen Ergebnisse liefern, wurde die Kameramatrix  $K'$  mit den Verhältnissen  $[2 : 2]$ ,  $[5 : 2]$ ,  $[2 : 1]$ ,  $[1 : 2]$  und  $[1.2 : 2.3]$  skaliert.

$$K'_{[2:2]} = \begin{bmatrix} \alpha_x \cdot 2 & s & x_0 \cdot 2 \\ 0 & \alpha_y \cdot 2 & y_0 \cdot 2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$K'_{[5:2]} = \begin{bmatrix} \alpha_x \cdot 5 & s & x_0 \cdot 5 \\ 0 & \alpha_y \cdot 2 & y_0 \cdot 2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$K'_{[2:1]} = \begin{bmatrix} \alpha_x \cdot 2 & s & x_0 \cdot 2 \\ 0 & \alpha_y \cdot 1 & y_0 \cdot 1 \\ 0 & 0 & 1 \end{bmatrix}$$

$$K'_{[1:2]} = \begin{bmatrix} \alpha_x \cdot 1 & s & x_0 \cdot 1 \\ 0 & \alpha_y \cdot 2 & y_0 \cdot 2 \\ 0 & 0 & 1 \end{bmatrix}$$

$$K'_{[1.2:2.3]} = \begin{bmatrix} \alpha_x \cdot 1.2 & s & x_0 \cdot 1.2 \\ 0 & \alpha_y \cdot 2.3 & y_0 \cdot 2.3 \\ 0 & 0 & 1 \end{bmatrix}$$

Die Formulierung, dass die jeweils neu rekonstruierten Szenen ähnlich sind, wurde deshalb verwendet, da durch die zuvorigen Fehler der korrespondierenden Punkte und später, bei der Triangulierung, durch die *Sampson-Approximation* Abweichungen auftreten können. Als Beweise werden im folgenden vier Beispiele für die vier Lösungen der rekonstruierten Translationsmatrizen  $R'$  aufgezeigt. Des Weiteren werden die 3D-Plots und 2D-Plots der rekonstruierten Szenen bei unterschiedlich Auflösungen im Vergleich mit der Szene bei gleichen Auflösungen gezeigt. Die Koordinaten sind in unskalierten Pixel единheiten gegeben. Die Originalszene ist in Abbildung 7.15 und 7.16 zu sehen. Zu beachten ist, dass die Ausgabe des 3D Plots in *Mathematica* manchmal rechtsdrehend, manchmal linksdrehend dargestellt sind, weshalb der Eindruck aufkommt, die Szene und die Kameraposition seien gespiegelt dargestellt. Dies leider auf ein generellen Darstellungsproblem von 3D-Plots in *Mathematica* zurückzuführen.

Dieses kann mit zusätzlichem Code bereinigt werden, wurde aber zu diesem Zeitpunkt noch nicht implementiert.

$$\begin{aligned} P1 &= \begin{pmatrix} 0.991092 & 0.120891 & 0.0558752 & -0.812277 \\ -0.120366 & 0.992649 & -0.0126719 & 0.0389145 \\ -0.0569964 & 0.00583352 & 0.998357 & 0.581973 \end{pmatrix} \\ P2 &= \begin{pmatrix} 0.991092 & 0.120891 & 0.0558752 & 0.812277 \\ -0.120366 & 0.992649 & -0.0126719 & -0.0389145 \\ -0.0569964 & 0.00583352 & 0.998357 & -0.581973 \end{pmatrix} \\ P3 &= \begin{pmatrix} 0.378236 & -0.0296342 & -0.925235 & -0.812277 \\ 0.0547644 & -0.997021 & 0.0543212 & 0.0389145 \\ -0.924088 & -0.0712161 & -0.375487 & 0.581973 \end{pmatrix} \\ P4 &= \begin{pmatrix} 0.378236 & -0.0296342 & -0.925235 & 0.812277 \\ 0.0547644 & -0.997021 & 0.0543212 & -0.0389145 \\ -0.924088 & -0.0712161 & -0.375487 & -0.581973 \end{pmatrix} \end{aligned}$$

Abbildung 6.17: Zeigt die rekonstruierte Matrix  $R'$  bei unveränderter Auflösung. Die Auflösungen von  $C_\delta$  und  $C'_\delta$  sind die selben.

$$\begin{aligned} P1 &= \begin{pmatrix} 0.376619 & -0.0296193 & -0.925895 & 0.812113 \\ 0.0551443 & -0.996999 & 0.0543246 & -0.0388039 \\ -0.924725 & -0.0715175 & -0.373856 & -0.582208 \end{pmatrix} \\ P2 &= \begin{pmatrix} 0.376619 & -0.0296193 & -0.925895 & -0.812113 \\ 0.0551443 & -0.996999 & 0.0543246 & 0.0388039 \\ -0.924725 & -0.0715175 & -0.373856 & 0.582208 \end{pmatrix} \\ P3 &= \begin{pmatrix} 0.991142 & 0.121017 & 0.0546967 & 0.812113 \\ -0.120498 & 0.992632 & -0.0126975 & -0.0388039 \\ -0.0558303 & 0.00599422 & 0.998422 & -0.582208 \end{pmatrix} \\ P4 &= \begin{pmatrix} 0.991142 & 0.121017 & 0.0546967 & -0.812113 \\ -0.120498 & 0.992632 & -0.0126975 & 0.0388039 \\ -0.0558303 & 0.00599422 & 0.998422 & 0.582208 \end{pmatrix} \end{aligned}$$

Abbildung 6.18: Zeigt die rekonstruierte Matrix  $R'$  wenn  $K'$  mit einem Verhältnis von  $[5 : 2]$  skaliert wurde

$$\begin{aligned} P1 &= \begin{pmatrix} 0.990947 & 0.120272 & 0.0596553 & 0.810768 \\ -0.119751 & 0.992728 & -0.0122401 & -0.0387536 \\ -0.0606937 & 0.0049855 & 0.998144 & -0.584083 \end{pmatrix} \\ P2 &= \begin{pmatrix} 0.990947 & 0.120272 & 0.0596553 & -0.810768 \\ -0.119751 & 0.992728 & -0.0122401 & 0.0387536 \\ -0.0606937 & 0.0049855 & 0.998144 & 0.584083 \end{pmatrix} \\ P3 &= \begin{pmatrix} 0.37685 & -0.0292569 & -0.925812 & 0.810768 \\ 0.0543725 & -0.997079 & 0.0536412 & -0.0387536 \\ -0.924677 & -0.0705534 & -0.374158 & -0.584083 \end{pmatrix} \\ P4 &= \begin{pmatrix} 0.37685 & -0.0292569 & -0.925812 & -0.810768 \\ 0.0543725 & -0.997079 & 0.0536412 & 0.0387536 \\ -0.924677 & -0.0705534 & -0.374158 & 0.584083 \end{pmatrix} \end{aligned}$$

Abbildung 6.19: Zeigt die rekonstruierte Matrix  $R'$  wenn  $K'$  mit einem Verhältnis von  $[1 : 2]$  skaliert wurde

$$\begin{aligned} P1 &= \begin{pmatrix} 0.990963 & 0.12034 & 0.0592445 & -0.81095 \\ -0.119818 & 0.99272 & -0.0122887 & 0.0387766 \\ -0.060292 & 0.0050791 & 0.998168 & 0.583829 \end{pmatrix} \\ P2 &= \begin{pmatrix} 0.990963 & 0.12034 & 0.0592445 & 0.81095 \\ -0.119818 & 0.99272 & -0.0122887 & -0.0387766 \\ -0.060292 & 0.0050791 & 0.998168 & -0.583829 \end{pmatrix} \\ P3 &= \begin{pmatrix} 0.377058 & -0.0293027 & -0.925726 & -0.81095 \\ 0.0544044 & -0.997073 & 0.0537206 & 0.0387766 \\ -0.92459 & -0.0706194 & -0.37436 & 0.583829 \end{pmatrix} \\ P4 &= \begin{pmatrix} 0.377058 & -0.0293027 & -0.925726 & 0.81095 \\ 0.0544044 & -0.997073 & 0.0537206 & -0.0387766 \\ -0.92459 & -0.0706194 & -0.37436 & -0.583829 \end{pmatrix} \end{aligned}$$

Abbildung 6.20: Zeigt die rekonstruierte Matrix  $R'$  wenn  $K'$  mit einem Verhältnis von  $[1.2 : 2.3]$  skaliert wurde

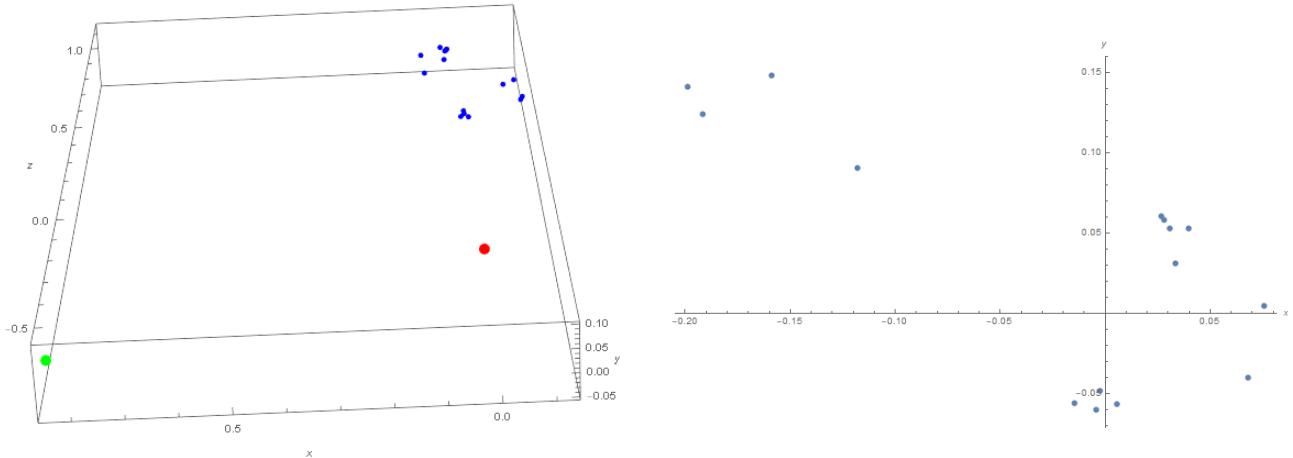


Abbildung 6.21: Rekonstruierte Szene, wenn  $K'$  mit einem Verhältnis von  $[5 : 2]$  skaliert wurde

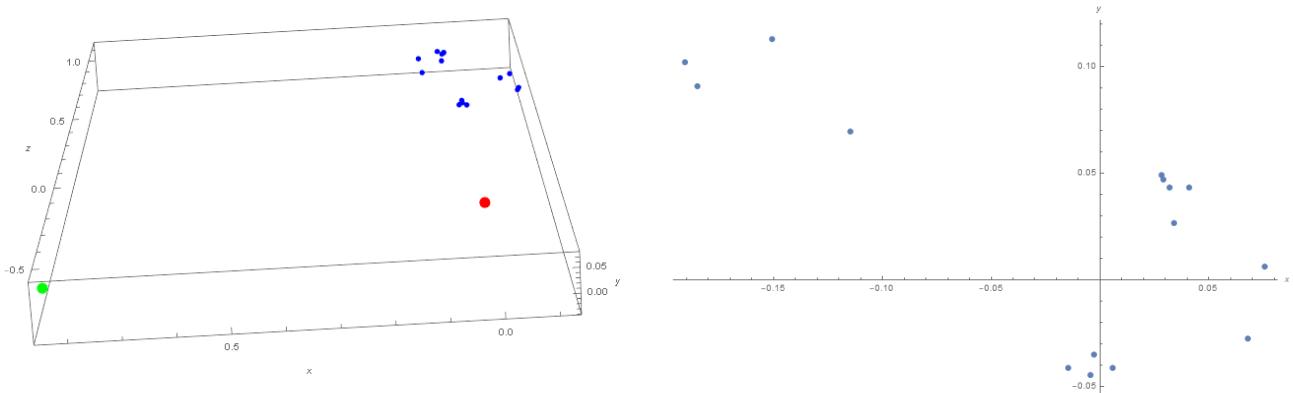


Abbildung 6.22: Rekonstruierte Szene, wenn  $K'$  mit einem Verhältnis von  $[2 : 1]$  skaliert wurde

