
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

1 Einleitung	4
2 Model der Bildaufnahme mit einer Kamera	5
2.1 Lochkameramodell zur Abbildung eines Punktes auf die Bildebene	5
2.2 Koordinatentransformation	6
2.3 Aufname mit einer willkürlichen Kameraorientierung	9
3 Geometrische Beziehungen zwischen Punktekorrespondenzen	12
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
4 Synthetische Rekonstruktion	21
4.1 Simulierte Bildaufnahme einer virtuellen Szene	22
4.2 Bildanalyse	24
4.2.1 Bestimmung der extrinsischen Kameraparameter	24
4.2.2 Szenenrekonstruktion durch Triangulation	27
4.3 Auswirkungen von unterschiedlichen Kameraauflösungen	29
4.4 Geometrie eines Sensors	30
4.5 Auswirkungen auf die Szenenrekonstruktion	31
5 Reelle Rekonstruktion	36
5.1 Stereoaufbau	36
5.2 Korrespondenzanalyse	37
5.3 Normierter acht-Punkt-Algorithmus	38
5.3.1 Singularität der Fundamentalmatrix	39
5.3.2 Singulärwerte der essentiellen Matrix	41
5.4 Szenenrekonstruktion mit Sampson-Approximation	42
5.5 Ergebnisse einer Stereoanalyse mit Kameras unterschiedlicher Auflösung	47
6 Vergleich mit anderen Computer Vision Applikationen	52
6.1 Szenenrekonstruktion mit Rektifizierung	52
6.2 Rektifizierung mit Homographien	54
6.2.1 Projektive Transformation	56
6.2.2 Ähnlichkeitstransformation	60
6.2.3 Scherungstransformation	62
6.3 Rektifizierung mit unterschiedlichen Kameraauflösungen	64
7 Punktesortierung in Schachbrettmustern	67
7.1 Sortierungsalgorithmus	67
7.2 Extrembeispiele	73
7.3 Modulübersicht	75
8 Fazit - Conclusion	78
Abbildungsverzeichnis	79
Literaturverzeichnis	83

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.

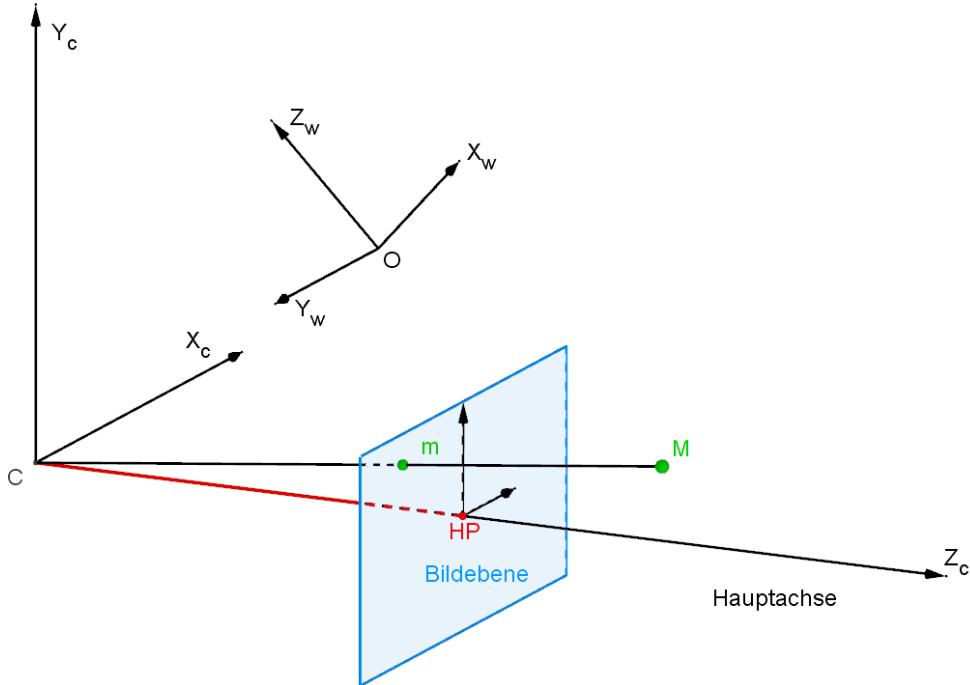


Abbildung 2.1: Schematik eines abbildenden Systems. Ein Punkt M im Weltkoordinatensystem O wird durch eine Kamera C aufgenommen. Diese Aufnahme wird durch eine Projektion, die als Verbindungslinie von M zu C zu sehen ist und M auf m abbildet, beschrieben.

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 ζ beschrieben[3, 7]. Der Bildpunkt m entsteht am Schnittpunkt der Verbindungsgerade von C und M mit der der Bildebene I .

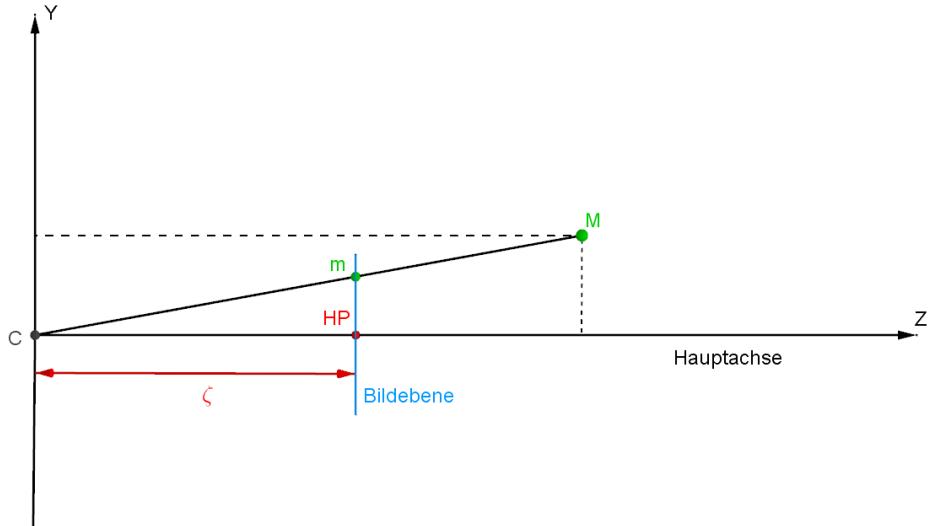


Abbildung 2.2: Die Abbildung zeigt einen Querschnitt des beschriebenen Lochkameramodells. Zu sehen ist das Projektionszentrum C der Kamera. C ist gleichzeitig das Kamerazentrum und bildet den Ursprung für das Kamerakoordinatensystem. ζ beschreibt den Abstand des Projektionszentrums zur Bildebene. Die 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 3×3 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, δ) mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O)$ in ein Kamerakoordinatensystem (C, β) mit $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3, C)$ beschrieben.



Abbildung 2.3: Ein Weltkoordinatensystem (O, δ) mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O)$ wird zu einem dazu verschobenen und rotiertem Kamerakoordinatensystem (C, β) 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_δ 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, δ) und (C, β) 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_β gegeben ist, erhält man auf diese Weise direkt P_δ . Die inverse Matrix D_β^{-1} kann verwendet werden um P_β aus P_δ 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 3×4 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, δ) mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3)$, das Kamerakoordinatensystem (C, β) mit $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3)$, das Bildeckenkoordinatensystem (I, τ) mit $\tau = (\hat{t}_1, \hat{t}_2)$ und als letztes das Sensorkoordinatensystem mit (S, σ) mit $\sigma = (\hat{u}, \hat{v})$. Abbildung 2.4 zeigt die Koordinatensysteme schematisch im Überblick.

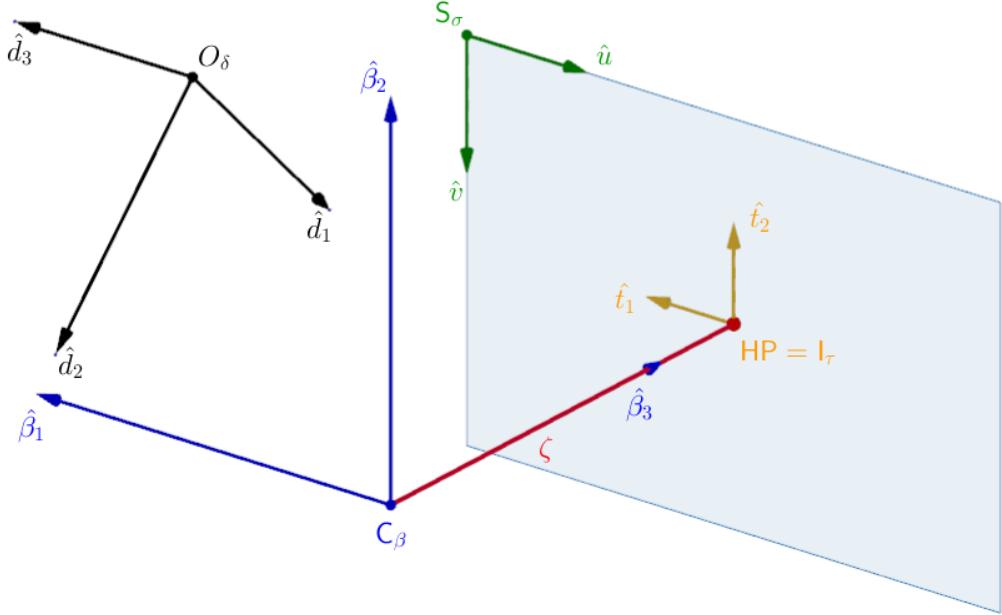


Abbildung 2.4: Das Schaubild zeigt die einzelnen Koordinatensysteme in einem Lochkameramodell. Das Weltkoordinatensystem (O, δ) mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3)$, das Kamerakoordinatensystem (C, β) mit $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3)$, das Bildeckenkoordinatensystem (I, τ) mit $\tau = (\hat{t}_1, \hat{t}_2)$ und das Sensorkoordinatensystem (S, σ) 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}_δ zu \vec{M}_β in das Kamerakoordinatensystem, erfolgt die Kameraprojektion von \vec{M}_β auf m_β 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}_δ auf den Bildpunkt \vec{m}_β 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}_β 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_σ 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 ζ 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 3×3 -Matrix H ausgedrückt, welche die Transformationsmatrizen, sowie die Kameramatrizen zusammenfasst. Die 3×3 -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, τ) und $m'_{\tau'} = (m'_{x\tau'}, m'_{y\tau'}, m'_{z\tau'})^T$ der dazu korrespondierende Punkt der Bildebene (I', τ') . 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 werden. Dies ist in Abbildung 3.1 dargestellt.

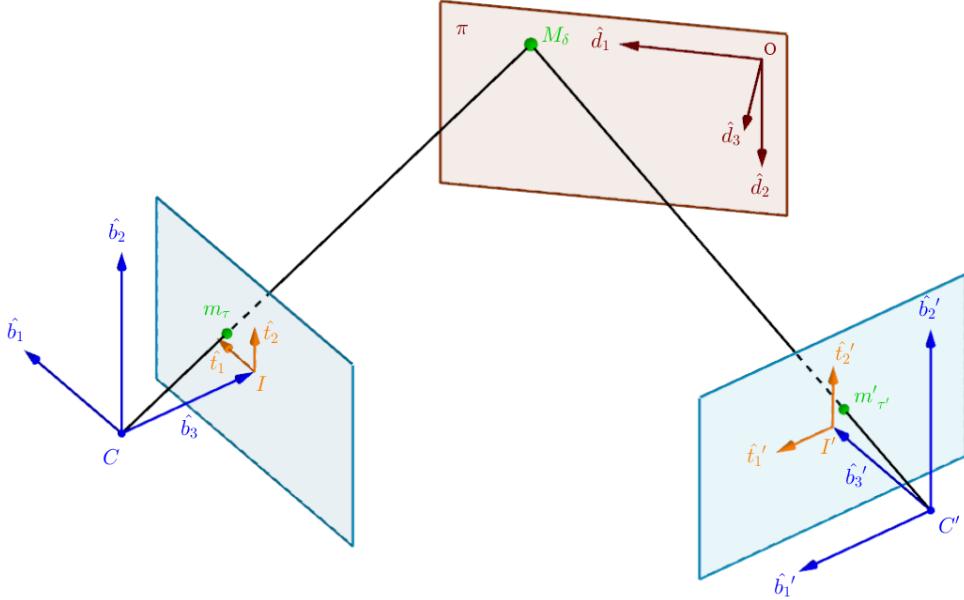


Abbildung 3.1: In der Abbildung sind die beiden Kameras C und C' mit ihren Bildebenen I und I' zu sehen. Ein Objektpunkt M_δ bezüglich eines Weltkoordinatensystems (O, δ) befindet sich auf der Ebene π , welche durch die Achsen \hat{d}_1 und \hat{d}_2 aufgespannt wird. M_δ wird auf I und I' projiziert.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

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_β durch Division mit der $m_{\beta,3}$ -Komponenten eindeutig zu m_τ wird ist die Rückrichtung nicht eindeutig. Alle Punkte auf den Geraden von C und m_τ , siehe Abbildung 3.2, werden auf denselben Bildpunkt projiziert. Die Projektion von m_τ auf m_β ist demnach nicht eindeutig. Die Gerade mit allen möglichen Punkten m_β kann als γm_τ , mit der freien Variablen $\gamma > 0$ ausgedrückt werden.

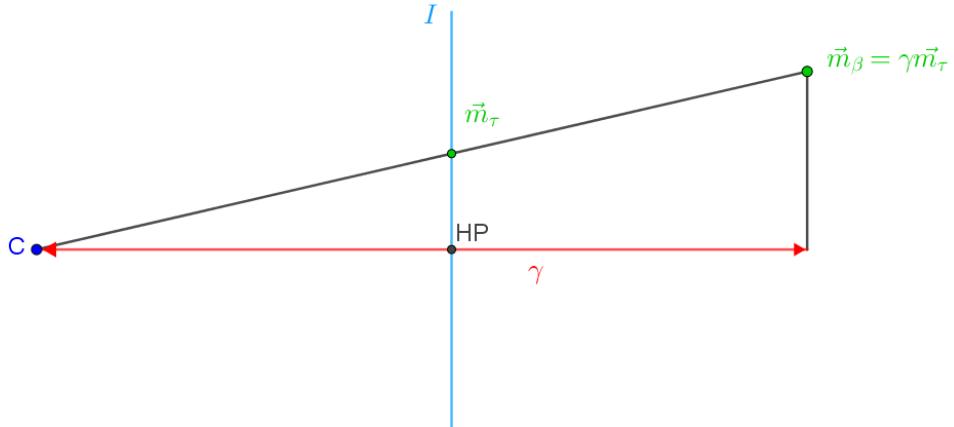


Abbildung 3.2: Auf der Geraden durch C und \vec{m}_τ , befinden sich alle möglichen Punkte für m_β . m_β wird auf Grund seiner Unbestimmtheit als γm_τ bezeichnet

Für zwei korrespondierende Bildpunkte m_τ und m'_τ , kann für alle möglichen Punkte m_β und m'_β 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_δ 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_δ ist der Objektpunkt im 3D-Raum und m_τ und $m'_{\tau'}$ sind die jeweiligen Abbildungen dieses Punktes auf den Bildebene. Die Verbindungsvektoren zwischen C, C' und M_δ 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].

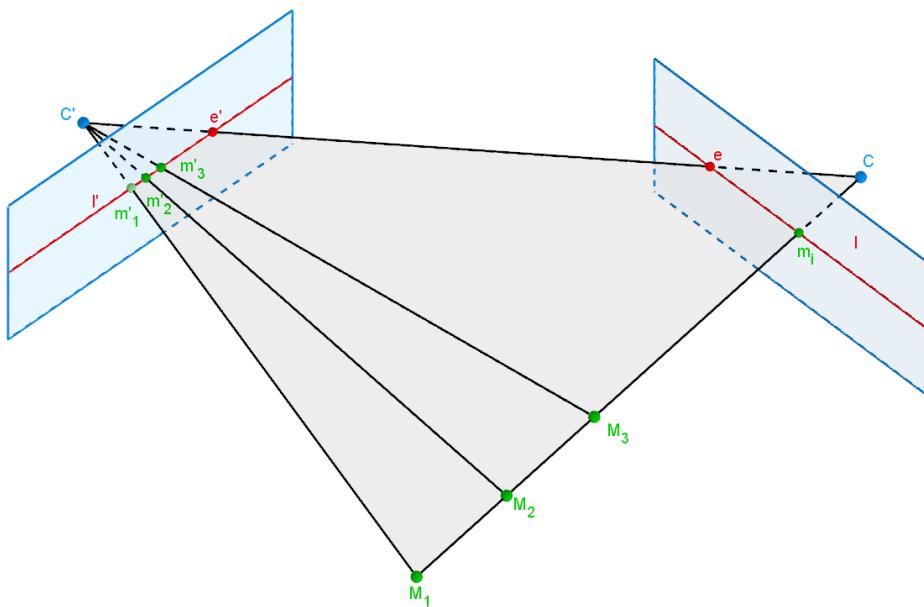


Abbildung 3.4: Die Objektpunkte M_1, M_2 und M_3 werden in I' als m'_1, m'_2 und m'_3 abgebildet, während sie in I immer den selben Bildpunkt m_1 ergeben.

Die hier gezeigte Abbildung von m_i auf l' wird nun genauer betrachtet. Es werden wieder die Gleichungen für m_τ 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 γ und γ' eliminiert werden und folgende Bedingung aus Gleichung 3.17 hergeleitet werden [14]

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

mit

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

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 mit $E = R' [\vec{C}'_\delta - \vec{C}_\delta]_x R^T$ definiert. Gleichung 3.18 kann zu einer Bedingung für 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

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

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

zu normierten Bildebenenkoordinaten 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 λ 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[3, 4, 12]. Der 8-Punkte-Algorithmus ist eine lineare Technik, welche angewandt wird, um die Fundamentalmatrix aus $n \geq 8$ Punkten schätzen zu können. 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 . 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 $A = U \cdot \Sigma \cdot V^T$ mit orthogonalen Matrizen $U \in \mathbb{R}^{m \times n}$ und $V \in \mathbb{R}^{m \times n}$ sowie mit einer Diagonalmatrix Σ .

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

Die Diagonalmatrix Σ beinhaltet die Singulärwerte der Matrix. Dabei soll für die diagonalen Singulärwerte in Σ 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 Σ 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.

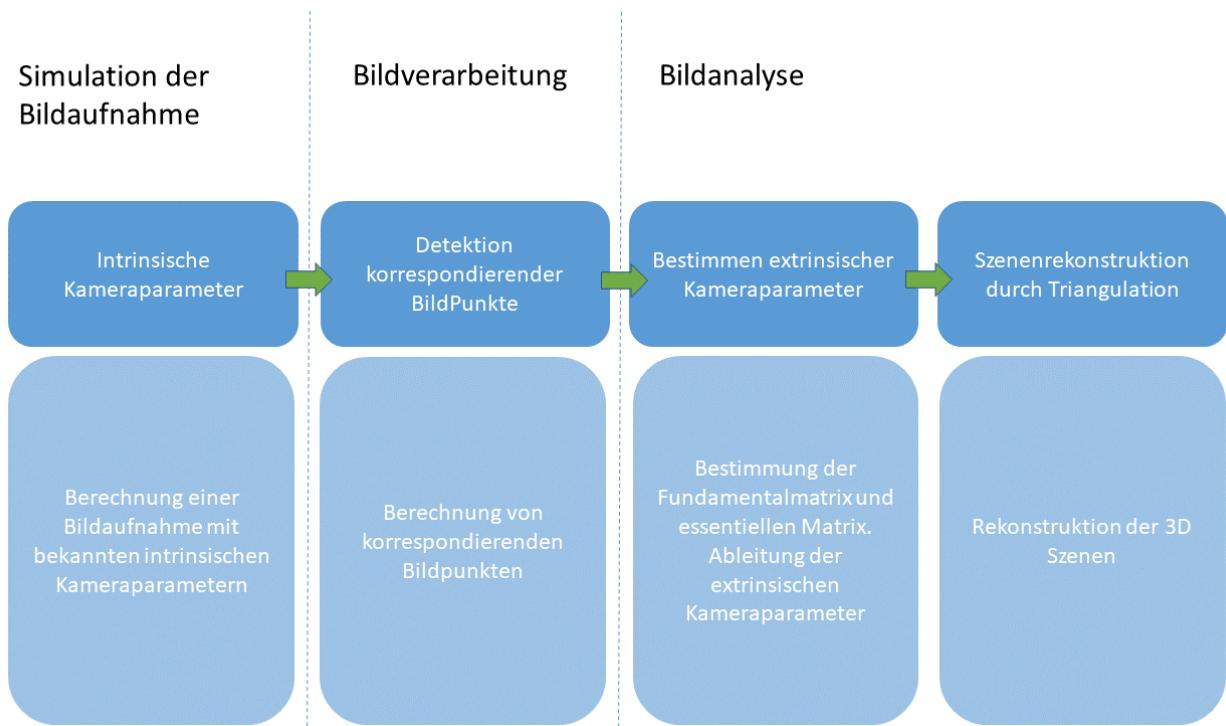


Abbildung 4.1: Ablaufdiagramm für das synthetische Beispiel

4.1 Simulierte Bildaufnahme einer virtuellen Szene

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

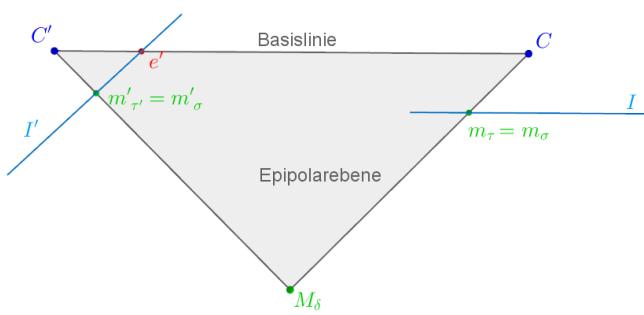


Abbildung 4.2: In der Abbildung ist der vereinfachte Stereoaufbau in einer Top-Down-Ansicht zu sehen

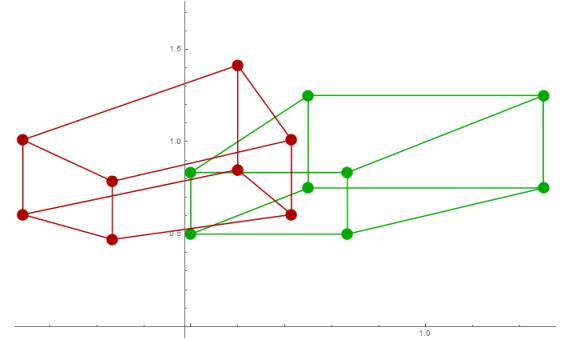


Abbildung 4.3: Simulierte Abbildung des Quaders auf die Kamera C in Grün und auf C' in rot

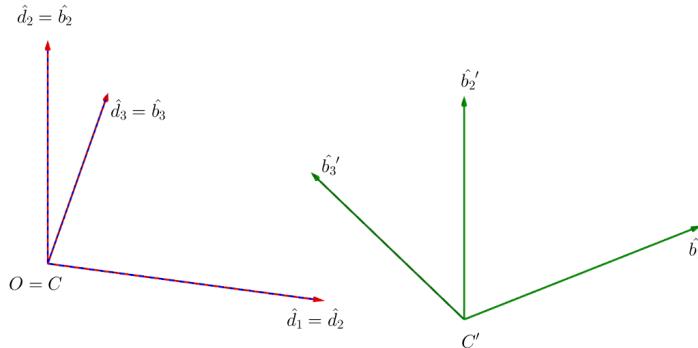


Abbildung 4.4: Abbildung der verschiedenen Kamerakoordinatensysteme, das Weltkoordinatensystem O ist zum Kamerakoordinatensystem C deckungsgleich und C' verschoben und rotiert.

Um die Eckpunkte des Quaders auf die Bildebene von C und C' abbilden zu können, werden zunächst die Projektionsmatrizen P und P' aufgestellt. C ist Deckungsgleich mit dem Weltkoordinatensystem (O, δ) . 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 einen 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, τ) und (I', τ') 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 Σ 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 Σ aus der Singulärwertszerlegung aus E modifiziert[3, 4].