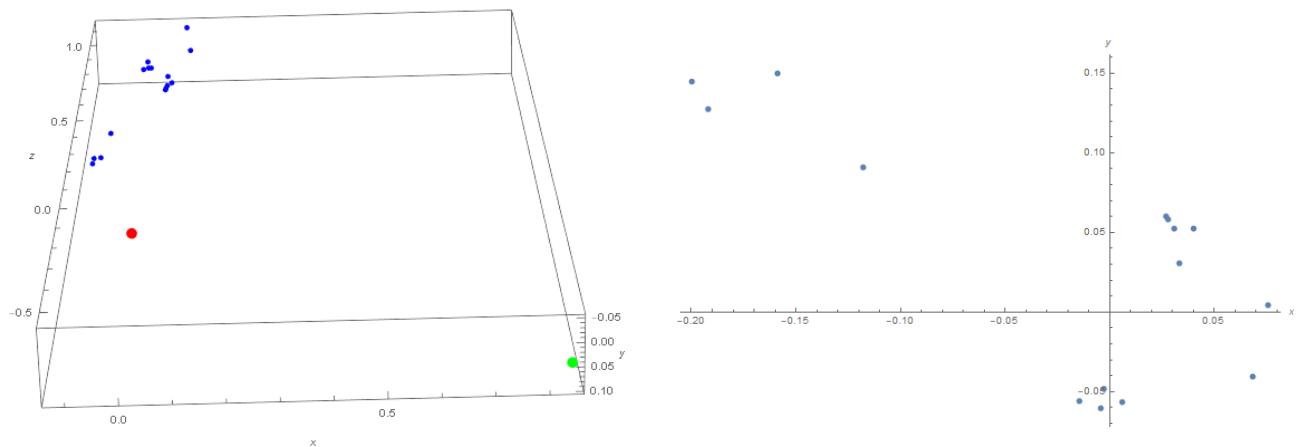


Abbildung 6.23: Rekonstruierte Szene, wenn  $K'$  mit einem Verhältnis von  $[1 : 2]$  skaliert wurde

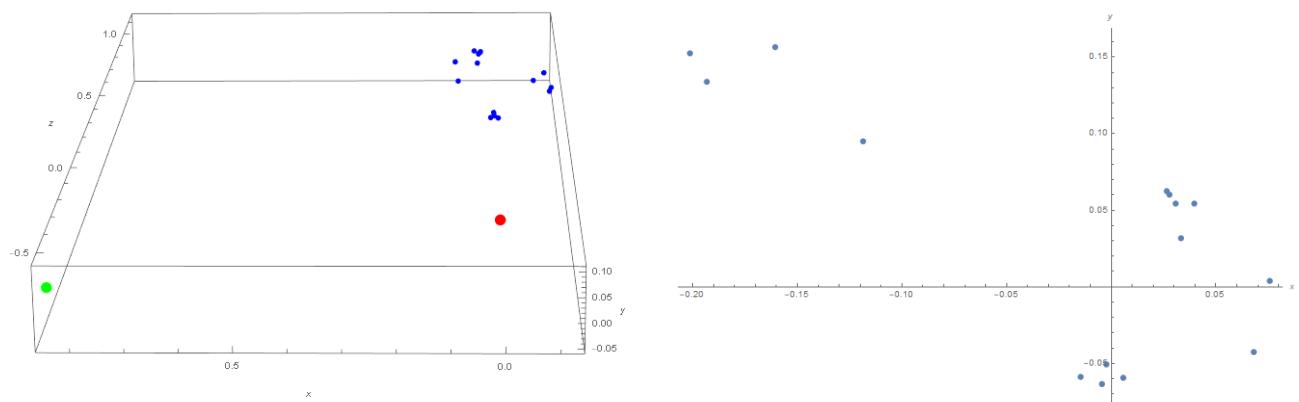


Abbildung 6.24: Rekonstruierte Szene, wenn  $K'$  mit einem Verhältnis von  $[1.2 : 2.3]$  skaliert wurde

# 7 Vergleich entwickelter RekonstruktionsAlgorithmus mit bereits vorhandenen (Matlab)

(ÜBERSCHRIFT ÄNDERN!)

(Der entwickelte Szenenrekonstruktionsalgorithmus wurde so entwickelt, dass eine Rektifizierung der Bilder, wie sie in vielen Programmen verwendet wird, nicht notwendig wird. Der Grund dafür ist, dass ein Algorithmus entsteht, welche auch mit Bildern unterschiedlicher Kameraauflösungen eine erfolgreiche Rekonstruktion vollbringt) Ausbauen, verbessern, genauer erklären wie warum, was ist mit rektifizierung gemeint, irgendwo mss noch die vorgeschichte dazu rein... Matlab verwendet verfahren über fundamentalmatrix und rektifizierung, problem: kommt nicht mit anderen Kameraauflösungen zurecht. Zwei lösungen: einmal rekonstruktion über essentielle matrix oder neuer rektifizierungsalgorithmus. Ansatz im anderen Kapitel....Hier soll es eher darum gehen den eigenem Ansatz mit dem aus Matlab zu vergleichen und auf das Prblem der Rektifizierung in Mtlab eingehen. Den hierigen Rektifizierungsansatz mit dem aus maltlab vergleichen. Problem der unterschiedlich großen Bilder ansprechen. Mein Ansatz rekonsturiert ohne dei ausgabebilder zu "kennenünd zu verändern, bbei rektifizierung wird mit den feriten Bildern gearbeitet, was bei unterschiedlichen Auflösungen zu starken verzerrungen kommen kann bei der Rektifizierung.

Arbeitsprozess Matlab eifügen

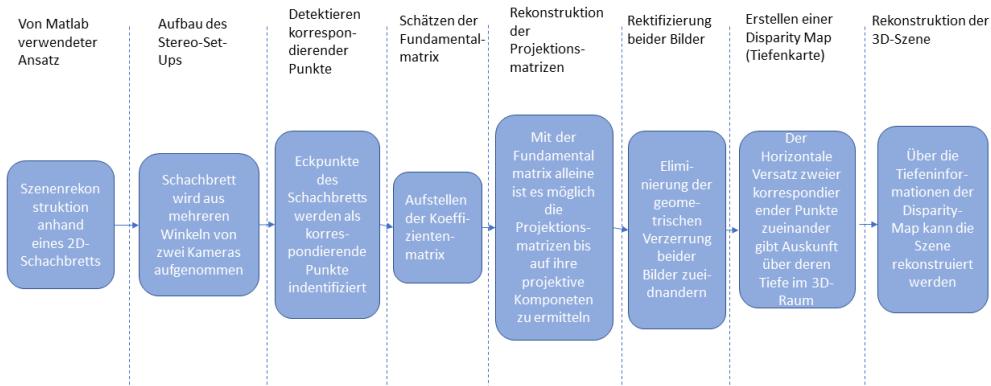


Abbildung 7.1: (Überarbeiten sie Abb 4.1) Durch Ungenauigkeiten in der korrespondierenden Punkte, verfehlten sich die Linien und es kommt zu keinem Schnittpunkt

Ein weiteres weit verbreitetes Verfahren, ist cor der Szenenrekonstruktion durch Triangulierung eine Rektifizierung beider Bilder vorzunehmen[26, 27, 28, 29]. Da bestimmte Formen der Rektifizierung keine vorherige Kalibrierung der Kameras benötigen, wird diese Methode in den meisten gängigen Echtzeit-Szenenrekonstruktionen eingesetzt. [29, 28, 30]. 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[27]. Mit Hilfe dieser Eigenschaften ist es somit möglich die entstandenen korrespondierenden Epipolarlinien als horizontale Scanlinien zu benutzen[28, 27]. 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[28, 27].



Abbildung 7.2: Beispiel eines rektifizierten Bildes. Quelle: [27]



Abbildung 7.3: Beispiel einer einfachen Tiefenkarthe eines Stereobildpaars nach der Rektifizierung. Quelle: [28]

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 setzen 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 [26]. Die Rektifizierung wurde anhand einer Methode implementiert, welcher sich ähnlich verhält wie in [31] 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[31, 29]. 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[31]. Eine Frage welche mit unter in dieser Arbeit beantwortet werden sollte, war, ob es möglich ist, ohne deutlich größeren Aufwand eine Kamerakalibrierung und Szeneriekonstruktion mit Kameras unterschiedlicher Auflösung zu gewährleisten. Im Kapitel ??, 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. [5].

In dieser Arbeit wurde ein Rektifizierungsalgorithmus nach *Zhang*[27] 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.[29, 27, 15]. 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 Fundamentalmatrix  $F$  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[27, 15]. 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.

$$\bar{x} = Hx \quad (7.1)$$

$$\bar{x}' = Hx' \quad (7.2)$$

Die Fundamentalmatrix, welche sich aus durch die Rektifizierten korrespondierenden Punkte resultiert, wird mit  $\bar{F}$  bezeichnet. Daraus folgt für die Fundamentalmatrix folgendes[27, 15]:

$$\bar{x}'^T \bar{F} \bar{x} = 0 \quad (7.3)$$

$$\rightsquigarrow x'^T H' T \bar{F} H x = 0 \quad (7.4)$$

$$\rightsquigarrow F = H'^T [i]_x H \quad (7.5)$$

Das Ziel ist es diese zwei Homographien in deren bereits erwähnten projektiven und affinen Komponenten zu zersetzen, 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.

$$H = \begin{bmatrix} u^T \\ v^T \\ w^T \end{bmatrix} = \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & w_c \end{bmatrix} \quad (7.6)$$

$$H' = \begin{bmatrix} u'^T \\ v'^T \\ w'^T \end{bmatrix} = \begin{bmatrix} u'_a & u'_b & u'_c \\ v'_a & v'_b & v'_c \\ w'_a & w'_b & w'_c \end{bmatrix} \quad (7.7)$$

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.

$$H = \begin{bmatrix} u^T \\ v^T \\ w^T \end{bmatrix} = \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & 1 \end{bmatrix} \quad (7.8)$$

$$H' = \begin{bmatrix} u'^T \\ v'^T \\ w'^T \end{bmatrix} = \begin{bmatrix} u'_a & u'_b & u'_c \\ v'_a & v'_b & v'_c \\ w'_a & w'_b & 1 \end{bmatrix} \quad (7.9)$$

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

$$H = H_p \cdot H_a \quad (7.10)$$

$$H' = H'_p \cdot H'_a \quad (7.11)$$

$H_p$  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.

$$H_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ w_a & w_b & 1 \end{bmatrix} \quad (7.12)$$

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

$$H_a = H \cdot H_p^{-1} = \begin{bmatrix} u_a - v_c w_b & v_c w_a - v_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad (7.13)$$

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.

$$H_a = H_s \cdot H_r \quad (7.14)$$

$$H_r = \begin{bmatrix} v_b - v_c w_b & v_a - v_c w_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad (7.15)$$

$$H_s = \begin{bmatrix} u_a & u_b & u_c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7.16)$$

$H_r$  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.

## 7.1 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

$$w = [e]_x \cdot z \quad (7.17)$$

$$w' = F \cdot z \quad (7.18)$$

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 bedeute, dass die Werte von  $w_a$  und  $w_b$  so nah wie möglich an den Wert 0 gebracht werden sollen.

$$H_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ w_a & w_b & 1 \end{bmatrix} \quad (7.19)$$

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[32]. 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**[27, 32]. 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[32, 33]. Zum Beispiel werden  $n$  Datenpunkte angenommen, dann gilt:

$$e = \sum_{i=1}^n [f(x_i) - g(x_i)]^2 \quad (7.20)$$

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.

$$w_i = w^T p_i \quad (7.21)$$

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.

$$\sum_{i=1}^n \left[ \frac{w_i - w_c}{w_c} \right]^2 \quad (7.22)$$

$$\rightsquigarrow \sum_{i=1}^n \left[ \frac{w^T(p_i - p_c)}{w^T p_c} \right]^2 \rightsquigarrow \sum_{i=1}^n \left[ \frac{w^T(p_i - p_c)(p_i - p_c)^T w}{w^T p_c p_c^T w} \right] \quad (7.23)$$

Vereinfacht lässt sich das auch in einer Matrixgleichung angeben

$$\frac{w^T P P^T w}{w^T p_c p_c^T w} \quad (7.24)$$

in welcher für  $P$  gilt:

$$P = \begin{bmatrix} p_{1,u} - p_{c,u} & p_{2,u} - p_{c,u} & \dots & p_{i,u} - p_{c,u} \\ p_{1,v} - p_{c,v} & p_{2,v} - p_{c,v} & \dots & p_{i,v} - p_{c,v} \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad (7.25)$$

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:

$$\frac{w'^T P' P'^T w'}{w'^T p'_c p_c'^T w'} \quad (7.26)$$

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.

$$\frac{z^T [e]_x^T P P^T [e]_x z}{z^T [e]_x^T p_c p_c^T [e]_x z} + \frac{z^T F^T P' P'^T F z}{z^T F^T p'_c p_c'^T F z} \quad (7.27)$$

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

$$A = [e]_x^T P P^T [e]_x \quad (7.28)$$

$$B = [e]_x^T p_c p_c^T [e]_x \quad (7.29)$$

$$A' = F^T P' P'^T F \quad (7.30)$$

$$B' = F^T p'_c p_c'^T F \quad (7.31)$$

$$\rightsquigarrow \frac{z^T A z}{z^T B z} + \frac{z^T A' z}{z^T B' F z} \quad (7.32)$$

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!!!* (7.33)

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.

$$z = \frac{\frac{\hat{z}_1}{\|\hat{z}_1\|} + \frac{\hat{z}_2}{\|\hat{z}_2\|}}{2} \quad (7.34)$$

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 Cholesky-Zerlegung in zwei höhere Dreiecksmatrizen zerlegt  $A = D^T D$ [34]. Dies geht nur da  $A$  nachweislich eine symmetrische und positiv-definite Matrix ist.[34] 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  $\hat{f}(y)$

$$A = D^T D \quad (7.35)$$

$$y = Dz \rightsquigarrow z = D^{-1}y \quad (7.36)$$

$$f(z) = \frac{z^T B z}{z^T A z} \quad (7.37)$$

$$\rightsquigarrow f(z) = \frac{z^T B z}{z^T D^T D z} \quad (7.38)$$

$$\hat{f}(y) = \frac{y^T D^{-T} B D^{-1} y}{y^T y} \quad (7.39)$$

Durch die Definition 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  $\hat{z}_2$  mit  $\frac{z^T B' z}{z^T A' z}$  angewandt. Sind  $\hat{z}_1, \hat{z}_2$  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  $\hat{z}_1$  und  $\hat{z}_2$ , 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 Matrizen  $H_p$  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  $45^\circ$  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.