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

$[\vec{C}'_\delta]_\times$ 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 $[\vec{C}'_\delta]_\times$ und R_1 und R_2 aufstellen[3, 4].

$$[\vec{C}'_\delta]_\times = \pm UZU^T \quad (4.29)$$

$$R'_1 = UW^TV^T \quad R'_2 = UWV^T \quad (4.30)$$

$[\vec{C}'_\delta]_\times$ 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 Modifizierung der Singulärwerte von E gilt für $\|v\| = 1$ [3, 4]. Das bedeutet, dass es sich bei dem Translationsvektor v lediglich um den normierten Richtungsvektor zwischen C und C' handelt[21]. Um v aus $[\vec{C}'_\delta]_\times$ zu extrahieren, wird der Kern von $[\vec{C}'_\delta]_\times$ bestimmt

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

Die Skaleninvarianz bewirkt, dass es bei der Rekonstruktion die Größe der Objekte von ihrer Originalgröße abweichen, da es sich bei v wie gesagt nur um den normierten Richtungsvektor der ursprünglichen Strecke handelt. 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]. λv 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. Die Entscheidung welche Lösung T die richtige für das implementierte Beispiel ist, fällt auf diejenige Lösung, bei welcher das Abbild des der Objekte vor den Kamera liegen.

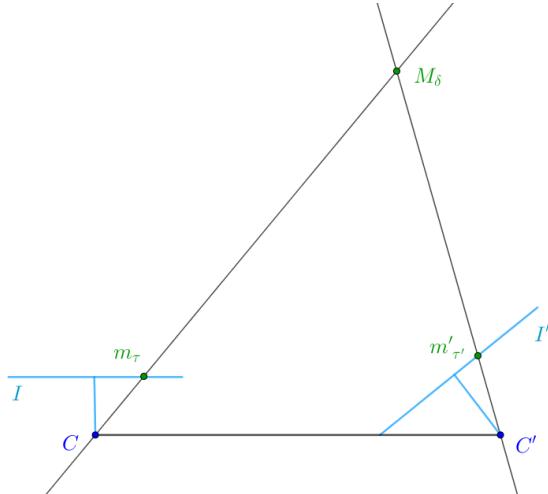


Abbildung 4.5: a)

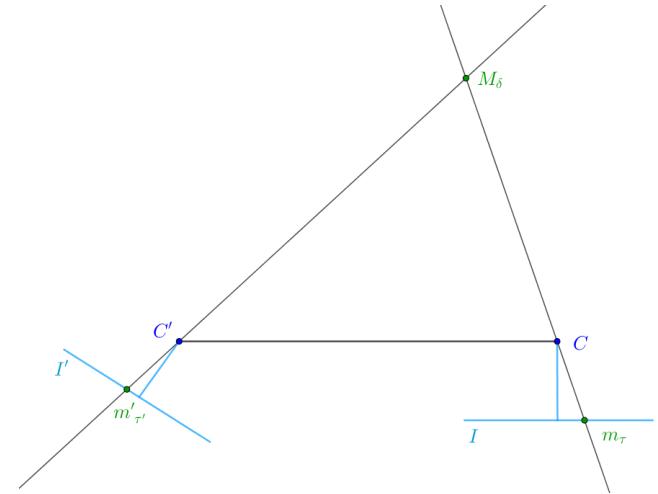


Abbildung 4.6: b)

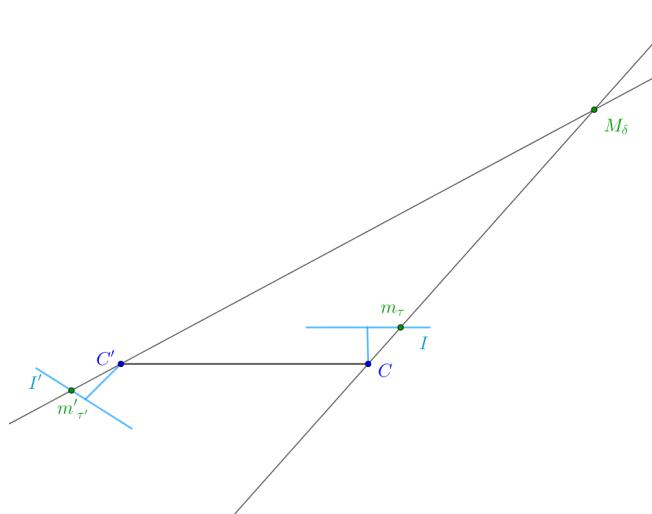


Abbildung 4.7: c)

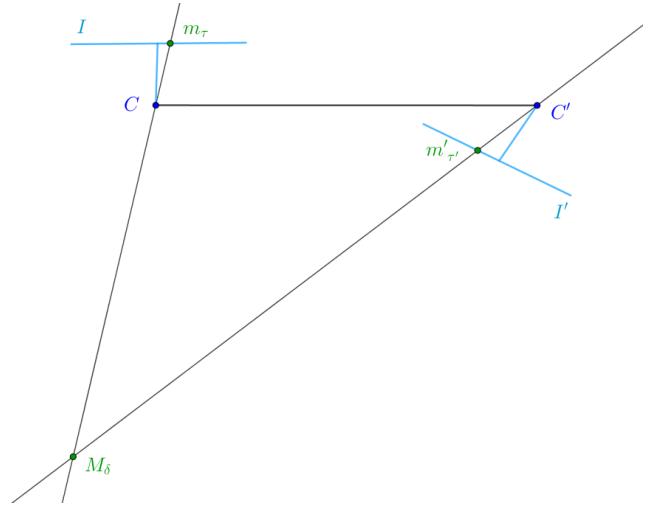


Abbildung 4.8: d)

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° 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.

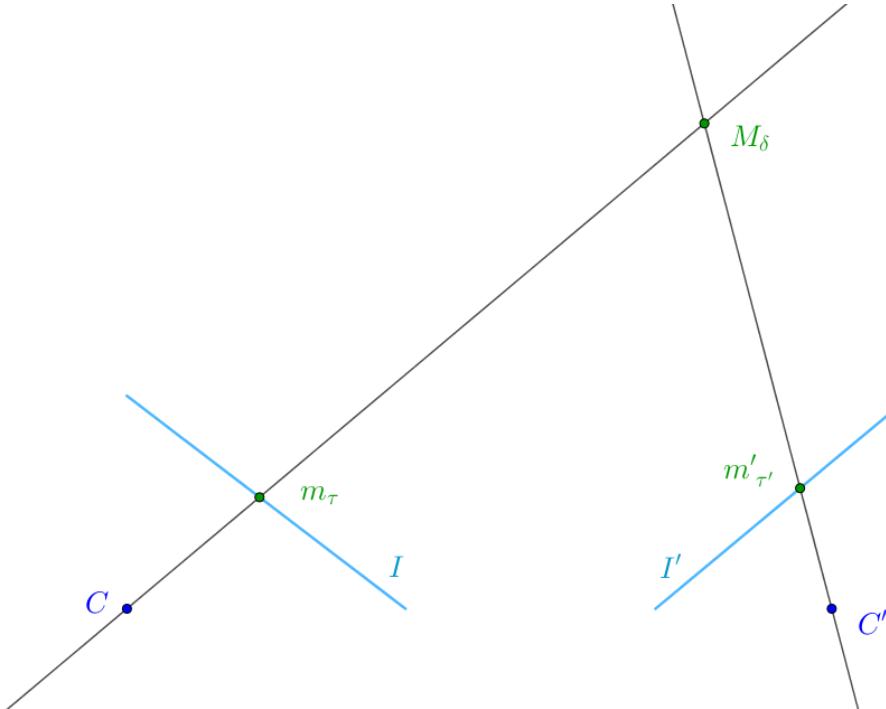


Abbildung 4.10: Optimale Triangulierung: Beide Geraden Treffen sich in einem Punkt im 3D-Raum

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_τ korrespondierender Bildpunkt $m'_{\tau'}$, liegt genau auf der zu m_τ korrespondierenden Epipolarlinien l' . Somit ist garantiert, dass sich die Bildpunkte m_τ 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_δ .

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 $P = K_0 T = K_0[I| - 0]$ ist. Die Projektionsmatrix P' für C' setzt sich aus einer der Lösungen von T' und der Kameramatrix K'_0 zusammen, so dass gilt $P' = K'_0 T' = K[R'| - R'C']$.

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, β) und (O, δ) 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 ζ und ζ' aus K_0 und K'_0 zu einer dreidimensionalen Koordinate erweitert.

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

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

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, δ) definiert. $m'_{\tau'}$ wird noch in Koordinaten bezüglich des Kamerakoordinatesystem (C, β) 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_{δ} entsprechen muss. Dementsprechend wird als letzter Schritt die rekonstruierte Szenen 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 zeigt die rekonstruierte Szene des synthetischen Beispiels. Die Abbildungen 4.14 und 4.15 zeigen die auf Ursprungsgröße skalierte rekonstruierte Szene des Quaders.

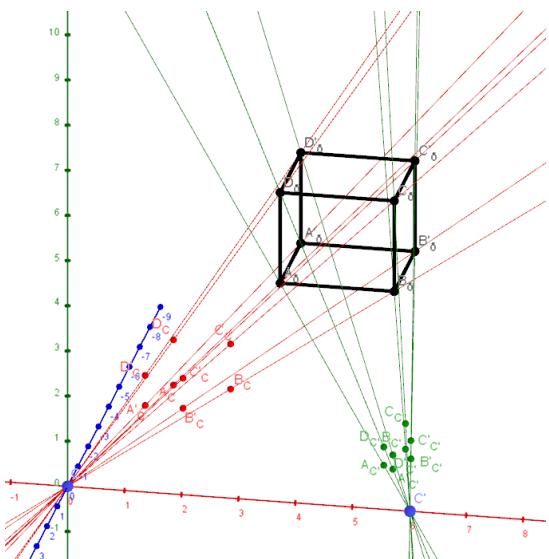


Abbildung 4.11

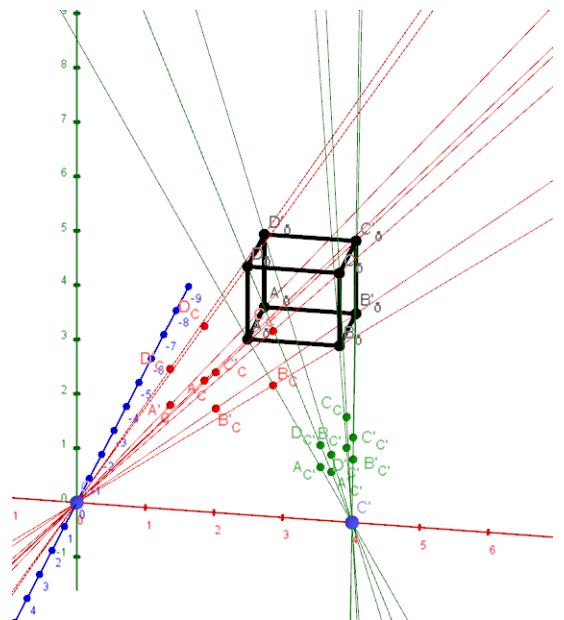


Abbildung 4.12

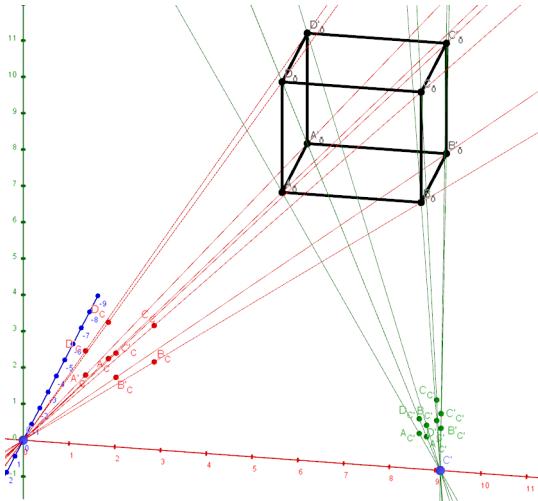


Abbildung 4.13: Veranschaulichung der Skaleninvarianz und dessen Auswirkung auf die geometrische Form und Größe der Objekte

4.3 Auswirkungen von unterschiedlichen Kameraauflösungen

Der entstandene Szenenrekonstruktionsalgorithmus wurde anhand von Kameras gleicher Auflösung implementiert und bereits validiert. Im folgenden soll nun getestet werden, ob der entwickelte Algorithmus auch im Stande ist für Kameras unterschiedlicher Auflösung eine Szene richtig zu Rekonstruieren. Zunächst wird beschrieben, welche Modifizierungen auf einem Sensor bei Veränderung der Auflösung stattfinden. 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 und somit auf die Bestimmung der extrinsischen Kameraparameter und der folgenden Szenenrekonstruktion hat. Zum Schluss werden die Ergebnisse der synthetischen Rekonstruktion mit unterschiedlichen Kameraauflösungen präsentiert und validiert.

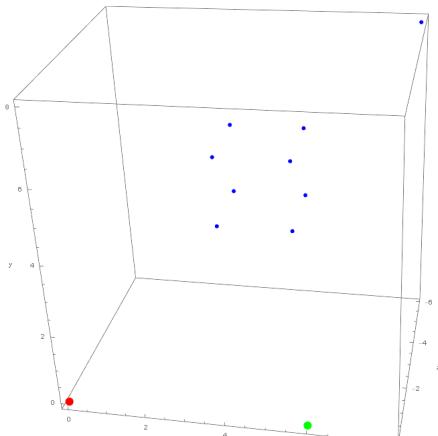


Abbildung 4.14

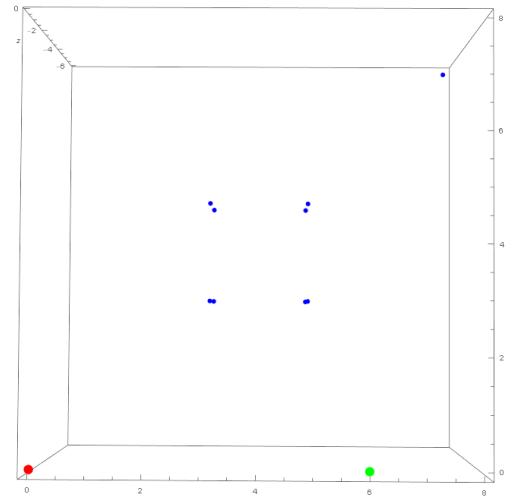


Abbildung 4.15

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. Die Abbildungen entstand aus dem in *Mathematica*[22] implementierten Algorithmus.

4.4 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 4.17 zeigt den schematischen Aufbau eines Sensors (CMOS).

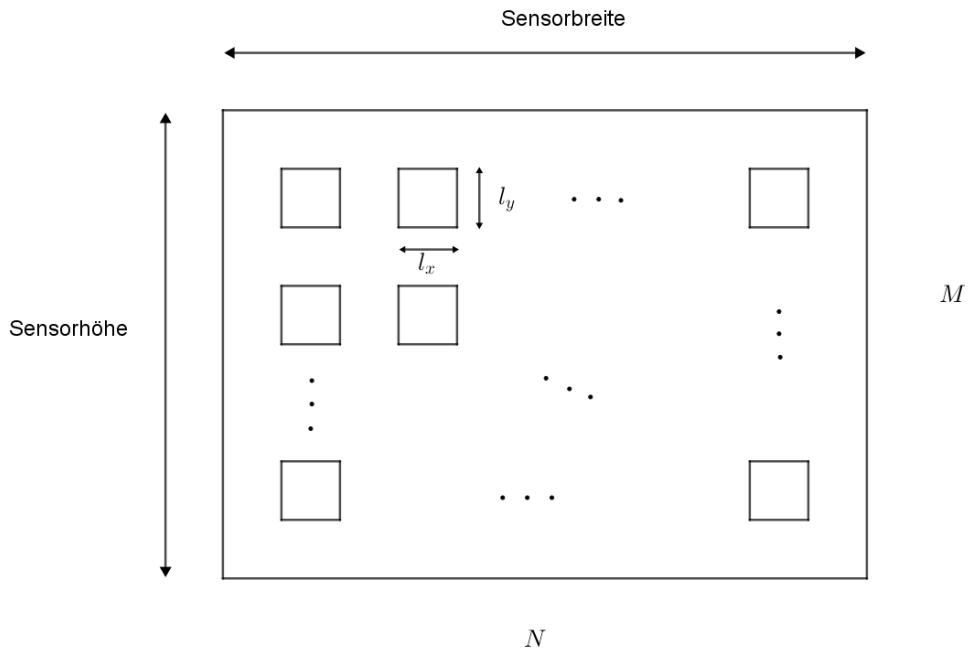


Abbildung 4.17: Rechteckiger Bildsensor mit darauf sich befindenden quadratischen Sensorelementen. Vergleiche [8]

Ein Sensor hat eine maximale Auflösung. Die maximale Anzahl der Sensorelemente auf einem Sensor beschreit die maximale Auflösung. Verschiedene Kameras können aus diesem Grund verschiedenen Auflösungen besitzen. Die Anzahl und Größe der Einzelnen Sensorelemente variert mit den Größen

der Sensorchips. Je größer ein Sensor ist und je mehr Sensorelemente er besitzt, desto besser ist die Bildqualität[8]. Bei maximaler Auflösung definiert genau ein Sensorelement einen Pixel. Ein Pixel wiederum entspricht einem Bildpunkt[8]. Auch ist es möglich die Auflösung eines Sensors digital zu verändern. Wird eine Auflösung kleiner der maximalen Auflösung eingestellt, desto geringer wird die Anzahl der Pixel. Der Prozess, welcher hier stattfindet, gehört zu den Nachbarschaftsoperationen. Benachbarte Pixel werden hier zu einem neuen Pixel definiert[8]. Ein neuer Pixel wird aus den benachbarten Pixeln berechnet.

Eine Veränderung der Auflösung kann auch eine Änderung der Seitenverhältnisse mit einschließen. Ändert sich das Seitenverhältnis so wird der Bereich der lichtempfindlichen Fläche auf dem Sensor beschränkt[8]. Dies führt dazu, dass sich die Bildausschnitte ändern. Abbildung 4.18 stellt schematisch da wie sich die lichtempfindlichen Bereiche auf dem Sensor bei unterschiedlichen Auflösungen ändert.

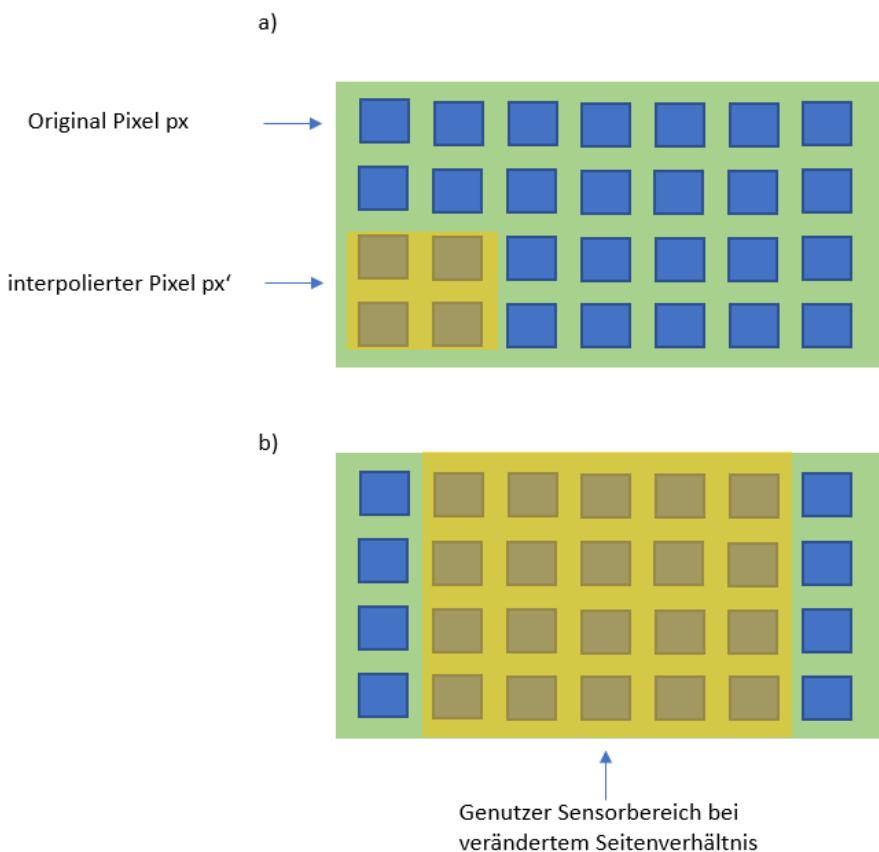


Abbildung 4.18: Bild a) zeigt die den Zusammenschluss mehrerer benachbarter Pixel zu einem neuen Pixel. Bild b) zeigt in gelb markiert, den aktiven lichtempfindlichen Bereich des Sensors, wenn sich das Seitenverhältnis geändert wird und nicht mehr der komplette Sensor genutzt wird.

4.5 Auswirkungen auf die Szenenrekonstruktion

Im folgenden wird zunächst analysiert, welche Änderungen sich im Lochkameramodell bei veränderter Auflösung ergeben und was für Auswirkungen diese auf die in Kapitel 3 aufgestellten Epipolaren Bedingungen hat.

Im Lochkameramodell hat eine Änderung der Kameraauflösung lediglich eine Auswirkung auf die Skalierung der Sensorkoordinatenachsen. Mit der Auflösung, ändern sich die Anzahl und die Größe der

Pixel. Die Längeneinheiten des Sensorkoordinatensystems orientieren sich, wie in Kapitel 2 beschrieben, an genau diesen Längenskalierung der Pixelkanten l_x und l_y . Folglich kommt es zu einer Skalierung des Sensorkoordinatensystems. Alle anderen Koordinatensysteme bleiben unverändert. Durch Nachbarschaftsoperationen werden aus mehreren benachbarten Pixel ein neuer, jedoch bleibt der Ort des Pixel der gleiche[23]. Die Abbildungen 4.19 und 4.20 zeigen, dass sich zwar die Projektion von Bildebene koordinaten auf das Sensorkoordinatensystem für den Punkt m_σ ändert,

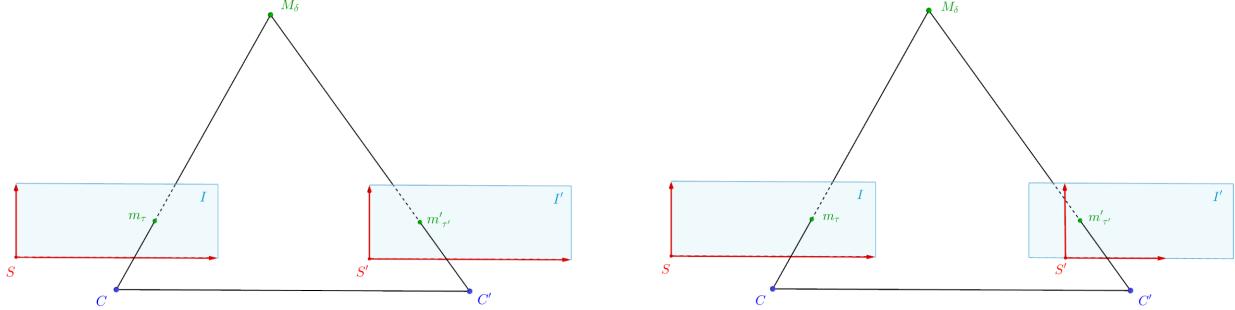


Abbildung 4.19: C und C' haben die selbe Auflösung eingestellt

Abbildung 4.20: C und C' haben unterschiedliche Auflösungen eingestellt

Eine Skalierung der Sensorkoordinaten bedeutet, dass sich die Brennweite in Pixeleinheiten gegeben ändert, jedoch ändert sich nicht die effektive Brennweite in Millimeter. Anhand des Aufbau der Kameramatrix aus Kapitel 2 kann das nochmal verdeutlicht werden.