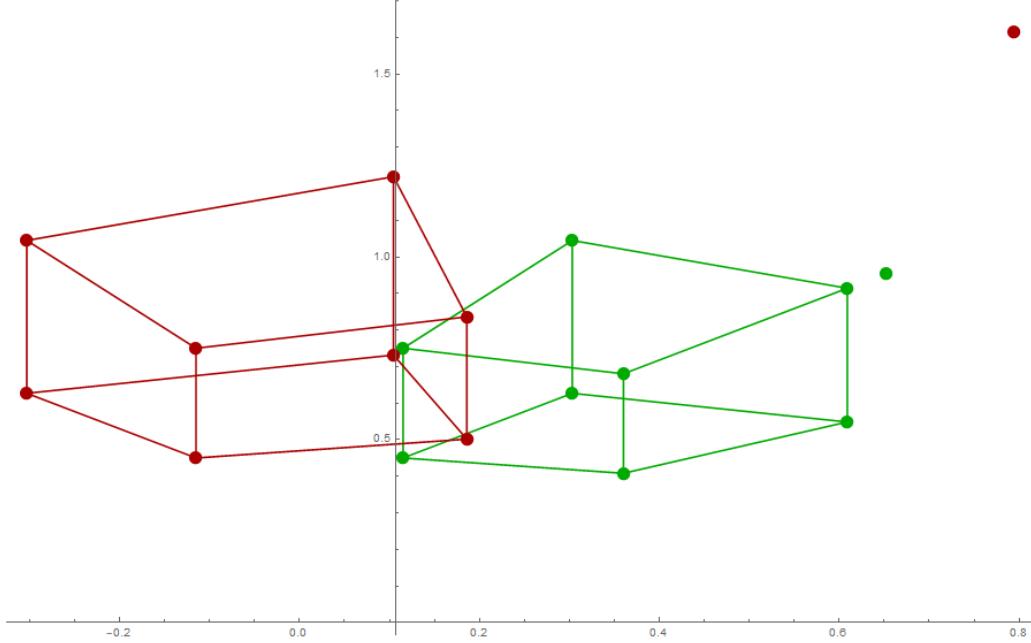


Abbildung 7.4: Aufnahmen zweier Kamerä mit den selben Auflösungen, Kamera eins(Grün) und Kamera(rot) zwei gelten jeweils  $\zeta = 1$

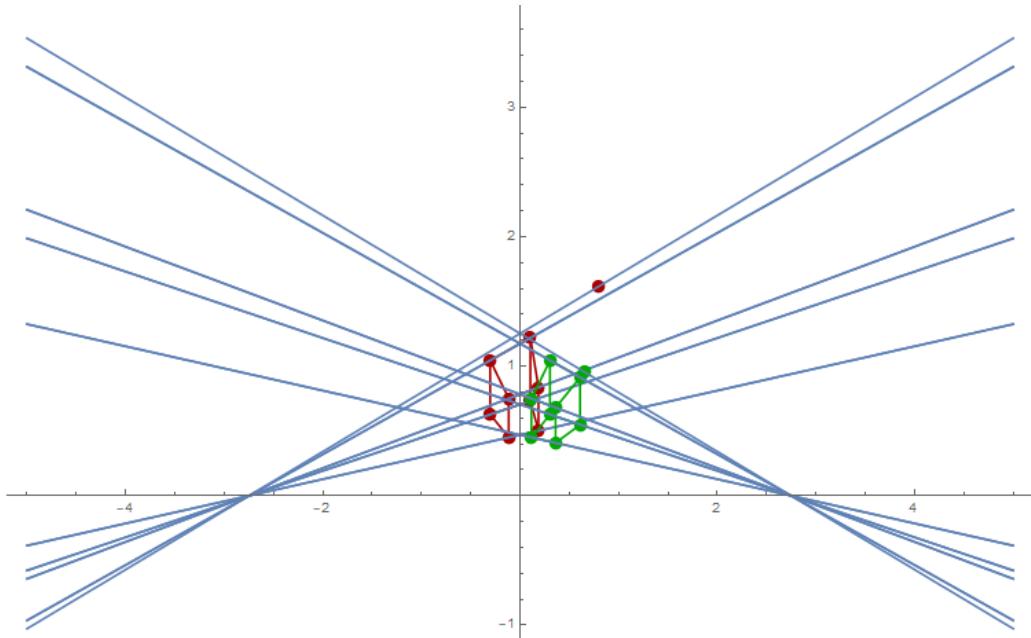


Abbildung 7.5: Epipole für Kamera eins und Kamera zwei vor der Rektifizierung

Werden nun die Matritzen  $H_p$  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 aus 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.

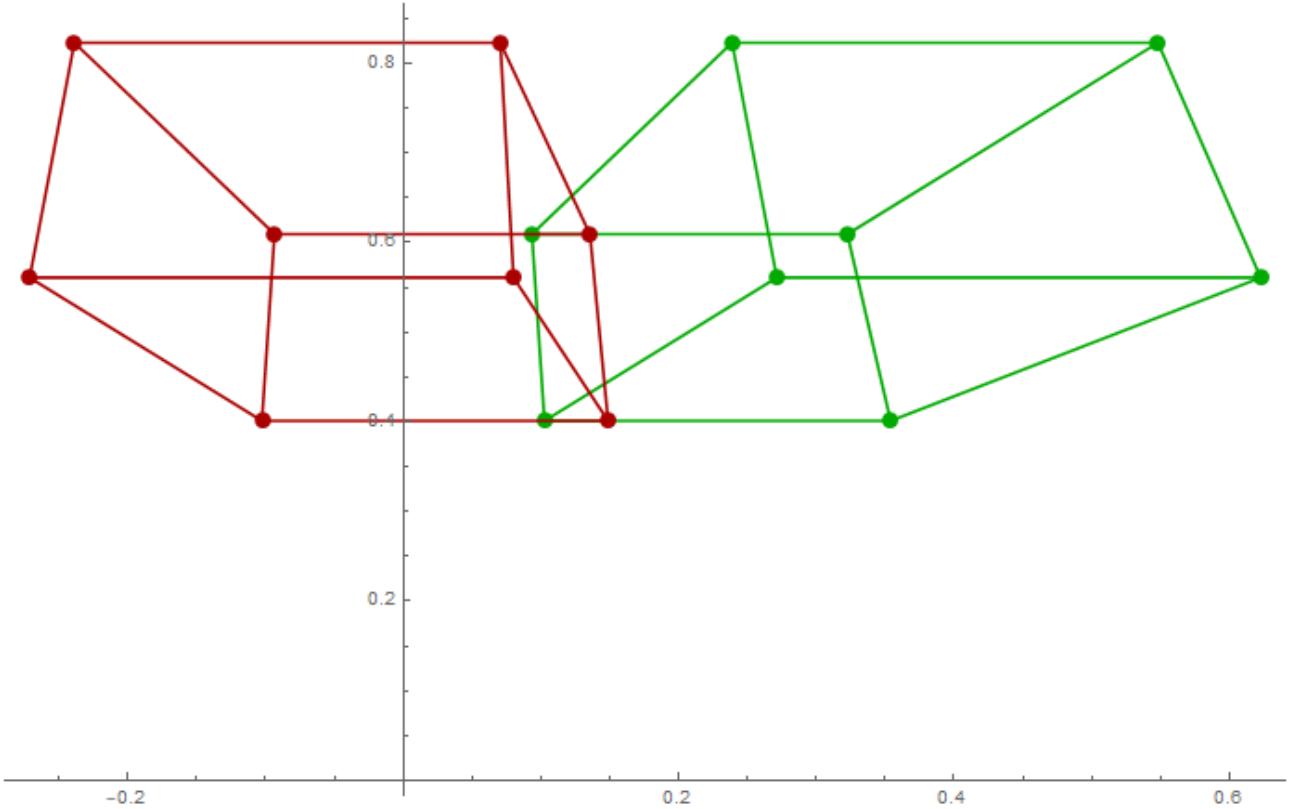


Abbildung 7.6: Abbildung beider Bilder nach anwenden der Matrizen  $H_p$  und  $H'_p$ . Die Epipole beider Bilder sind nun im unendlichen. Das die entstehenden parallelen Epipolarlinien auch hier schon horizontal ausgerichtet sind ist Zufall. Die Epipolarlinien sind immer parallel nach dieser Transformation aber die Richtung ist nicht immer automatisch bereit  $i = [1, 0, 0]$ .

## 7.2 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.

$$H_r = \begin{bmatrix} v_b - v_c w_b & v_a - v_c w_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad (7.40)$$

$$H'_r = \begin{bmatrix} v'_b - v'_c w'_b & v'_a - v'_c w'_a & 0 \\ v'_a - v'_c w'_a & v'_b - v'_c w'_b & v'_c \\ 0 & 0 & 1 \end{bmatrix} \quad (7.41)$$

$$(7.42)$$

$w$  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.

$$F = H'^T[i]_x H \quad (7.43)$$

$$F = \begin{bmatrix} v_a w'_a - v'_a w_a & v_b w'_a - v'_a w_b & v_c w'_a - v'_a \\ v_a w'_b - v'_b w_a & v_b w'_b - v'_b w_b & v_c w'_b - v'_b \\ v_a - v'_c w_a & v_b - v'_c w_b & v_c - v'_c \end{bmatrix} \quad (7.44)$$

$$v_a = F_{31} + v'_c w_a \quad (7.45)$$

$$v_b = F_{32} + v'_c w_b \quad (7.46)$$

$$v_c = F_{33} + v'_c \quad (7.47)$$

$$v'_a = v_c w'_a - F_{13} \quad (7.48)$$

$$v'_b = v_c w'_b - F_{23} \quad (7.49)$$

$$v'_c = v_c - F_{33} \quad (7.50)$$

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

$$H_r = \begin{bmatrix} F_{32} - w_b F_{33} & w_a F_{33} - F_{31} & 0 \\ F_{31} - w_a F_{33} & F_{32} - w_b F_{33} & F_{33} + v'_c \\ 0 & 0 & 1 \end{bmatrix} \quad (7.51)$$

$$H'_r = \begin{bmatrix} w'_b F_{33} - F_{23} & F_{13} - w'_a F_{33} & 0 \\ w'_a F_{33} - F_{13} & w'_b F_{33} - F_{23} & v'_c \\ 0 & 0 & 1 \end{bmatrix} \quad (7.52)$$

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.

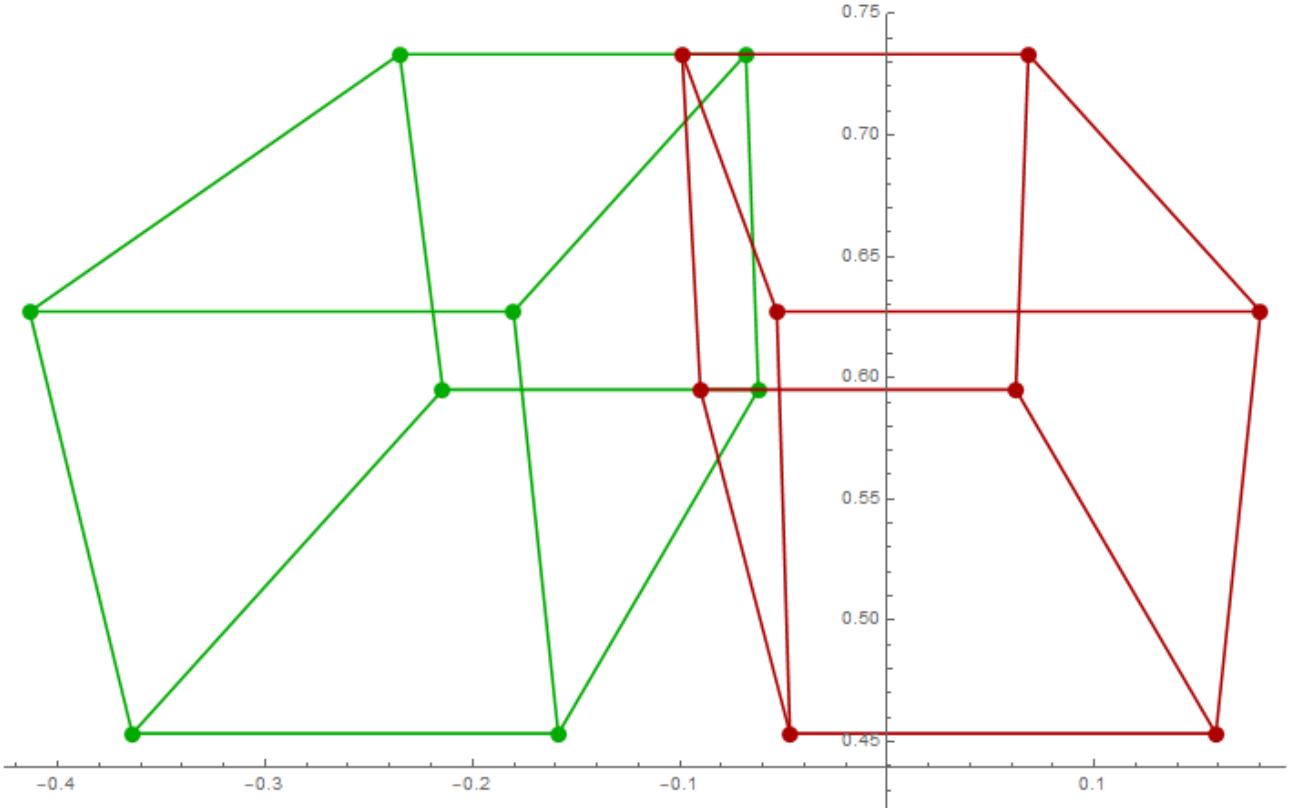


Abbildung 7.7: Abbildung beider Bilder nach anwenden der Matrizen  $H_r \cdot H_p$  und  $H'_r \cdot H'_p$ . Die Epi-polarlinien sind nun horizontal zueinander ausgerichtet