$$K = \begin{bmatrix} k_x \zeta & 0 & V_{\sigma,x} \\ 0 & k_y \zeta & V_{\sigma,y} \\ 0 & 0 & 1 \end{bmatrix} \quad (4.39)$$

Wie bereits bekannt wird kommt es bei der Transformation von Bildebene koordinaten m_τ auf Sensorkoordinaten m_σ zu einer Skalierung der Bildebene koordinaten in Millimeter auf Sensorkoordinaten in Pixel. ζ_x und ζ_y stehen für die Brennweite. Durch die Multiplikation mit k_x und k_y wird die Brennweite auf Pixeleinheiten skaliert. Die ursprüngliche Brennweite beträgt $\zeta_x = \zeta_y = 1$. Kommt jetzt eine Skalierung von $k_x = k_y = 2$ dazu.

$$K_0 = \begin{bmatrix} \zeta & 0 & V_{\tau,x} \\ 0 & \zeta & V_{\tau,y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & V_{\tau,x} \\ 0 & 1 & V_{\tau,y} \\ 0 & 0 & 1 \end{bmatrix} \quad (4.40)$$

$$K = \begin{bmatrix} k_x \zeta & 0 & k_x V_{\tau,x} \\ 0 & k_y \zeta & k_y V_{\tau,y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 \cdot 1 & 0 & 2 \cdot V_{\tau,x} \\ 0 & 2 \cdot 1 & 2 \cdot V_{\tau,y} \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & V_{\sigma,x} \\ 0 & 2 & V_{\sigma,y} \\ 0 & 0 & 1 \end{bmatrix} \quad (4.41)$$

Die Veränderung der Kameraauflösung hat in diesem Beispiel zur Folge, dass es so wirkt, als wäre die Brennweite verdoppelt worden. Das würde bedeuten, dass sich die Kamera von der Bildebene entfernt hätte, jedoch verändert weder Kamera noch Bildebene ihre Position. Dennoch vergrößert oder verkleinert sich durch die Skalierung der Pixel effektiv die Bildgröße.

Um zu überprüfen, ob die Änderung der Kameraauflösung auch eine Änderung der Epipolaren Bedingungen mit sich führt, werden wieder die Gleichungen der Epipolaren Bedingungen in Kapitel 3 genauer betrachtet.

Die Fundamentalmatrix beinhaltet, sowohl die intrinsischen als auch die extrinsischen Parameter, um von F auf E zu kommen, müssen die intrinsischen Kameraparameter bekannt sein. Mit bekannten K und K' gilt, dass:

$$F = K'^{-T} R' \left[\vec{C}'_\delta - \vec{C}_\delta \right]_x R^T K^{-1} \quad (4.42)$$

$$E = K'^T F K \quad (4.43)$$

$$E = K'^T (K'^{-T} R' \left[\vec{C}'_\delta - \vec{C}_\delta \right]_x R^T K^{-1}) K \quad (4.44)$$

$$\rightsquigarrow E = R' \left[\vec{C}'_\delta - \vec{C}_\delta \right]_x R^T \quad (4.45)$$

Da bei der Bestimmung von E aus F die intrinsischen Kameraparameter eliminiert werden, haben unterschiedliche Auflösungen keine Auswirkung auf die essentielle Matrix. Folglich sollte das Ergebnis bei der Bestimmung der extrinsischen Kameraparameter unverändert sein. Um die Aufgestellte Theorie zu überprüfen, wurde im synthetischen Beispiel die Kameramatrix K' von C' modifiziert. Für C wurde $\zeta_x = \zeta_y = 1$ definiert, so das für Kameramatrix K gilt:

$$K = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.46)$$

Für die Kameramatrix K' von C' galt ursprünglich, dass $K = K'$. Die Auflösung von C' wird geändert indem die Skalierungsfaktoren k_x und k_y mit ζ'_x und ζ'_y multipliziert werden.

Für das Beispiel wurden drei verschiedenen Auflösungen getestet. K'_1 mit $\zeta'_x \cdot 2$ und $\zeta'_y \cdot 2$, K'_2 mit $\zeta'_x \cdot 3.2$ und $\zeta'_y \cdot 1.2$ und K'_e mit $\zeta'_x \cdot 0.5$ und $\zeta'_y \cdot 4.3$. Da ursprünglich galt, dass $\zeta'_x = \zeta'_y = 1$ ergeben sich die folgenden Kameramatrizen für K' .

$$K'_1 = \begin{bmatrix} 2 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.47)$$

$$K'_2 = \begin{bmatrix} 3.2 & 0 & 0 & 0 \\ 0 & 1.2 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.48)$$

$$K'_e = \begin{bmatrix} 0.5 & 0 & 0 & 0 \\ 0 & 4.3 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.49)$$

Die Unterschiede der entstehenden Abbildungen in C' sind in den Abbildungen 4.21, 4.22, 4.23 und 4.24 zu sehen.

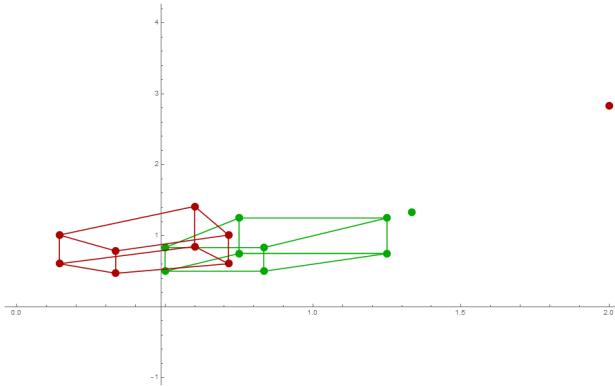


Abbildung 4.21: C und C' haben die selbe Auflösung eingestellt

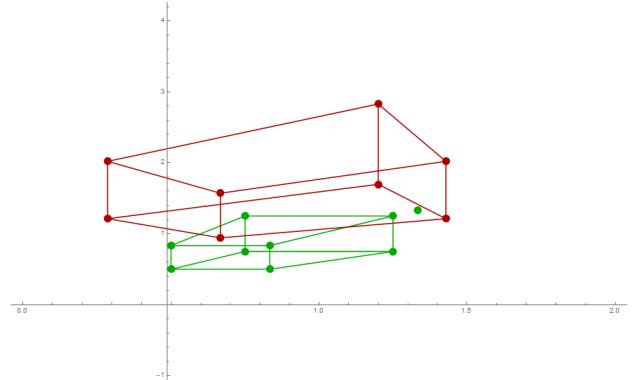


Abbildung 4.22: C und C' haben unterschiedliche Auflösungen eingestellt. C mit K und C' mit K'_1

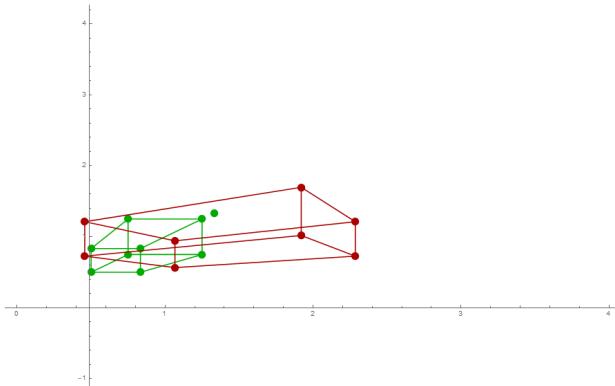


Abbildung 4.23: C mit K und C' mit K'_2

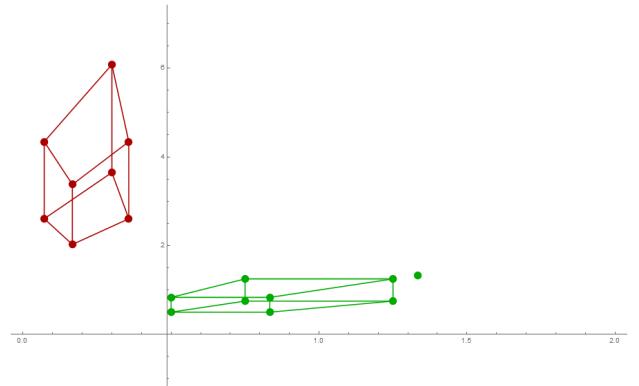


Abbildung 4.24: C mit K und C' mit K'_3

Wird das synthetische Beispiel jeweils mit den drei verschiedenen modifizierten K' durchgerechnet, so ergeben sich für die essentielle Matrix folgende Ergebnisse.

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

Wie beobachtet werden kann, werden, trotz unterschiedlicher Kameraauflösungen für K' , immer die selbe essentielle Matrix im Algorithmus bestimmt. Zu Erinnerung, in Kapitel 3 wurde gezeigt, dass jedes Vielfache von E eine gültige Lösung ist. Somit gilt die Behauptung, dass die Kameraauflösung keine Auswirkung auf die Bestimmung der extrinsischen Kameraparameter hat, im synthetischen Beispiel, als bestätigt. Als Vergleich kann die Abbildung 4.25, welche das Ergebnis der Rekonstruktion der

Szene mit K'_3 als Kameramatrix für C' veranschaulicht, mit der Rekonstruierten Szene in Abbildung 4.14 und 4.15 aus dem ersten Beispiel, betrachtet werden. Für die anderen Varianten von K' wurden ebenfalls die selbe 3D-Szene rekonstruiert.

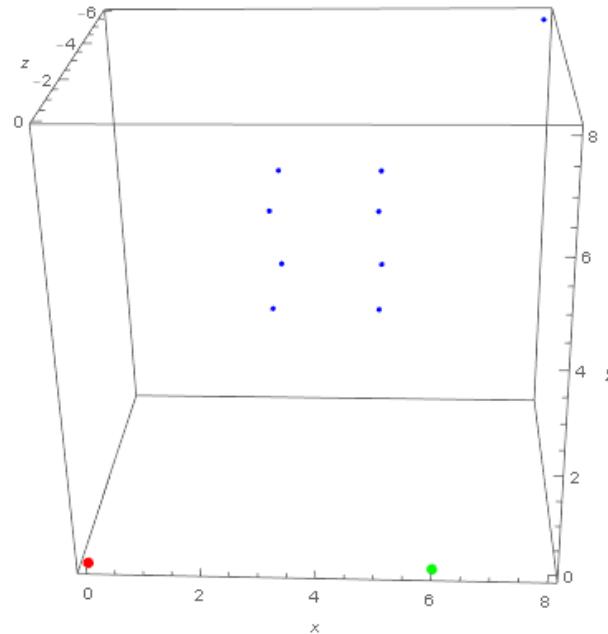


Abbildung 4.25: Die Abbildung zeigt die rekonstruierte Szenen des synthetischen Beispiels mit K'_3 als intrinsische Parameter für C' .

5 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 5.1 zeigt den Ablauf für die Reelle Rekonstruktion.