### 7.3 Scherungstransformation

Die letzte Transformation, welche an den Bilder durchgeführt werden soll, ist die sogenannten Scherungstransformation. Sie soll vor allem dazu dienen, die horizontale Verzerrung der Bilder zueinander nochmal weiter zu minimieren. Die Matrizen  $H_s$  und  $H'_s$  wirken sich hauptsächlich auf die  $u$  und  $u'$  Komponenten aus.

$$H_s = \begin{bmatrix} u_a & u_b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7.53)$$

$$H'_s = \begin{bmatrix} u'_a & u'_b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (7.54)$$

Um die richtigen Werte für  $a, a', b$  und  $b'$  zu bekommen, werden zunächst Punkte an den jeweiligen gegenüberliegenden Kanten der Bilder definiert. Da die Bilder des Quaders nicht aus tausenden von Pixeln bestehen, wie ein reales Bild, sondern nur über dessen Eckpunkte bestimmt ist, wird eine Bildbreite  $w$  und  $w'$  und eine Bildhöhe  $h$  und  $h'$  definiert. Die Höhen und Breiten der Bilder rahmen die abgebildeten Quader ein, somit wurde quasi eine Bildgröße für beide Bilder definiert. Nun können die Punkte an den Kantenhalbierenden  $a = [\frac{w-1}{2} \ 0 \ 1]^T, b = [w-1 \ \frac{h-1}{2} \ 1]^T, c = [\frac{w-1}{2} \ h-1 \ 1]^T, d = [0 \ \frac{h-1}{2} \ 1]^T$  gebildet werden. Der Gedanke, der damit verfolgt wird ist, dass die Punkte der jeweiligen gegenüberliegenden Kanten mit einander verbunden werden können und dann so ausgerichtet werden sollen, dass sie sich wieder direkt gegenüber liegen. Schematisch wird as in Abbildung ???? aufgezeigt.

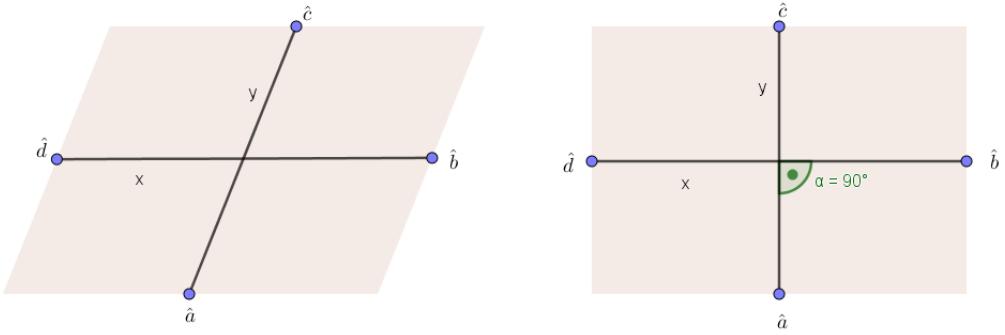


Abbildung 7.8: Die Abbildung verdeutlicht noch mal Schematisch, wie sich die Punkte ausrichten sollen. Bild a) zeigt die durch die Rektifizierung verschobenen Bildkantenmitten. Bild b= zeigt, wie sich die Bildkantenmitten durch die Scherungstransformation wieder ausrichten sollen.

Die Punkte  $a, b, c, d$  und auch  $a', b', c', d'$  geben die Bildbreiten der noch unberührten Bilder an. Nach der Rektifizierung sind die Bilder so verzerrt, dass die Kanten mitten sich meistens nicht mehr direkt gegenüber von einander befinden. Die Punkte  $a, b, c, d$  und  $a', b', c', d'$  werden mit den Matrizen  $H_p, H'_p, H_r$  und  $H'_r$  verrechnet, so dass man die genaue neue Position der Kanten Mitten nach der Rektifizierung hat.

$$\hat{a} = H_r \cdot H_p \cdot a$$

$$\hat{b} = H_r \cdot H_p \cdot b$$

$$\hat{c} = H_r \cdot H_p \cdot c$$

$$\hat{d} = H_r \cdot H_p \cdot d$$

$$\hat{a}' = H'_r \cdot H'_p \cdot a'$$

$$\hat{b}' = H'_r \cdot H'_p \cdot b'$$

$$\hat{c}' = H'_r \cdot H'_p \cdot c'$$

$$\hat{d}' = H'_r \cdot H'_p \cdot d'$$

Um aus  $\hat{a}, \hat{b}, \hat{c}, \hat{d}$  und auch  $\hat{a}', \hat{b}', \hat{c}', \hat{d}'$  wieder Punkte der affinen Ebene zu machen werden sie jeweils durch ihre dritte Komponenten geteilt, so das  $\hat{a}_w, \hat{b}_w, \hat{c}_w, \hat{d}_w$  und  $\hat{a}'_w, \hat{b}'_w, \hat{c}'_w, \hat{d}'_w$  jeweils den Wert eins besitzen. Danach können die Vektoren  $\vec{x}$  und  $\vec{y}$  aus den Differenzen der sich ursprünglich gegenüberliegenden Punkte gebildet werden.

$$x = \hat{b} - \hat{d} \quad (7.55)$$

$$y = \hat{c} - \hat{a} \quad (7.56)$$

$$x' = \hat{b}' - \hat{d}' \quad (7.57)$$

$$y' = \hat{c}' - \hat{a}' \quad (7.58)$$

$x$  und  $y$  sind Vektoren der euklidischen Bildebene. Die Rechtwinkligkeit beider wird also erhalten, wenn gilt:

$$(H_s x)^T (H_s y) = 0 \quad (7.59)$$

$$(H'_s x')^T (H'_s y') = 0 \quad (7.60)$$

Die Seitenverhältnisse der Bilder werden beibehalten, wenn gilt:

$$\frac{(H_s x)^T (H_s x)}{(H_s y)^T (H_s y)} = \frac{w^2}{h^2} \quad (7.61)$$

$$\frac{(H'_s x')^T (H'_s x')}{(H'_s y')^T (H'_s y')} = \frac{w'^2}{h'^2} \quad (7.62)$$

Für  $u_a, u'_a, u_b$  und  $u'_b$  jeweils Gleichungen auf Basis der jeweiligen Bild Höhen und Breiten  $w, w', h, h'$  und  $x, x', y$  und  $y'$  und unter einhaltung der Aussagen der Gleichungen 5.118 bis 5.121, aufgestellt werden[27, 35].

$$u_a = \frac{h^2 x_v^2 + w^2 + y_v^2}{h w (x_v y_u - x_u y_v)} \quad (7.63)$$

$$u_b = \frac{h^2 x_u x_v + w^2 y_u y_v}{h w (x_u y_v - x_v y_u)} \quad (7.64)$$

Selbe Gleichungen werden auch für  $u'_a$  und  $u'_b$  aufgestellt. Das Ergebnis der Scherungstransformation ist in Abbildung 5.17 dargestellt. Wie zu sehen ist, ist die Minimierung noch nicht zu hindert prozent perfekt, hierfür müsste man noch ein paar mehr Interationsschritte bei finden von  $z$  einfügen.(ICH WEIß GANZ EHRLICH NICHT WORAN ES LIEGT...)

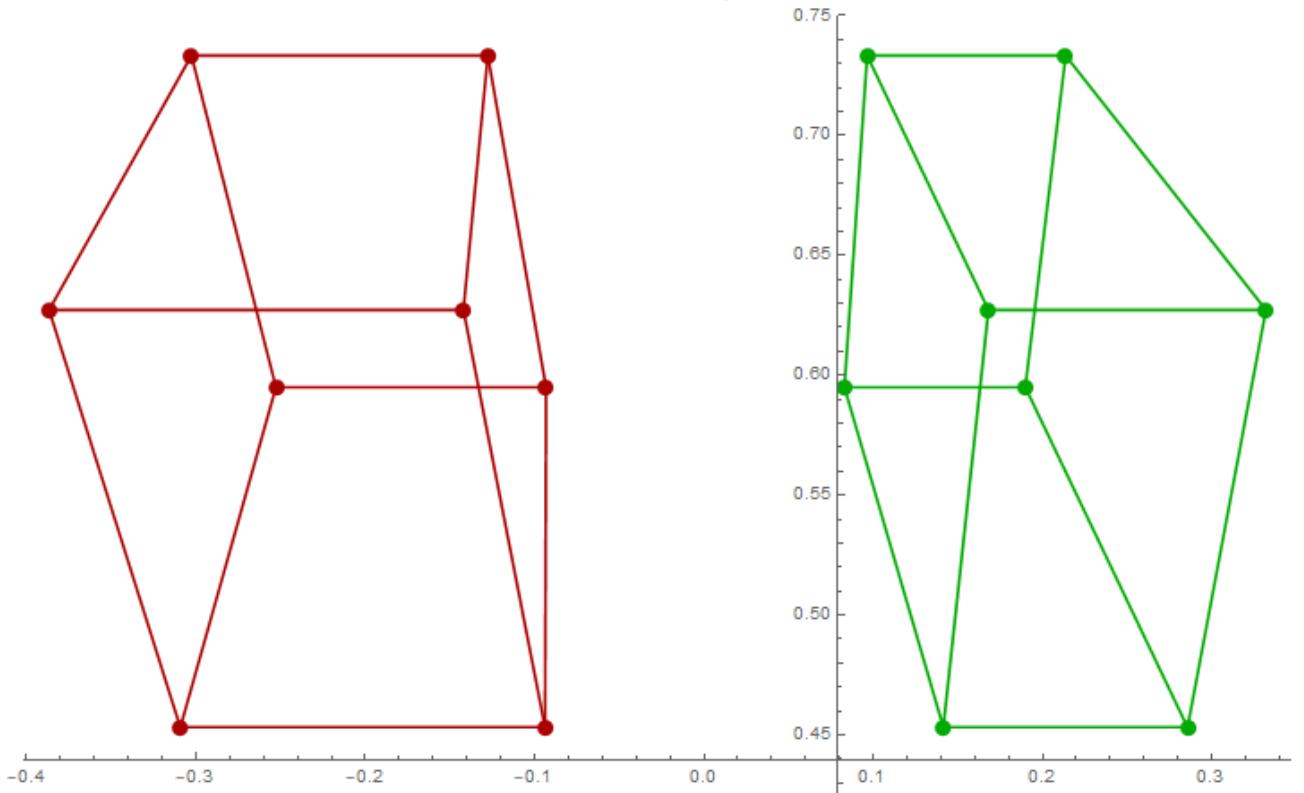


Abbildung 7.9: Abbildung beider Bilder nach anwenden der Matrizen  $H_s \cdot H_r \cdot H_p$  und  $H'_s \cdot H'_r \cdot H'_p$ . Die horizontale Verzerrung wurde reduziert.

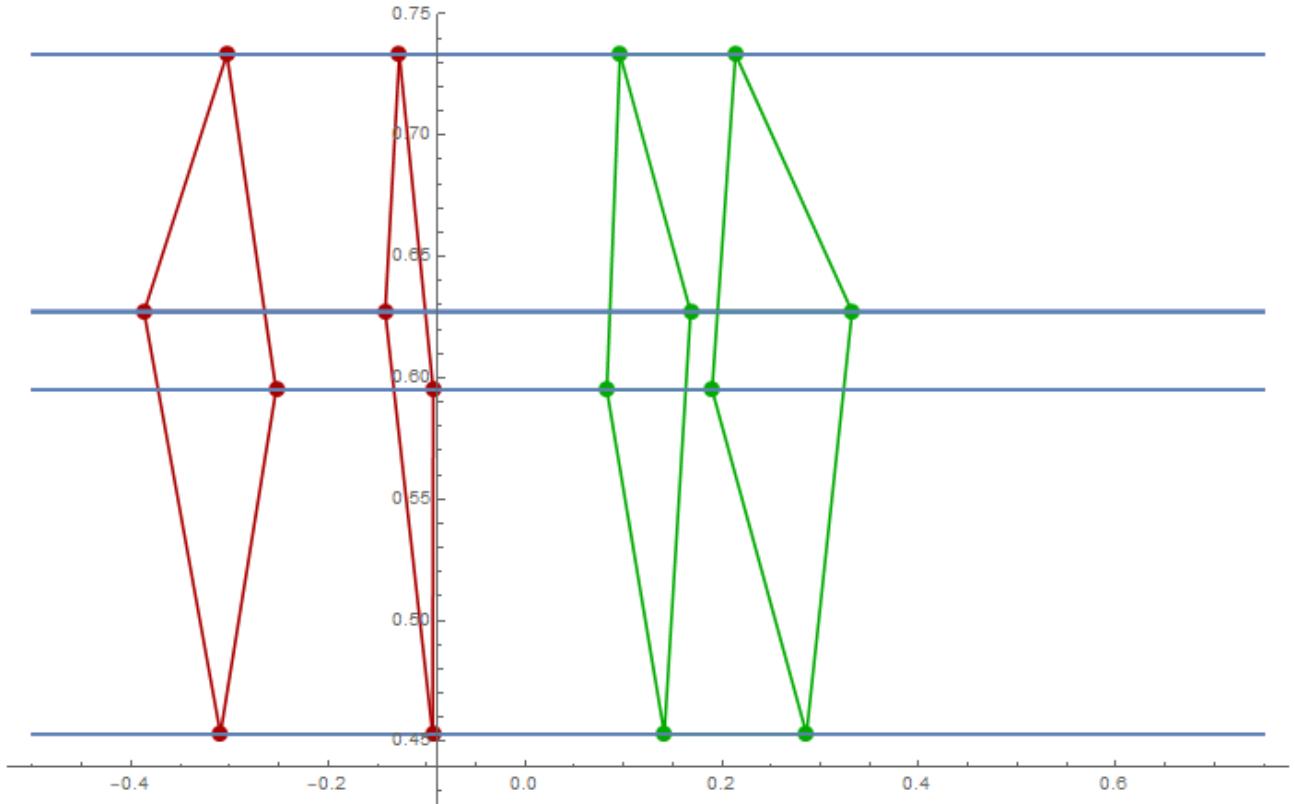


Abbildung 7.10: In dieser Abbildung wurden die Epipolarlinien noch in den Grafikplot mit eingebaut

## 8 Punktesortierung in Schachbrettmustern

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 Tonnennverzeichnungen 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.

Dieser Algorithmus nimmt die Liste mit den Koordinaten der Eckpunkte entgegen und sortiert und nummeriert diese Zeilen- und Spaltenweise durch.

Jeder Punkt ist somit über zwei Indizes codiert und enthält die Information, in welcher Zeile und in welcher Spalte des Schachbrettmusters er sich befindet.

Da nicht immer garantiert ist, dass alle Punkte innerhalb des Schachbretts zuvor gefunden worden, enthält der Sortierungsalgorithmus eine Funktion, in welchem er Lücken des innerhalb ausfindig macht und synthetische Eckpunkte setzt. Diese synthetisch gesetzten Punkte, werden markiert, so dass sie nicht in die Liste der möglichen korrespondierenden Punkte fallen.

## 8.1 Vorläufiges Klassendiagramm

Module	Parameter	Lokale Variablen	Funktion
FindMinMax	Pointlist	Imin, imax, jmin, jmax, iSplits, jSplits, iDistance, jDistance	<ul style="list-style-type: none"> <li>Die minimas und maximas der i und j-Werte der Koordinaten werden gesucht, um den „Rahmen“ des Gitters um das Schachbrett festzulegen</li> <li>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.</li> </ul>
SortPointList	iSplits, jSplits, Pointlist	pi,pj	<ul style="list-style-type: none"> <li>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</li> <li>Danach werden die Eckpunkte der Größe nach nach ihren J-Werten sortiert und bekommen hier ebenfalls einen Index zugeordnet</li> <li>(Diese Sortierung ist nach jetzigem Stand des Algorithmus vlt nicht mehr zwingend notwendig)</li> </ul>
GoThroughConvex Hulls	iSplits, jSplits, pj	ConvexHull	<ul style="list-style-type: none"> <li>Nun wird herausgefiltert, welcher Punkt in welche Zelle des erstellten Gitters gehört, somit wird eine grobe Vorsortierung der Punkte für den weiteren Verlauf vorgenommen.</li> <li>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.</li> <li>Der Funktion werden die momentanen iSplits der Durchzählung übergeben und alle Jsplits. Des Weiteren wird die nach J sortierte Punkteliste übergeben</li> </ul>
FindPointsInConvexHull	iSplits[[1,ii]], iSplits[[1,ii+1]], jSplits, pj	ConvexHullCell={}, ConvexHullList, ConvexHullCellKeys = <  >	<ul style="list-style-type: none"> <li>Eine Liste namens ConvexHullCell und eine Association nach dem ConvexHullKeys wird angelegt</li> <li>Zwei For-Schleifen werden gestartet. Die erste läuft durch alle Jsplits, die zweite geht alle Punkte von pj durch.</li> <li>Innerhalb der For-Schleife wird dann überprüft, welche Punkte aus pj sich innerhalb der übergebenen Jsplits und den dazugehörigen iSplits befinden.</li> <li>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 an die Liste ConvexHull angehängt</li> <li>Wiederholung des Vorganges mit neuen iSplits.</li> </ul>

Abbildung 8.1: Klassendiagramm

FindStartVectors	ConvexHull	StartPointCloud={}, StartPointCloudKeys=<  >, VecI,VecJ,countI,countJ, Start, nextI,nextJ,	<ul style="list-style-type: none"> <li>Die Punkte der Zellen (<math>i = 1, j = \text{All}</math>) und (<math>i = \text{all}, j = 1</math>) werden in eine neue Liste namens StartPointCloud gespeichert.</li> <li>Die Liste wird zweimal durchlaufen <ul style="list-style-type: none"> <li>Alle Punkte welche sich in den Zellen <math>j = 1</math> und <math>i = \text{all}</math> aufhalten. Aus ihnen wird der geringste i- Wert ermittelt (VecI)</li> <li>Alle Punkte welche sich in den Zellen <math>j = \text{all}</math> und <math>i = 1</math> aufhalten. Aus ihnen wird der geringste j- Wert ermittelt (VecJ)</li> </ul> </li> <li>Die Punkte mit den geringsten Werten werden in VecI und VecJ gespeichert.</li> <li>Jetzt wird die Liste nochmals zweimal durchgegangen. <ul style="list-style-type: none"> <li>Alle Punkte welche sich in den Zellen <math>j = 1</math> und <math>i = \text{all}</math> aufhalten werden durchgegangen. Aus ihnen wird derjenige Wert ermittelt, welcher einen Wert für J besitzt der kleiner ist als der momentan 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</li> <li>Alle Punkte welche sich in den Zellen <math>j = \text{all}</math> und <math>i = 1</math> aufhalten. Aus ihnen wird derjenige Wert ermittelt, welcher einen Wert für i besitzt der kleiner ist als der momentan i- Wert von VecJ und dessen j- Wert kleiner ist als der j- Wert von VecI plus einem Offset. Dieser Wert ist das neue VecJ</li> </ul> </li> <li>VecI und VecJ ergeben den gleichen Punkt und somit ist der Startwert gesetzt.</li> <li>Nun sollen die ersten Punkte in i- und j-Richtung vom Startpunkt aus gefunden werden.</li> <li>Es wird ein nexti und ein nextj definiert, dessen Koordinaten sehr groß anfangen</li> <li>Es wird wieder die StartPointListe zweimal durchlaufen <ul style="list-style-type: none"> <li>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 der momentane nexti und größer als der Startpunkt, jedoch nicht gleich dem Startpunkt. So nimmt nexti dessen Wert an.</li> <li>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.</li> </ul> </li> </ul>
------------------	------------	--	--

Abbildung 8.2: Klassendiagramm

			<ul style="list-style-type: none"> <li>Ist dies der Fall so wird dieser Punkt zum neuen nexti.</li> <li>Mit dem potentiellen nextj wird ebenso verfahren.</li> </ul>
CreatePossiblePoint - ListsIAndJ	nextI, nextJ, Start, ConvexHull	IList= {}, JList= {}, IDir, JDir, distance, cache, PotNextI, PotNextJ	<ul style="list-style-type: none"> <li>IDir und JDir sind die Richtungsvektoren vom Startpunkt aus in beide Kantenrichtungen des Schachbretts.</li> <li>Danach werden die ersten beiden Spalten in I- und J-Richtung jeweils durchlaufen, und in IList und JList gespeichert.</li> <li>Diese Listen enthalten weitere potentielle Punkte entlang der gesuchten Kante.</li> <li>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</li> </ul>
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"> <li>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).</li> <li>Diese drei bereits bekannten Punkte werden dann auch in eine angelegte CheckPointList gespeichert, diese wird für das spätere Prüfen von weiteren Punkten benötigt.</li> <li>Nun wird zunächst in einer For-Schleife die Punkte von startPoint und NextPointJ aus gesucht. <ul style="list-style-type: none"> <li>Anmerkung: Für die Punkte in I-Richtung des Schachbretts wird das selbe Verfahren angewandt.</li> </ul> </li> <li>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.</li> <li>Die Schleife selbst durchläuft alle Punkte, welche in der für die Richtung entsprechenden Richtung Liste sind. In diesem Fall die JList</li> <li>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.</li> <li>Innerhalb der durchlaufenden Liste wird derjenige Punkt gesucht welcher zum NextPointJ den geringsten Abstand in J-Richtung hat und dessen Abstand in I-Richtung &lt;= der i-Koordinate des NextPointJ + proportion+noch einen Puffer ist und &gt;= der i-Koordinate des NextPointJ – proportion+noch einen Puffer.</li> </ul>

Abbildung 8.3: Klassendiagramm

			<ul style="list-style-type: none"> <li>Ist der nächste Punkt gefunden, so wird dieser der SortedPointsList und der der CheckPointsList übergeben mit den passenden NeighbourI und NeighbourJ associationKey.</li> <li>Des Weiteren bekommt für den nächsten Schleifendurchlauf startPointJ die Werte von NextPointJ und NextPointJ' in wird der neu gefundenen Punkt aus der JList gespeichert.</li> <li>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.</li> <li>Nicht zu vergessen: selbiges wie oben wird auch mit den Punkten in I-Richtung vollzogen, bis auf die CompleteGrid Funktion</li> </ul>
SaftyList	Start, CheckLastPointJ , proportionJ, LastJPointsCell, ConvexHull, NextJDir	SaftyList = {}, SaftyKeys =<>, SaftyKeysList = {}, propJ, lastDir, lastdistanceJ	<ul style="list-style-type: none"> <li>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.</li> <li>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</li> <li>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.</li> <li>Sollte es noch einen geben wird dieser ebenfalls der CheckPointList und der SortedPointsList zugewiesen, ansonsten passiert nichts.</li> </ul>
CompleteJGrid	StartPointI, ConvexHull, StartDistanceJ, proportionJ, Start,	PossiblePointsList = {}, SortedPointsKeys = <>, SaftyPossiblePointsListJ = {}, propJ, StartPointForJGrid, distanceJ,	<ul style="list-style-type: none"> <li>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.</li> <li>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</li> </ul>