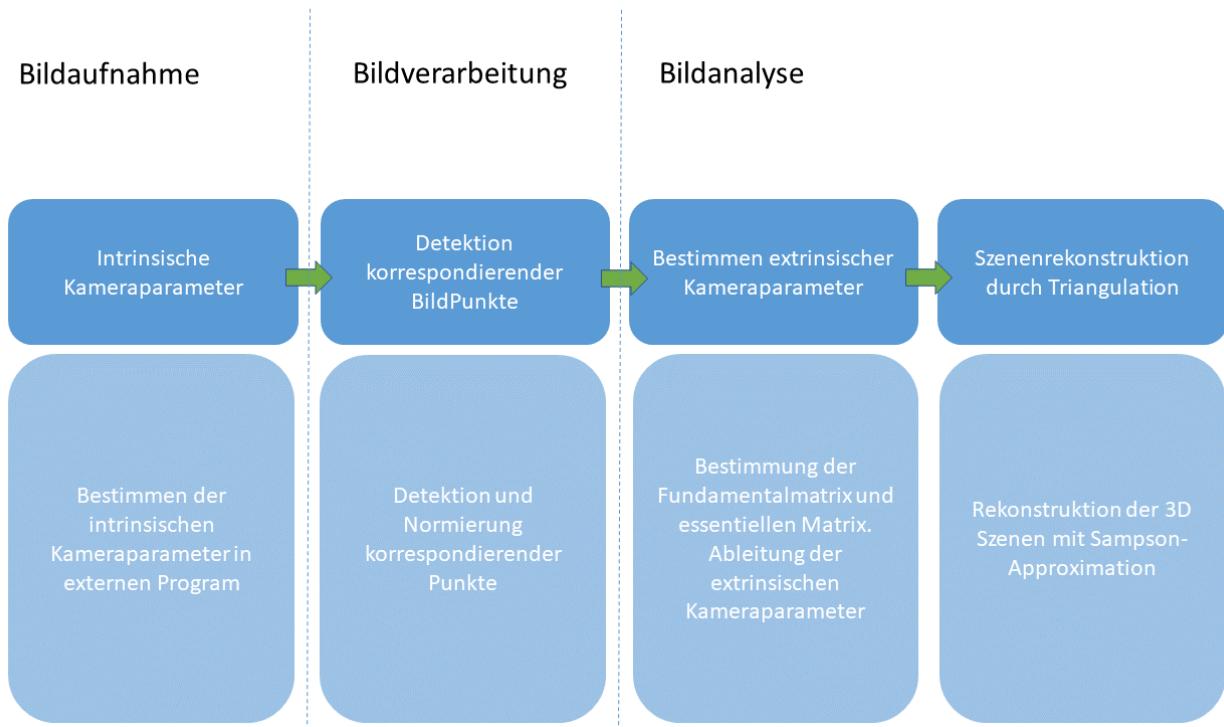


Abbildung 5.1: Ablaufdiagramm für die reelle Rekonstruktion

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.

5.1 Stereoaufbau

Für die Stereobildaufnahme, wurde eine Szene vor zwei nebeneinander platzierten Kameras platziert. Beide Kameras wurden zur Szene hin rotiert. Abbildung 5.2 zeigt den Stereoaufbau.



Abbildung 5.2: 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.

Die auf Abbildung 5.2 zu sehende linke Kamera wurde als Kamera eins definiert. Für Kamerakoordinatensystem (C, β) wurde gleich dem Weltkoordinatensystem (O, δ) gesetzt. Kamera zwei mit (C', β') befindet sich auf der Abbildung rechts. Für beide Kameras wurde in einem externen Programm die intrinsischen Kameraparameter K und K' bestimmt. An den Stereoaufnahmen der Szene wurde dann eine Korrespondenzanalyse gemacht.

5.2 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, 24]. 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, 24]. Die Abbildungen 5.4 und 5.3 zeigen die Ergebnisse nach der Anwendung des SURF-Algorithmus auf das Stereobildpaar. Eine eigens implementierte alternative für die Korrespondenzanalyse zwischen Stereoaufnahmen eines zweidimensionalen Schachbretts wird in Kapitel 7 vorgestellt.



Abbildung 5.3: a

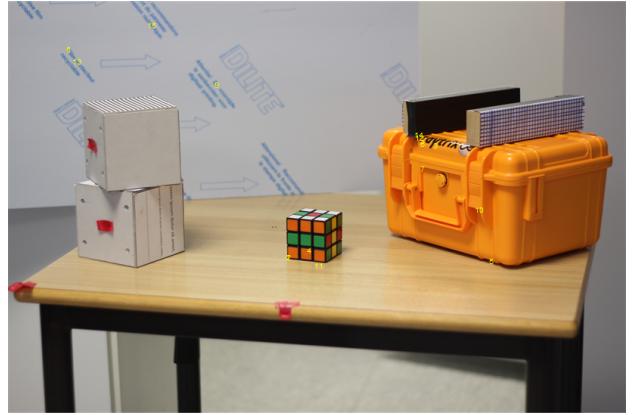


Abbildung 5.4: b

Die mit dem *SURF*-Algorithmus 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*-Algorithmus, können immer Fehler und Abweichungen mit sich bringen. Die Ursprünge der Fehler können sowohl durch den Algorithmus 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.

5.3 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, 25].

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, 26, 27]. 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, 26]. Wird angenommen, dass AA^T eine 9×9 -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 Sensorkoordinatensystems liegt und der durchschnittliche Abstand der Punkte zum Ursprung $\sqrt{2}$ beträgt[3, 4, 25].

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.

$$\begin{pmatrix} s_x \\ s_y \end{pmatrix} = \frac{1}{n} \sum_{i=1}^n \begin{pmatrix} p_i x \\ p_i y \end{pmatrix} \quad (5.1)$$

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:

$$T = \begin{pmatrix} \frac{\sqrt{2}}{d} & 0 & -s_x \\ 0 & \frac{\sqrt{2}}{d} & -s_y \\ 0 & 0 & 1 \end{pmatrix} \quad (5.2)$$

$$T' = \begin{pmatrix} \frac{\sqrt{2}}{d'} & 0 & -s'_x \\ 0 & \frac{\sqrt{2}}{d'} & -s'_y \\ 0 & 0 & 1 \end{pmatrix} \quad (5.3)$$

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, 25]. Nachdem \hat{F} aus den normierten Koordinaten bestimmt wurde, wird sie mit T und T' wieder denormalisiert.

$$F = T'^T \hat{F} T \quad (5.4)$$

5.3.1 Singularität der Fundamentalmatrix

Eine Fundamentalmatrix ist im optimalen Fall eine singuläre Matrix mit Rang 2. Die Singularität der Fundamentalmatrix sorgt zum einen dafür dass ihr rechter und linker Kern jeweils den Epipol des jeweiligen Bildes ergibt und die Epipolarlinien auch alle durch eben diese Epipole verlaufen[3]. Durch Ungenauigkeiten in korrespondierenden Bildpunkten kann es dazu kommen, dass die aus dem normierten-acht-Punkt-Algorithmus bestimmte Fundamentalmatrix \hat{F} in ihrem Rang steigt und somit keine singuläre Matrix mehr ist. Sollte dies der Fall sein, so ergeben der linke und der Rechte Kern von F keine eindeutigen Lösungen mehr für e und e' und die Epipolarlinien in beiden Bildern gehen dem entsprechend auch nicht mehr durch genau einen Punkt, wie man in den Abbildungen 5.5 und 5.6 erkennen kann. Die Abbildungen bilden Epipolarlinien aus dem Stereobildpaar 5.3 und 5.4 ab.

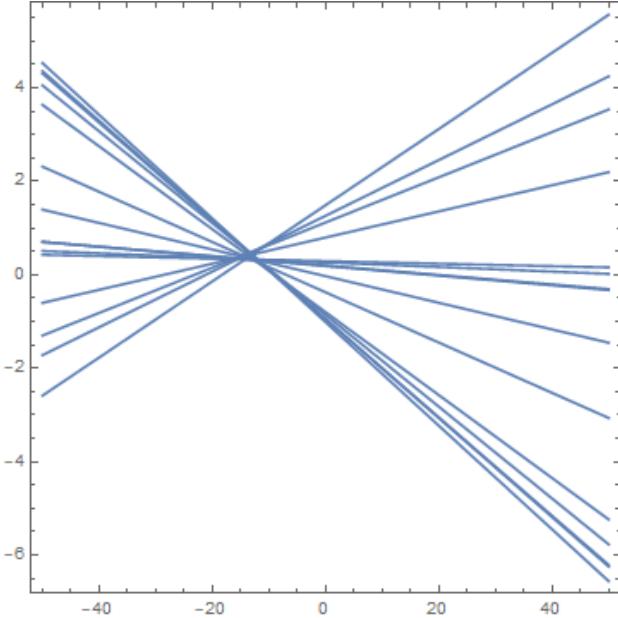


Abbildung 5.5: Epipolarlinien ohne *Epipolar-constraint* im Bild der Canon 6D

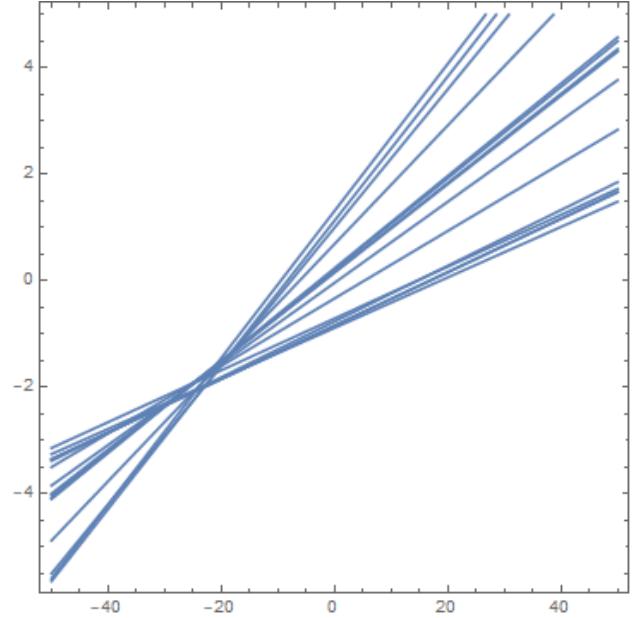


Abbildung 5.6: Epipolarlinien ohne *Epipolar-constraint* im Bild der Canon 60D

Um mit dem Algorithmus weiter verfahren zu können, muss die Singularität in \hat{F} erzwungen werden[3]. Hierfür wird eine Singulärwertszerlegung an \hat{F} durchgeführt, so dass \hat{F} in $\hat{F} = U\Sigma V^T$ zerlegt wird. Σ beinhaltet in einer Diagonalmatrix die Singulärwerte $D = \text{diag}(r, s, t)$. Die Diagonaleinträge erfüllen die Bedingung, dass $r \leq s \leq t$ gilt. Damit \hat{F} zu einer singulären Matrix wird, muss für die Diagonaleinträge gelten, dass $\Sigma = \text{diag}(r, s, 0)$ ist. Dem entsprechend wird der letzte Eintrag t auf $t = 0$ gesetzt und eine modifizierte Fundamentalmatrix mit $\bar{F} = U\text{diag}(r, s, 0)V^T$ wieder zusammengesetzt. Die resultierende Fundamentalmatrix \bar{F} besitzt jetzt einen Rang 2 und ist somit singulär[3]. \bar{F} ist somit, laut Frobenius Norm, die nächste zu F liegende singuläre Matrix[3]

Werden aus \bar{F} der rechte und linke Kern bestimmt, so ergeben sich eindeutige Lösungen für e und e' und die Epipolarlinien l und l' verlaufen jeweils durch ihre entsprechenden Epipole[3]. Die Abbildungen 5.7 und 5.8 zeigen die Auswirkung der erzwungenen Singularität von F auf dem Stereobildpaar 5.3 und 5.4. Die Abbildungen 5.9 und 5.10 die selben Epipolarlinien nur ist \bar{F} mit T und T' denormalisiert worden.