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

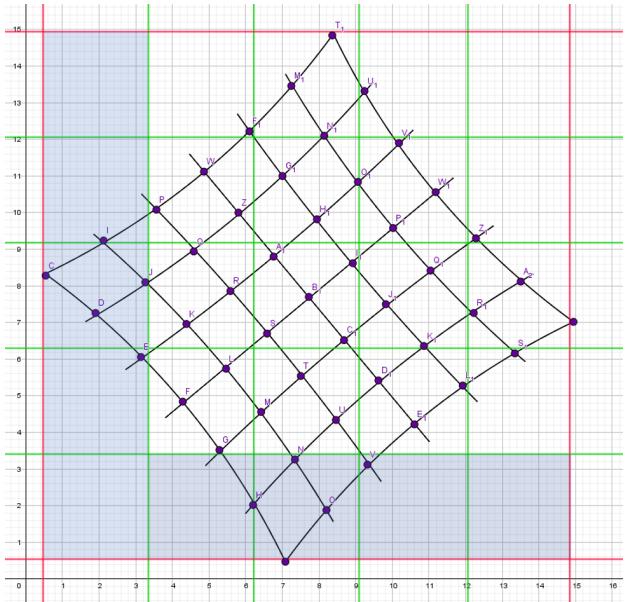


Abbildung 8.6

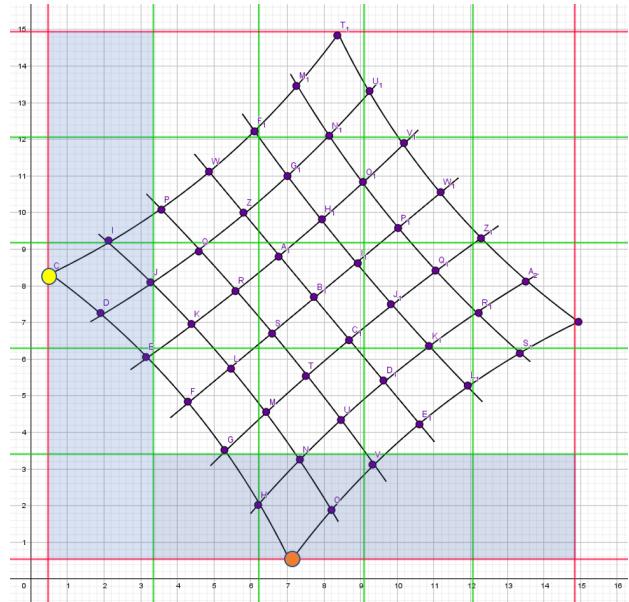


Abbildung 8.7

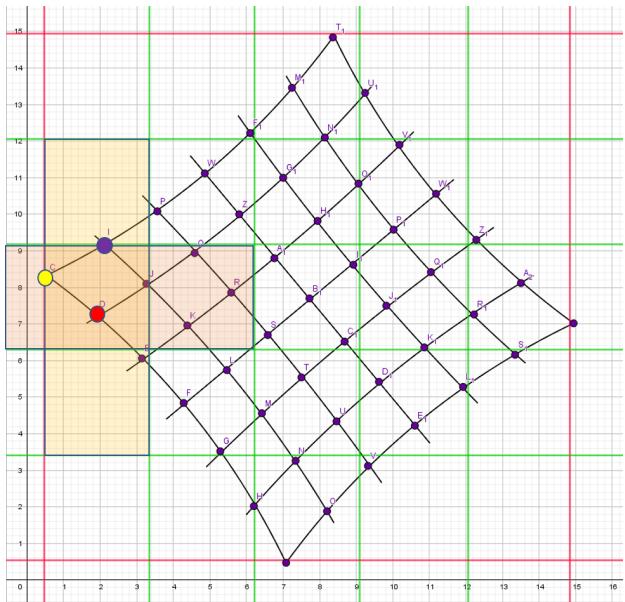


Abbildung 8.8

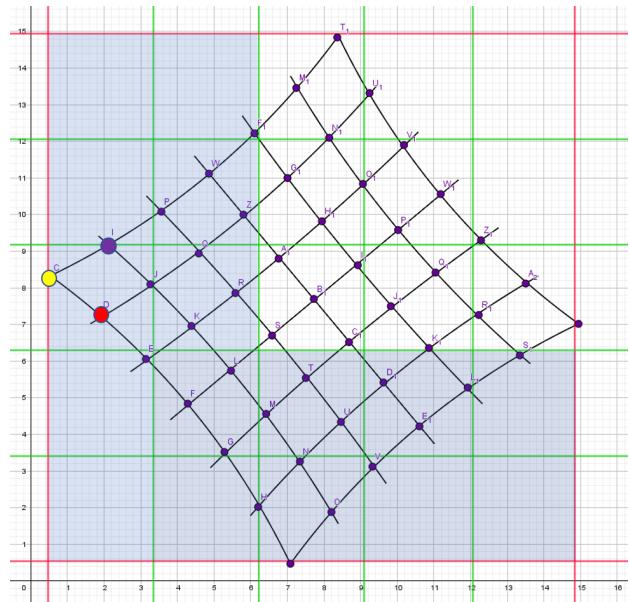


Abbildung 8.9

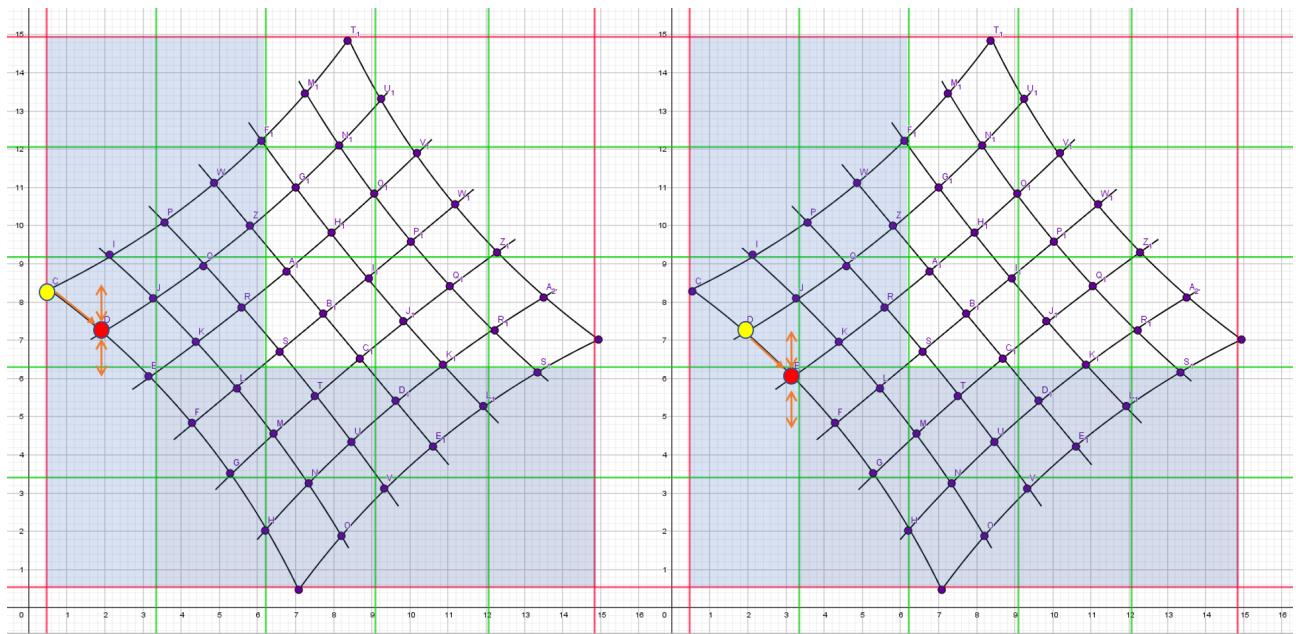


Abbildung 8.10: Klassendiagramm

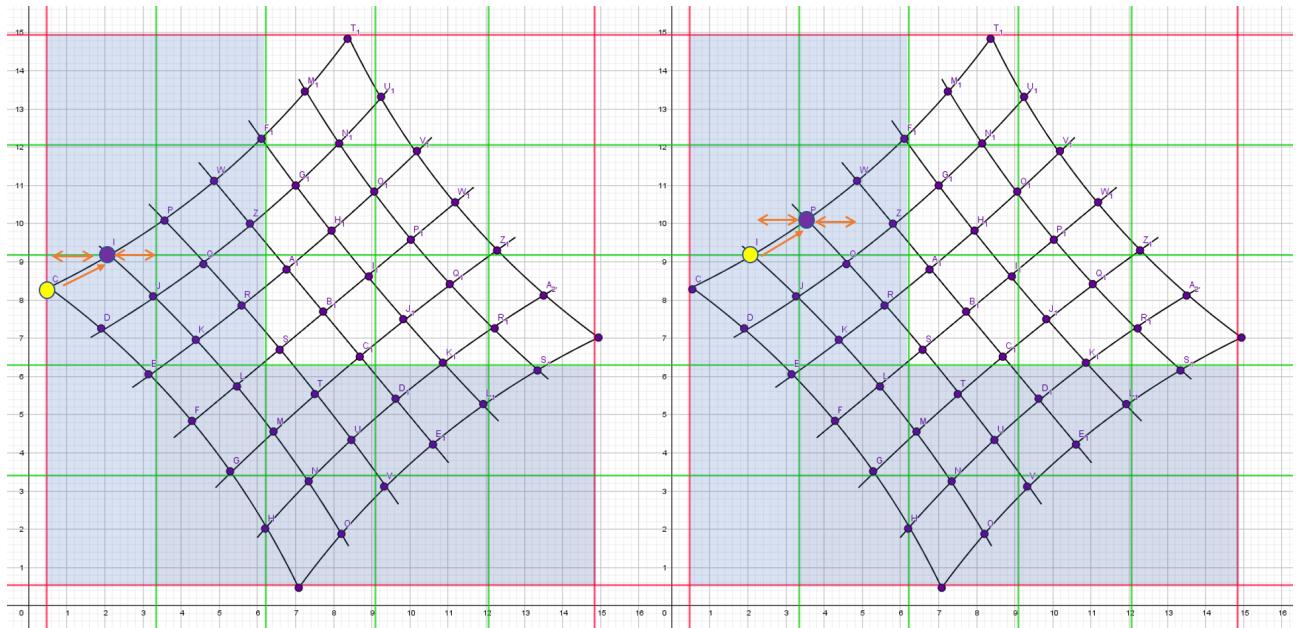


Abbildung 8.11: Klassendiagramm

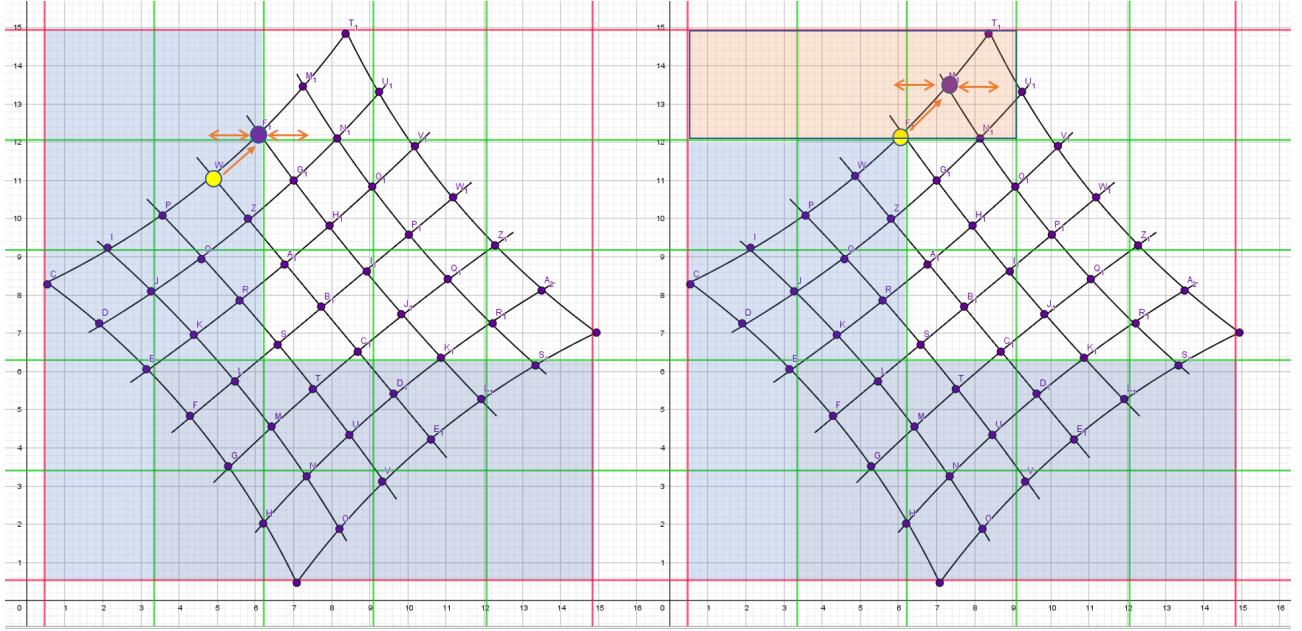


Abbildung 8.12: Klassendiagramm

## 8.2 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.

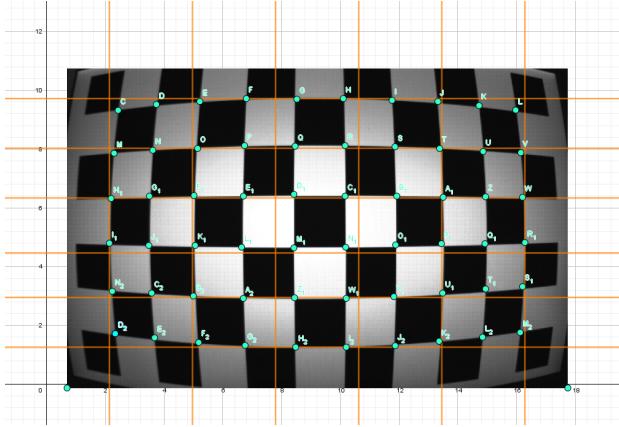


Abbildung 8.13: Bild eines Tonnenförmig verzeichneten Schachbretts

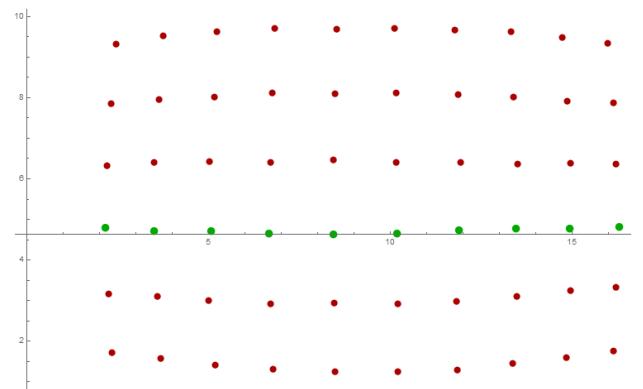


Abbildung 8.14: Algorithmisch detektierte Linie der dritten i-Reihe

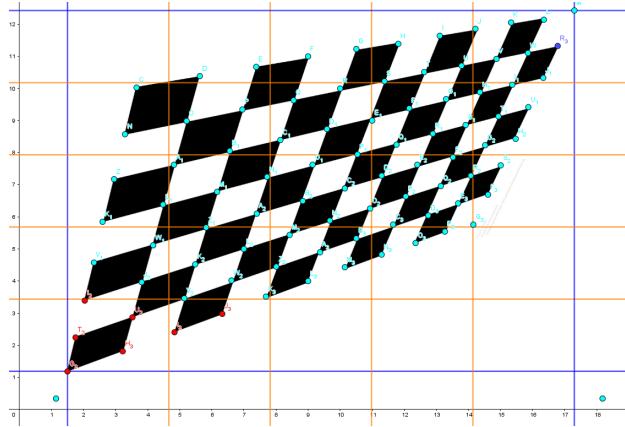


Abbildung 8.15: Bild eines perspektivisch verzerrtem Schachbretts

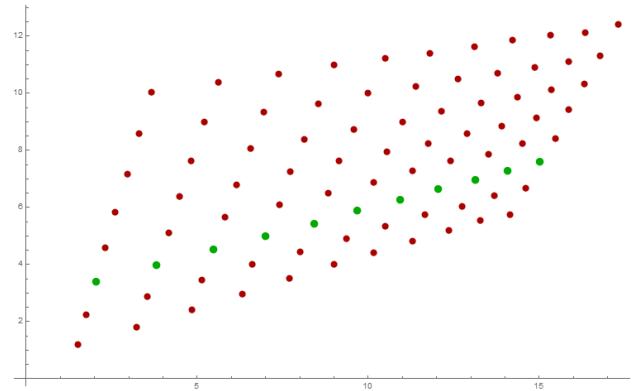


Abbildung 8.16: Algorithmisch detektierte Linie der dritten i-Reihe

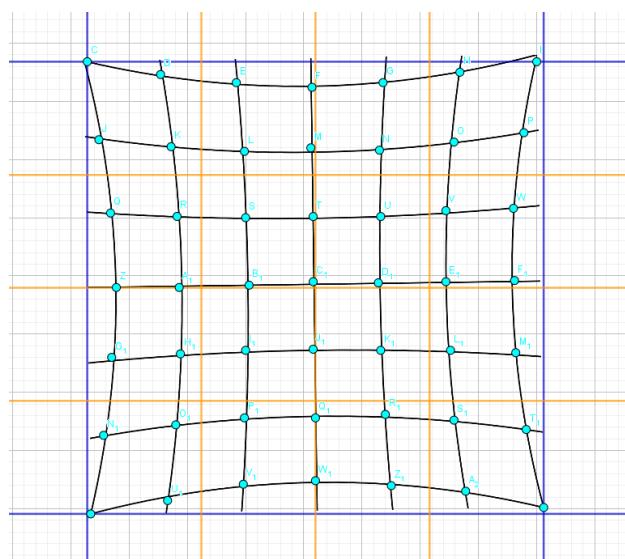


Abbildung 8.17: Bild eines Kissenförmig verzeichnetem Schachbretts

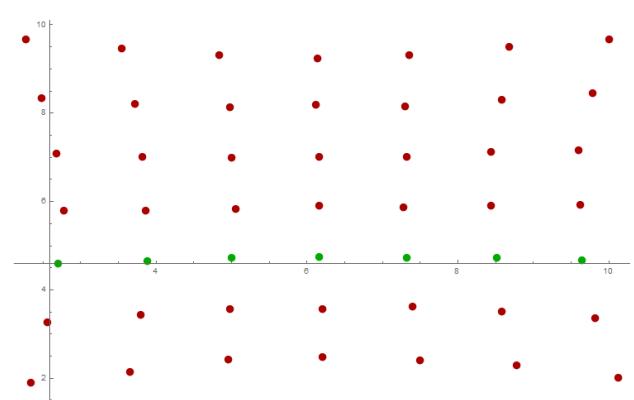


Abbildung 8.18: Algorithmisch detektierte Linie der dritten i-Reihe

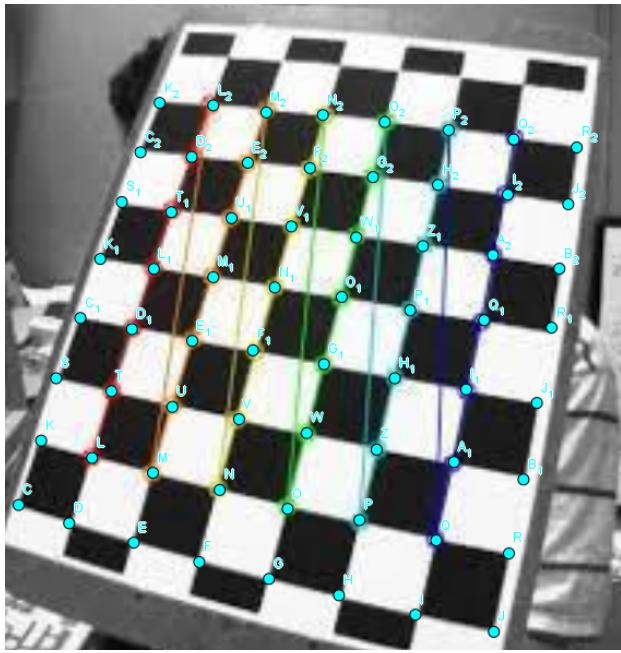


Abbildung 8.19: Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts(GRFIK AUSTAUSCHEN BILD IS KACKE)

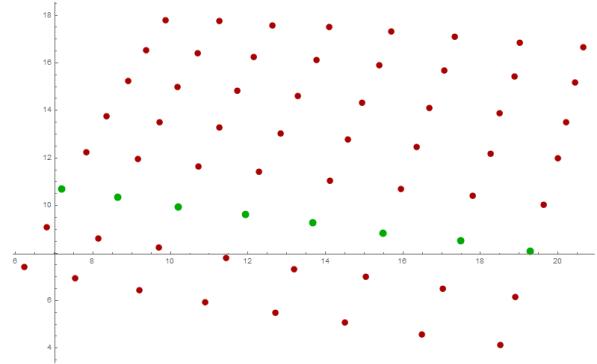


Abbildung 8.20: Algorithmisch detektierte Linie der dritten i-Reihe

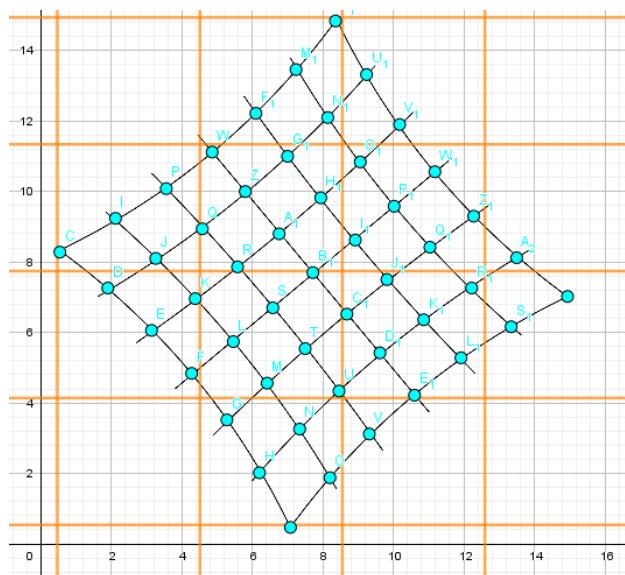


Abbildung 8.21: Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts

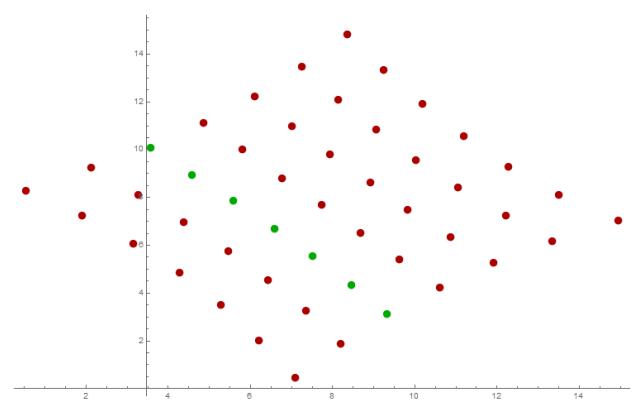


Abbildung 8.22: Algorithmisch detektierte Linie der dritten i-Reihe

## **9 Fazit - Conclusion**

## **10 Alternativen**

## **11 Abkürzungsverzeichnis - List of Abbreviations**

# Abbildungsverzeichnis

2.1	Schematik eines abbildenden Systems. Ein Punkt $M$ im Weltkoordinatensystem $O$ wird durch eine Kamera $C$ aufgenommen. Diese Aufnahme wird durch eine Projektion, die als Verbindungsgerade von $M$ zu $C$ zu sehen ist dargestellt ist und $M$ auf $m$ abbildet, beschrieben. . . . .	5
2.2	Die Abbildung zeigt einen Querschnitt des beschriebenen Lochkameramodells. Zu sehen ist das Projektionszentrum $C$ der Kamera. $C$ ist gleichzeitig das Kamerazentrum und bildet den Ursprung für das Kamerakoordinatensystem. $\zeta$ beschreibt den Abstand des Projektionszentrums zur Bildebene. Die Hauptachse beschreibt die Blickrichtung der Kamera. Der Punkt an dem die Hauptachse die Bildebene schneidet wird Hauptpunkt genannt und ist gleichzeitig der Ursprung für das Bildebenenkoordinatensystem. Der Bildpunkt $m$ entsteht am Schnittpunkt der Verbindungsgeraden von $C$ und $M$ mit der Bildebene $I$ . . . . .	6