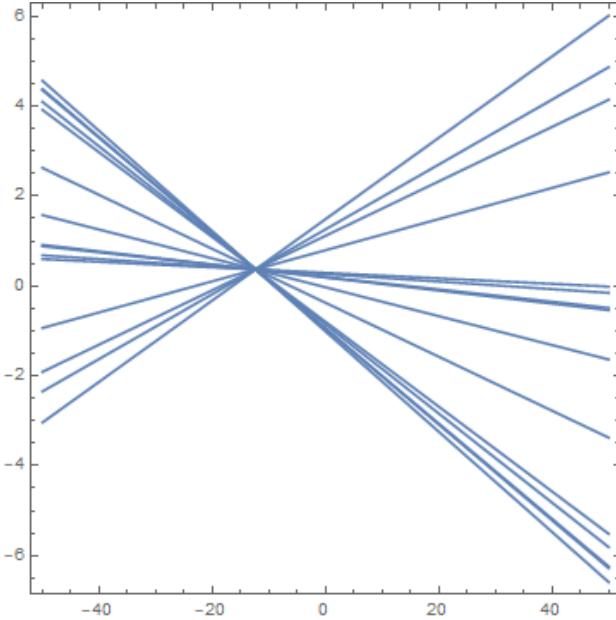


Abbildung 5.7: Die Abbildung zeigt, dass die Epipolarlinien auf der Aufnahme von C , nach dem Erzwingen der Singularität in der normierten \hat{F} , alle durch einen Epipol verlaufen

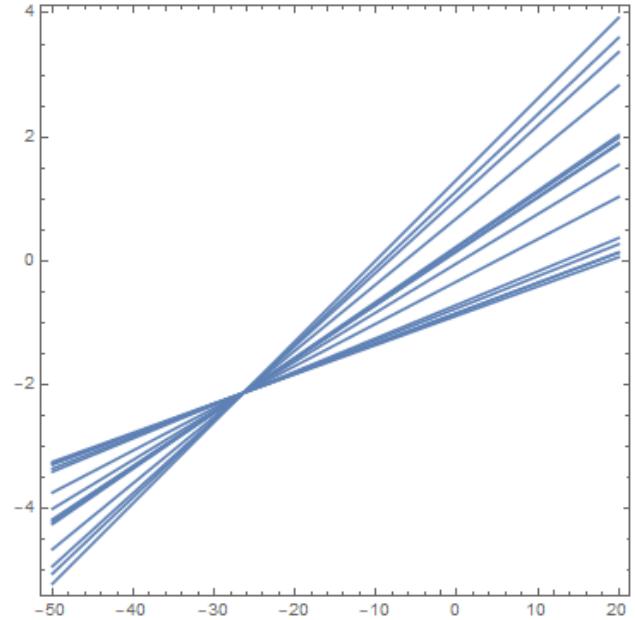


Abbildung 5.8: Die Abbildung zeigt, dass die Epipolarlinien auf der Aufnahme von C' , nach dem Erzwingen der Singularität in der normierten \hat{F} , alle durch einen Epipol verlaufen

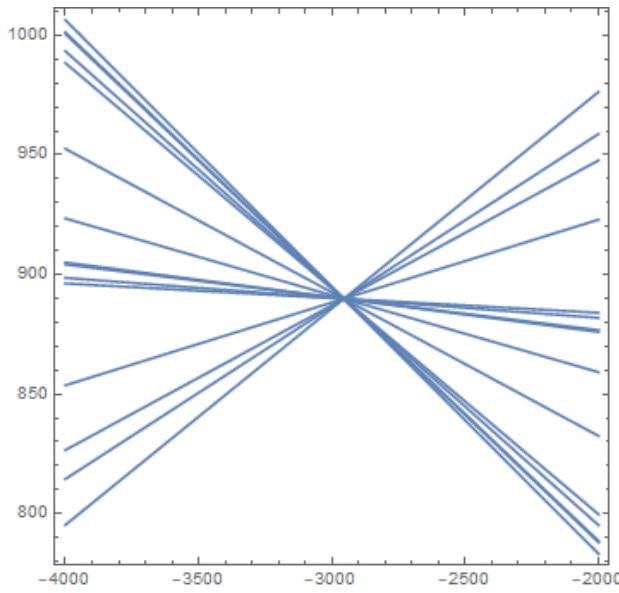


Abbildung 5.9: Die Abbildung zeigt die Epipolarlinien in C nachdem die Fundamentalmatrix \bar{F} mit $F = T' \bar{F} T$ denormalisiert wurde

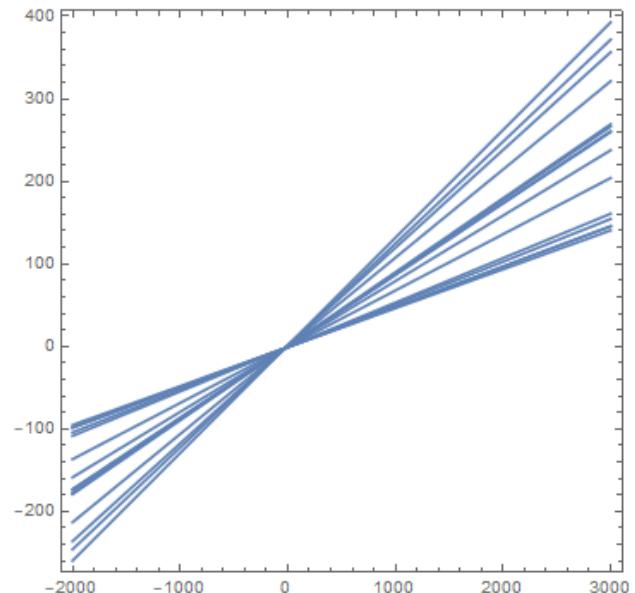


Abbildung 5.10: Die Abbildung zeigt die Epipolarlinien in C' nachdem die Fundamentalmatrix \bar{F} mit $F = T' \bar{F} T$ denormalisiert wurde

5.3.2 Singulärwerte der essentiellen Matrix

Die essentielle Matrix wird im entwickelten Algorithmus aus der Fundamentalmatrix F bestimmt. Da zuvor der die Singularität von F erzwungen wurde, ist die essentielle Matrix ebenfalls eine Matrix von Rang 2[3]. Im synthetischen Beispiel wurde gezeigt, dass für die Bestimmung der extrinsischen Kameraparameter für E gelten muss, dass für ihre Singulärwerte $\Sigma = \text{diag}(1, 1, 0)$ gelten muss.

Die Ungenauigkeit der Punkte, kann auf E die Auswirkung haben, dass ihre Singulärwerte die Form $\Sigma = \text{diag}(a, b, c)$ mit $c \leq b \leq a$ annehmen. Eine Matrix gilt nur dann als gültige essentielle Matrix, wenn zwei ihrer Singulärwerte gleich sind ($a = b$) und der dritte ($c = 0$). Um diese Bedingung zu erzwingen, wird diejenige essentielle Matrix \hat{E} gesucht, welche sich laut der Frobenius Norm am nächsten an der Ursprünglichen E befindet[3, 4]. Diese Matrix lässt sich aus $E = U\Sigma V^T$ bestimmen, indem eine neue essentielle Matrix \hat{E} aus $\hat{E} = U\hat{\Sigma}V^T$ mit $\hat{\Sigma} = \text{diag}(\frac{a+b}{2}, \frac{a+b}{2}, 0)$ [3].

Nach erzwingen der Bedingung $\hat{E} = U\hat{\Sigma}V^T$ mit $\hat{\Sigma} = \text{diag}(\frac{a+b}{2}, \frac{a+b}{2}, 0)$, ist E wieder eine gültige essentielle Matrix und der Algorithmus kann mit der Bestimmung der extrinsischen Kameraparameter, wie in Kapitel 4 gezeigt, fortfahren.

5.4 Szenenrekonstruktion mit Sampson-Approximation

Aufgrund der Ungenauigkeit der korrespondierenden Punkte ist es nicht möglich die 3D-Objektpunkte durch einfache Rückprojektion der Bildpunkte zu rekonstruieren. Liegt der zu m_σ korrespondierende Bildpunkt $m'_{\sigma'}$ nicht ganz genau auf der zu m_σ korrespondierenden Epipolarlinie, so ist der in Kapitel 3 aufgestellt *Epipolar-Constraint* aus Gleichung 3.18 nicht mehr erfüllt. Durch einsetzen der durch den SURF-Algorirthmus detektierten korrespondierenden Punkte m_σ und $m'_{\sigma'}$ in die Gleichung

$$m'^T_\sigma F m_\sigma = 0 \quad (5.5)$$

kommt ein Wert $\neq 0$ heraus. Je weiter der Wert von 0 abweicht, desto ungenauer ist die Korrespondenz beider Bildpunkte. Dies führt dazu, dass bei der Rückprojektion der Bildpunkte m_σ und $m'_{\sigma'}$ sich die Linien nicht im Raum treffen. Die Abbildungen 5.11 und 5.12 veranschaulichen eine Konsequenz von ungenauen Punktekorrespondenzen.

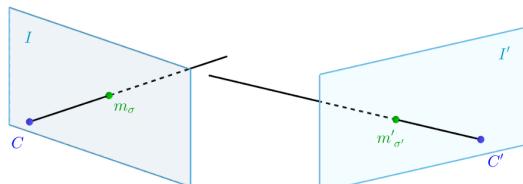


Abbildung 5.11: a)

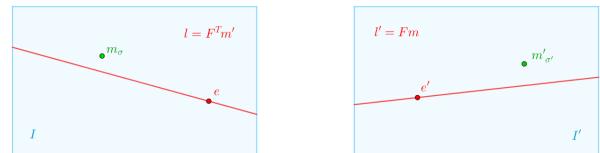


Abbildung 5.12: b)

Abbildung 5.13: a) Die rückprojizierten Strahlen der ungenauen korrespondierenden Punkte m_σ und $m'_{\sigma'}$ sind windschief zueinander und treffen sich nicht in einem Punkt M_δ im Raum.
b) Die korrespondierenden Bildpunkte m_σ und $m'_{\sigma'}$ erfüllen nicht den *Epipolar-Constraint*. Die Epipolarlinie $l' = Fm$ ist die korrespondierende Epipolarlinie zu m_σ und $l = F^T m'$ ist die korrespondierende Epipolarlinie zu $m'_{\sigma'}$. Da weder m_σ noch $m'_{\sigma'}$ auf der Epipolarlinie zum jeweils korrespondierenden Punkt liegt, kommt es zu keinem Schnittpunkt der rückprojizierten Strahlen

Um trotz der ungenauen korrespondierenden Punkte eine Triangulation zu ermöglichen, wurde ein Verfahren voran geschaltet, welches zwei Punkte \hat{m}_σ und $\hat{m}'_{\sigma'}$ sucht, die möglichst nah an den ursprünglichen Punkten m_σ und $m'_{\sigma'}$ liegen und gleichzeitig den *Epipolar-Constraint* $\hat{m}'^T_\sigma F \hat{m}_\sigma = 0$ erfüllen. \hat{m}_σ und $\hat{m}'_{\sigma'}$ sollen durch Minimierung einer Kostenfunktion C bestimmt werden, welche die Distanz d zwischen m_σ und \hat{m}_σ und $m'_{\sigma'}$ und $\hat{m}'_{\sigma'}$ minimiert. Für die Minimierung wird das Verfahren der Sampson-Approximation gewählt[3]. (Literatur zu Sampson approximation?? erklären wie??) Voraussetzung für die Triangulierung ist, dass die Projektionsmatrizen P und P' , sowie die Fundamentalmatrix F bekannt sein müssen[3].

$$C(m, m') = d(m, \hat{m})^2 + d(m', \hat{m}')^2 \quad (5.6)$$

Die optimalen \hat{m} und \hat{m}' liegen auf den korrespondierenden Epipolarlinien \hat{l} und \hat{l}' am Fuße des Lots, welches von den ursprünglich projizierten Punkten m und m' auf die Epipolarlinien \hat{l} und \hat{l}' gefällt wird[3].

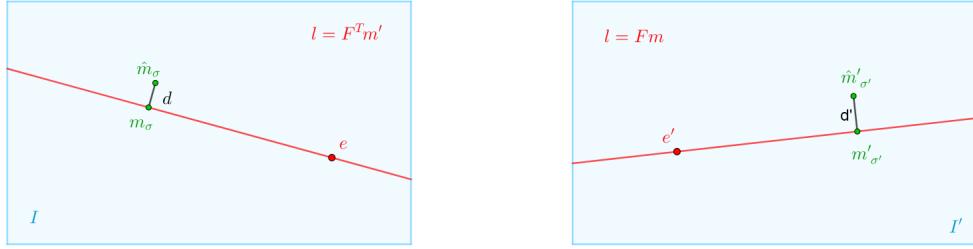


Abbildung 5.14: Die Abbildung zeigt die zwei korrespondierenden Epipolarlinien \hat{l} und \hat{l}' mit den gesuchten Punkten \hat{m}_σ und $\hat{m}'_{\sigma'}$

Jedes andere Punktpaar auf \hat{l} und \hat{l}' würde den *Epipolar-Constraint* erfüllen, jedoch minimieren nur $m_{\sigma\perp}$ und $m'_{\sigma'\perp}$ die quadratischen Distanzen $d(m_\sigma, \hat{m}_\sigma)^2$ und $d(m'_{\sigma'}, \hat{m}'_{\sigma'})^2$ in der Kostenfunktion C . Gesucht wird also der geringste Abstand von m_σ zu \hat{l} und $m'_{\sigma'}$ zu \hat{l}' . Die Kostenfunktion C kann dem entsprechend umformuliert werden in

$$C(m, m') = d(\hat{m}_\sigma, \hat{l})^2 + d(\hat{m}'_{\sigma'}, \hat{l}')^2 \quad (5.7)$$

Aus allen möglichen Epipolarlinien, welche \hat{l} und \hat{l}' annehmen können, wird immer der senkrechte Abstand von m_σ und $m'_{\sigma'}$ zur jeweiligen Epipolarlinie berechnet. jedoch soll genau das Epipolarlinienpaar gewählt werden, welches die Kostenfunktion C minimal werden lässt[3].