2.3	Ein Weltkoordinatensystem $(O, \delta)$ mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O)$ wird zu einem dazu verschobenen und rotiertem Kamerakoordinatensystem $(C, \beta)$ mit $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3, C)$ transformiert	7
2.4	Das Schaubild zeigt die einzelnen Koordinatensysteme in einem Lochkameramodell. Das Weltkoordinatensystem $(O, \delta)$ mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3)$ , das Kamerakoordinatensystem $(C, \beta)$ mit $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3)$ , das Bildebenenkoordinatensystem $(I, \tau)$ mit $\tau = (\hat{t}_1, \hat{t}_2)$ und das Sensorkoordinatensystem $(S, \sigma)$ mit $\sigma = (\hat{u}, \hat{v})$ . . . . .	9
3.1	In der Abbildung sind die beiden Kameras $C$ und $C'$ mit ihren Bildebenen $I$ und $I'$ zu sehen. Ein Objektpunkt $M_\delta$ bezüglich eines Weltkoordinatensystems $(O, \delta)$ befindet sich auf der Ebene $\pi$ , welche durch die Achsen $\hat{d}_1$ und $\hat{d}_2$ aufgespannt wird. $M_\delta$ wird auf $I$ und $I'$ projiziert. Es entstehen die Bildpunkte $m_\tau$ und $m'_\tau$ . . . . .	13
3.2	Auf der Geraden durch $C$ und $\vec{m}_\tau$ , befinden sich alle möglichen Punkte für $m_\beta$ . $m_\beta$ wird auf Grund seiner Unbestimmtheit als $\gamma m_\tau$ bezeichnet . . . . .	13
3.3	$C$ und $C'$ sind die Projektionszentren zweier Kameras. Beide Kameras besitzen jeweils eine Bildebene. Die Basislinie verbindet die Projektionszentren der Kameras. Die Punkte an welchen die Basislinie die Bildebenen schneidet, werden als Epipole $e$ und $e'$ bezeichnet. Durch einen Epipol verlaufen alle Epipolarlinien des Bildes. $M_\delta$ ist der Objektpunkt im 3D-Raum und $m_\tau$ und $m'_{\tau'}$ sind die jeweiligen Abbildungen dieses Punktes auf den Bildebenen. Die Verbindungsvektoren zwischen $C, C'$ und $M_\delta$ bilden die sogenannte Epipolarebene[10, 11, 3, 1]. . . . .	15
3.4	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. . . . .	15
4.1	Ablaufdiagramm für das synthetische Beispiel . . . . .	21
4.2	In der Abbildung ist der vereinfachte Stereoaufbau in einer Top-Down-Ansicht zu sehen	22
4.3	Simulierte Abbildung des Quaders auf die Kamera $C$ in Grün und auf $C'$ in rot . . . . .	22
4.4	Abbildung der verschiedenen Kamerakoordinatensysteme, das Weltkoordinatensystem $O$ ist zum Kamerakoordinatensystem $C$ deckungsgleich und $C'$ verschoben und rotiert.	22
4.5	a) . . . . .	26
4.6	b) . . . . .	26
4.7	c) . . . . .	26
4.8	d) . . . . .	26
4.9	Die Abbildungen a, b, c und d veranschaulichen, welche Bilder aus den vier Lösungen entstehen. In den Abbildungen a und b kommt es zu einer Umkehrung der Basisline. In den Abbildungen c und d wird $C'$ um $180^\circ$ gedreht . . . . .	26

4.10	Optimale Triangulierung: Beide Geraden Treffen sich in einem Punkt im 3D-Raum . . . . .	27
4.11	. . . . .	28
4.12	. . . . .	28
4.13	Veranschaulichung der Skaleninvarianz und dessen Auswirkung auf die geometrische Form und Größe der Objekte . . . . .	29
4.14	. . . . .	29
4.15	. . . . .	29
4.16	Der rote Punkt stellt Die Postion von $C$ dar, der grüne steht für die Position von $C'$ relativ zu $C$ . Die blauen Punkte stellen den rekonstruierten Quader und den extern platzierten neunten Punkt da. Das Abbild entstand aus dem in <i>Mathematica</i> implementierten Algorithmus. . . . .	29
5.1	Rechteckiger Bildsensor mit darauf sich befindendenden quadratischen Sensorelementen. Vergleiche [8] . . . . .	30
5.2	Bild a) zeigt die den Zusammenschluss mehrerer benachbarter Pixel zu einem neuen Pixel. Bild b) zeigt in gelb markiert, den aktiven lichtempfindlichen Bereich des Sensors, wenn sich das Seitenverhältnis geändert wird und nicht mehr der komplette Sensor genutzt wird. . . . .	31
5.3	$C$ und $C'$ haben die selbe Auflösung eingestellt . . . . .	32
5.4	$C$ und $C'$ haben unterschiedliche Auflösungen eingestellt . . . . .	32
5.5	$C$ und $C'$ haben die selbe Auflösung eingestellt . . . . .	33
5.6	$C$ und $C'$ haben unterschiedliche Auflösungen eingestellt. $C$ mit $K$ und $C'$ mit $K'_1$ . . . . .	33
5.7	$C$ mit $K$ und $C'$ mit $K'_2$ . . . . .	33
5.8	$C$ mit $K$ und $C'$ mit $K'_3$ . . . . .	33
5.9	Die rekonstruierten Szenenpunkte und Kamerapositionen bleibt auch bei unterschiedlichen Auflösungen die selben . . . . .	35
6.1	Ablaufdiagramm für die reelle Rekonstruktion . . . . .	36
6.2	Szenenaufbau: Die Canon 60D befindet sich in dieser Abbildung auf der linken Seite, die Canon 60 D auf der rechten. Auf dem Tisch zwischen den Kameras ist die in den Abbildungen 6.1 und 6.1 abgebildete Szene zu sehen. Beide Kameras sind zu Szene hin gedreht. . . . .	37
6.3	a . . . . .	38
6.4	b . . . . .	38
6.5	Epipolarlinien ohne <i>Epipolar-constraint</i> im Bild der Canon 6D . . . . .	39
6.6	Epipolarlinien ohne <i>Epipolar-constraint</i> im Bild der Canon 60D . . . . .	39
6.7	Epipolarlinien mit <i>Epipolar-constraint</i> im Bild der Canon 6D . . . . .	40
6.8	Epipolarlinien ohne <i>Epipolar-constraint</i> im Bild der Canon 60D . . . . .	40
6.9	Epipolarlinien mit <i>Epipolar-constraint</i> im Bild der Canon 6D, nach der denormalisierung von $F$ . . . . .	40
6.10	Epipolarlinien ohne <i>Epipolar-constraint</i> im Bild der Canon 60D, nach der denormalisierung von $F$ . . . . .	40
6.11	a) . . . . .	41
6.12	b) . . . . .	41
6.13	a) Die rückprojizierten Strahlen der ungenauen korrespondierenden Punkte $m$ und $m'$ sind schief und treffen sich nicht in einem Punkt im 3D-Raum. b) The epipolar geometry for $m$ , $m'$ . The measured points do not satisfy the epipolar constraint. The epipolar line $l' = Fm$ is the image of the ray through $n$ , and $l = F^T m'$ is the image of the ray through $m'$ . Since the rays do not intersect, $m'$ does not lie on $l'$ , and $m$ does not lie on $l$ . . . . .	41
6.14	Frafische Darstellung der optimalen Punkte $\hat{m}$ und $\hat{m}'$ . . . . .	42
6.15	Rekonstruierte Szene, unskaliert in Pixeleinheiten . . . . .	47
6.16	Rekonstruierte Szene, unskaliert, in Pixeleinheiten und in einem 2D-Plot geschrieben .	47
6.17	Zeigt die Die rekonstruierte Matrix $R'$ bei unveränderter Auflösung. Die Auflösungen von $C_\delta$ und $C'_\delta$ sind die selben. . . . .	48

6.18 Zeigt die rekonstruierte Matrix $R'$ wenn $K'$ mit einem Verhältnis von [5 : 2] skaliert wurde . . . . .	48
6.19 Zeigt die rekonstruierte Matrix $R'$ wenn $K'$ mit einem Verhältnis von [1 : 2] skaliert wurde . . . . .	48
6.20 Zeigt die rekonstruierte Matrix $R'$ wenn $K'$ mit einem Verhältnis von [1.2 : 2.3] skaliert wurde . . . . .	48
6.21 Rekonstruierte Szene, wenn $K'$ mit einem Verhältnis von [5 : 2] skaliert wurde . . . . .	49
6.22 Rekonstruierte Szene, wenn $K'$ mit einem Verhältnis von [2 : 1] skaliert wurde . . . . .	49
6.23 Rekonstruierte Szene, wenn $K'$ mit einem Verhältnis von [1 : 2] skaliert wurde . . . . .	50
6.24 Rekonstruierte Szene, wenn $K'$ mit einem Verhältnis von [1.2 : 2.3] skaliert wurde . . . . .	50
 7.1 (Überarbeiten sie Abb 4.1) Durch Ungenauigkeiten in der korrespondierenden Punkte, verfehlten sich die Linien und es kommt zu keinem Schnittpunkt . . . . .	51
7.2 Beispiel eines rektifizierten Bildes. Quelle: [27] . . . . .	52
7.3 Beispiel einer einfachen Tiefenkarte eines Stereobildpaars nach der Rektifizierung. Quelle: [28] . . . . .	52
7.4 Aufnahmen zweier Kamerae mit den selben Auflösungen, Kamera eins(Grün) und Kamera(rot) zwei gelten jeweils $\zeta = 1$ . . . . .	59
7.5 Epipole für Kamera eins und Kamera zwei vor der Rektifizierung . . . . .	59
7.6 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 $i = [1,0,0]$ . . . . .	60
7.7 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 . . . . .	62
7.8 Die Abbildung verdeutlicht noch mal Schematisch, wie sich die Punkte ausrichten sollen. Bild a) zeigt die durch die Rektifizierung verschobenen Bildkantenmitten. Bild b= zeigt, wie sich die Bildkantenmitten durch die Scherungstransformation wieder ausrichten sollen. . . . .	63
7.9 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. . . . .	64
7.10 In dieser Abbildung wurden die Epipolarlinien noch in den Grafikplot mit eingebaut . . . . .	65
 8.1 Klassendiagramm . . . . .	67
8.2 Klassendiagramm . . . . .	67
8.3 Klassendiagramm . . . . .	68
8.4 Klassendiagramm . . . . .	68
8.5 Klassendiagramm . . . . .	68
8.6 . . . . .	69
8.7 . . . . .	69
8.8 . . . . .	69
8.9 . . . . .	69
8.10 Klassendiagramm . . . . .	70
8.11 Klassendiagramm . . . . .	70
8.12 Klassendiagramm . . . . .	71
8.13 Bild eines Tonnenförmig verzeichneten Schachbretts . . . . .	71
8.14 Algorithmisch detektierte Linie der dritten i-Reihe . . . . .	71
8.15 Bild eines perspektivisch verzerrtem Schachbretts . . . . .	72
8.16 Algorithmisch detektierte Linie der dritten i-Reihe . . . . .	72
8.17 Bild eines Kissenförmig verzeichnetem Schachbretts . . . . .	72
8.18 Algorithmisch detektierte Linie der dritten i-Reihe . . . . .	72
8.19 Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts(GRFIK AUSTAUSCHEN BILD IS KACKE) . . . . .	73
8.20 Algorithmisch detektierte Linie der dritten i-Reihe . . . . .	73

8.21 Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts . . . . .	73
8.22 Algorithmisch detektierte Linie der dritten i-Reihe . . . . .	73

## **Tabellenverzeichnis**

# Literaturverzeichnis

- [1] Zhengyou Zhang Gang Xu. *Epipolar Geometry in Stereo, Motion and Object Recognition: A Unified Approach*. Springer-Science and Business Media, 1996.
- [2] Lutz Priese. *Computer Vision, Einführung in die Verarbeitung und Analyse digitaler Bilder*. Springer-Verlag Berlin Heidelberg, 2014.
- [3] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge, 2004, Second Edition.
- [4] Ferid Bajrmovic. *Self- Calibration of Multi- Camera Systems*. Logos Verlag Berlin GmbH, 2010.
- [5] Tomas Pajdla. *Elements of Geometry for Computer Vision*. "<http://people.ciirc.cvut.cz/pajdla/>", 2013, überarbeitet am 27.2.2017.
- [6] Christian Heipke. *Photogrammetrie und Fernerkundung*. 2017 Springer, 1. Auflage.
- [7] Ramalingam Tardif S.Gasparini J.Barreto R.Sturm, S. Camera models and fundamental concepts used in geometric computer vision. Mitsubishi Electric Research Laboratories, 2011.
- [8] Rolf Martin Ekbert Hering. *Photonik, Grundlagen, Technologien und Anwendung*. Springer-Verlag Berlin Heidelberg, 2006.
- [9] 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.
- [10] Christoph Stiller Thao Dang, Christian Hoffmann. Continuous stereo self-calibration by camera parameter tracking.
- [11] Branislav Micusik. *Two-View Geometry of Omnidirectional Cameras*. Dissertation, Technische Universität Prag.
- [12] Zhengyou Zhang. *Epipolar Geometry*, pages 247–258. Springer US, Boston, MA, 2014.
- [13] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. Received July 16, 1996; Accepted February 13, 1997.
- [14] K.A. Semendjajew I.N. Bronstein. *Taschenbuch der Mathematik*, volume 5. Auflage. "<http://doi.org/10.1002/bimj.19640060108>", First Published 1962.
- [15] Sascha jockel. *3-dimensionale Rekonstruktion einer Tischszene aus monokularen Handkamera-Bildsequenzen im Kontext automotiver Serviceroboter*. Dissertation, Fakultät für Mathematik, Informatik und Naturwissenschaften, Universität Hamburg.
- [16] Richard I. Hartley. In defence of the 8-point algorithm. GE-Corporate Research and Development, Schenectady, NY, 12309.
- [17] Herbert Bay, Tinne Tuytelaars, and Luc Van Gool. Surf: Speeded up robust features. In Aleš Leonardis, Horst Bischof, and Axel Pinz, editors, *Computer Vision – ECCV 2006*, pages 404–417, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.
- [18] Norbert Köckler Hans Rudolf Schwarz. *Numerische Mathematik*. 2011, Springer Verlag, 8. Auflage.

- [19] Daniel Scholz. *Numerik interaktiv, Grundlagen verstehen, Modelle erforschen und Verfahren anwenden mit taramath*. 2016, Springer Verlag.
- [20] LongQuan. *Image Based Modeling*, volume 1. Auflage. Springer US, 2010.
- [21] Olaf Dössel. *Bildgebende Verfahren in der Medizin*, volume 2. Auflage. Springer-Verlag Berlin Heidelberg, 2016.
- [22] Wolfram Research, Inc. Mathematica, Version 11.1.1. Champaign, IL, 2018.
- [23] Emmanuel Habets Nicolas Paparoditis Xiaozhi Qu, Bahman Soheilian. Evaluation of sift and surf for vision based localization. *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, Volume XLI-B3, 2016.
- [24] Antin van den Hengel Darren Gawey Wojciech Chojnacki, Michael J. Brooks. Revisiting hartley's normalized eight-point-algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Volume: 25, 2003.
- [25] Jarosław Bylina Anna Pyzara, Beata Bylina. The influence of a matrix condition number on iterative methods convergence. *Proceedings of the Federated Conference on Computer Science and Information Systems*, pp. 459–464, 2011.
- [26] MathWorks. Mathworks documentation, rectify stereo images.
- [27] 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.
- [28] 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.
- [29] Luca Irsara Andrea Fusiello. Euclidean epipolar rectification of uncalibrated images. Eurac researc, IT.
- [30] Richard I. Hartley. Euclidean reconstruction from uncalibrated views. G.E. CRD, Schenectady, NY, 12301.
- [31]
- [32] Dongqing Li, editor. *Encyclopedia of Microfluidics and Nanofluidics*, pages 999–999. Springer US, Boston, MA, 2008.
- [33] S. Margulies. Fitting experimental data using the method of least squares. Department of Physics, University of Illinois at Chicago Circle, Chicago, Illinois 60680, 1967.
- [34] 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.
- [35] James Demmel Dinesh Manocha. Algorithms for interesting parametric and algebraic curves 1: Simple intersections. *ACM Transactions of Graphics*:73–100, 1994.