Im ersten Schritt werden die Epipolarlinien Parametrisiert, so dass eine Linie als $\hat{l}(t)$ geschrieben werden kann. Durch die Parametrisierung der Epipolarlinien, kann die Kostenfunktion C als Funktion von t umformuliert werden.

$$C(m_\sigma, m'_{\sigma'}) = d(m_\sigma, \hat{l}(t))^2 + d(m'_{\sigma'}, \hat{l}'(t))^2 \quad (5.8)$$

Um zu verhindern dass Bildpunkte, die mit dem Epipol des anderen Bildes korrespondieren Bei Bildpunkten, welche mit dem Epipol des anderen Bildes korrespondieren, würde sich der rückprojizierte Objektpunkt M_δ auf der Basislinie der zwei Projektionszentren befinden. Eine Rekonstruktion des Punktes wäre nicht möglich[3]. Um diesen Fall zu vermeiden, werden die Bildpunkte m_σ und $m'_{\sigma'}$ mit jeweils einer Matrix T und T' in den Ursprung $(0, 0, 1)^T$ verschoben.

$$T = \begin{bmatrix} 1 & 0 & -m_{x\sigma} \\ 0 & 1 & -m_{y\sigma'} \\ 0 & 0 & 1 \end{bmatrix} \rightsquigarrow \bar{m}_\tau = T \cdot m_\tau \quad (5.9)$$

$$T' = \begin{bmatrix} 1 & 0 & -m'_{x\sigma'} \\ 0 & 1 & -m'_{y\sigma'} \\ 0 & 0 & 1 \end{bmatrix} \rightsquigarrow \bar{m}'_{\tau'} = T' \cdot m'_{\tau'} \quad (5.10)$$

Die Fundamentalmatrix F wird ebenfalls mit T und T' transformiert, so dass sie an die verschobenen Punkte \bar{m}_σ und $\bar{m}'_{\sigma'}$ angepasst ist.

$$\bar{F} = T'^{-T} F T^{-1} \quad (5.11)$$

Der rechte und linke Kern von \bar{F} ergeben die Epipole \bar{e} und \bar{e}' . Angenommen f und f' seien genau 0, so liegen die Epipole $e = (1, 0, f)^T$ und $e' = (1, 0, f')^T$ im unendlichen. 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[3].

$$\bar{F} = \begin{pmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{pmatrix} \quad (5.12)$$

$$\begin{pmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ f \end{pmatrix} = \begin{pmatrix} ff'd + (-ff'd) \\ -fb + fb \\ -fd + fd \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (5.13)$$

$$(1 \ 0 \ f') \cdot \begin{pmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{pmatrix} = \begin{pmatrix} ff'd + (-ff'd) \\ -f'c + f'c \\ -f'd + f'd \end{pmatrix} = (0 \ 0 \ 0) \quad (5.14)$$

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 $Re = (1, 0, e_3) = (1, 0, f)$ und $R'e' = (1, 0, e'_3) = (1, 0, f')$ rotieren.

$$R = \begin{bmatrix} e_1 & e_2 & 0 \\ -e_2 & e_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.15)$$

$$R' = \begin{bmatrix} e'_1 & e'_2 & 0 \\ -e'_2 & e'_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (5.16)$$

\bar{F} wird dann nochmals mit $\bar{F}_{Rot} = R'F R^T$ ersetzt. Die Einträge in \bar{F}_{Rot} haben nun die Form wie in Gleichung 5.12, mit $f = e_3$, $f' = e'_3$, $a = \bar{F}_{Rot,22}$, $b = \bar{F}_{Rot,23}$, $c = \bar{F}_{Rot,32}$ und $d = \bar{F}_{Rot,33}$. Verläuft eine Epipolarlinie 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 Epipolarlinie $l(t)$.

$$\hat{l}(t) = \begin{pmatrix} 0 \\ t \\ 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ f \end{pmatrix} = \begin{pmatrix} tf \\ 1 \\ -t \end{pmatrix} \quad (5.17)$$

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

$$d(\bar{m}_\sigma, \hat{l}(t))^2 = \frac{t^2}{1 + (tf)^2} \quad (5.18)$$

Für die Herleitung von Gleichung 5.18 wird eine Gerade zunächst in Koordinatenform Dargestellt.

$$Ax + By - C = 0 \quad (5.19)$$

Die Selbe Gerade in Normalform ausgedrückt lautet:

$$\vec{n} \cdot (\vec{x} - \vec{p}) = 0 \quad (5.20)$$

$$\begin{pmatrix} A \\ B \end{pmatrix} \cdot (\vec{x} - \vec{p}) = 0 \quad (5.21)$$

Der Abstand $\| \vec{v} \|$ eines Punktes zur Geraden 5.21 kann folgendermaßen berechnet werden.

$$\vec{v} = \frac{\vec{p} \cdot \vec{n}}{\vec{n} \cdot \vec{n}} \cdot \vec{n} \rightsquigarrow \frac{-C}{A^2 + B^2} \cdot \begin{pmatrix} A \\ B \end{pmatrix} \quad (5.22)$$

$$\| \vec{v} \| = \frac{|\vec{p} \cdot \vec{n}|}{\| \vec{n} \|^2} \cdot \| \vec{n} \| \rightsquigarrow \| \vec{v} \| = \frac{|\vec{p} \cdot \vec{n}|}{\| \vec{n} \|} \quad (5.23)$$

$$\Rightarrow |C| = |\vec{p} \cdot \vec{n}| \quad (5.24)$$

$$\Rightarrow |\sqrt{A^2 + B^2}| = \| \vec{n} \| \quad (5.25)$$

$$\| \vec{v} \| = \frac{|C|}{\sqrt{A^2 + B^2}} \quad (5.26)$$

Werden nun A, B und C mit den Werten der Geraden $(tf, 1, -t)^T$ ersetzt, kann Gleichung 5.18 rekonstruiert werden.

$$A = tf, B = 1, C = -t, \vec{v} = d \quad (5.27)$$

$$d^2 = \frac{t^2}{\sqrt{((tf)^2 + 1^2)^2}} = \frac{t^2}{(tf)^2 + 1^2} = \frac{t^2}{1 + (tf)^2} \quad (5.28)$$

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

$$l'(t) = F(0, t, 1)^T = (-f'(ct + d), at + b, ct + d)^T. \quad (5.29)$$

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

$$d(\bar{m}_\sigma, \hat{l}(t))^2 = \frac{(ct + d)^2}{(at + b)^2 + f'^2(ct + d)^2} \quad (5.30)$$

Die Kostenfunktion C kann jetzt in eine Funktion $s(t)$ umformuliert werden.

$$s(t) = \frac{t^2}{1 + (tf)^2} + \frac{(ct + d)^2}{(at + b)^2 + f'^2(ct + d)^2} \quad (5.31)$$

Ein Minimum für $s(t)$ kann jetzt beispielsweise durch Bestimmung der Minimas und Maximas mit $s'(t) = 0$ gefunden werden.

$$s'(t) = \frac{2t}{(1 + f^2 t^2)^2} - \frac{2(ad - bc)(at + b)(ct + t)}{((at + b)^2 + f'^2(ct + d)^2)^2} \quad (5.32)$$

Werden die beiden Terme in $s'(t)$ auf einen gemeinsamen Nenner gebracht und der Zähler dann gleich Null gesetzt, ergibt sich der folgende Ausdruck $g(t)$

$$g(t) = t((at+b)^2 + f'^2(ct+d)^2)^2 - (ad-bc)(1+f^2t^2)^2(at+b)(ct+d) \quad (5.33)$$

Funktion $g(t)$ ist ein Polynom vom Grad 6. Das Minimum für $s(t)$ ergibt sich aus einer der 6 möglichen Lösungen für t aus $g(t)$. Für die Bestimmung 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 einsetzen in $s(t)$ den kleinsten Wert ergibt, ist das gesuchte Minimum t_{min} .

Mit t_{min} können die Epipolarlinien $\hat{l}(t) = (tf, 1, -t)$ und $\hat{l}'(t) = F(0, t, 1)^T$ durch einsetzen von t_{min} berechnet werden. Danach werden die zwei neuen Punkte \hat{m}_σ und $\hat{m}'_{\sigma'}$ auf den Epipolarlinien bestimmt. $\hat{m}_{\sigma Rot}$ und $\hat{m}'_{\sigma' Rot}$ sind die Punkte auf der Epipolarlinie welche dem Ursprung am nächsten sind. Der Punkt, welcher vom Ursprung aus am nächsten auf einer Linie (λ, μ, v) liegt, kann mit $(-\lambda \cdot v, -\mu \cdot v, \lambda^2 + \mu^2)$ berechnet werden.

$$\hat{l} = (tf, 1, -t) \quad \hat{m}_{\sigma Rot} = (-(tf) \cdot v, -1 \cdot v, (tf)^2 \cdot 1^2) \quad (5.34)$$

$$(5.35)$$

Nachdem zu beiden Linien \hat{l} und \hat{l}' der jeweils nächste Punkte $\hat{m}_{\sigma Rot}$ und $\hat{m}'_{\sigma' Rot}$ vom Ursprung aus gefunden wurden, werden diese nun mit T , T' , R und R' wieder an ihre Ausgangsposition zurück transformiert.

$$\hat{m} = T^{-1}R^T\hat{m}_{\sigma Rot} \quad (5.36)$$

$$\hat{m}' = T'^{-1}R'^T\hat{m}'_{\sigma' Rot} \quad (5.37)$$

Für auf diese weise neu berechneten korrespondierenden Punkte ist der *Epipolar-Constraint* 3.18 erfüllt und es ist gewährleistst, dass sich ihre jeweiligen Rückprojektionen in einem Punkt im Raum treffen.

Für die Rückprojektion der einzelnen Bildpunkte wurde ein lineares Triangulationsverfahren gewählt[3]. Für die Rückprojektion werden pro korrespondierendem Punktpaar zunächst die Projektionsgleichungen $\hat{m}_\sigma = P\hat{M}_\delta$ und $\hat{m}'_{\sigma'} = P'\hat{M}'_\delta$ aufgestellt. Diese werden so in eine Koeffizientenmatrix A eingetragen dass gilt $A \cdot x = 0$. Durch die Verwendung des Kreuzproduktes, wird die Homogene Komponente eliminiert[3].

$$\hat{m} \times (P\hat{M}) = 0 \quad (5.38)$$

$$\hat{m}' \times (P'\hat{M}') = 0 \quad (5.39)$$

Was ausgeschrieben für \hat{m} und \hat{m}' zu den folgenden drei Gleichungen führt. $x \times (PX) = 0$ ergibt ausgeschrieben

$$x(p^{3T}X) - (p^{1T}X) = 0 \quad (5.40)$$

$$y(p^{3T}X) - (p^{2T}X) = 0 \quad (5.41)$$

$$x(p^{2T}X) - y(p^{1T}X) = 0 \quad (5.42)$$

p^{iT} bezeichnet hier jeweils die Reihen der Projektionsmatrix P beziehungsweise P' . Zwei der drei Gleichungen sind linear unabhängig und werden in die Koeffizientenmatrix A geschrieben

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix} \quad (5.43)$$

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix} \cdot (X_1) \quad (5.44)$$

Die zwei Wege eine solche Matrix zu lösen wurden in Kapitel 3 vorgestellt. Zum einen kann die Inhomogenene Methode angewandt werden und der Kern dieser Koeffizientenmatrix bestimmt werden, oder es kann das homogene Verfahren angewandt werden, welches die Methode der Singulärwertzerlegung beinhaltet.

Die rekonstruierten Punkte $M_{\delta 0}$ sind bis auf einen Skalierungsfaktor genau bestimmt. Die Abbildungen 5.15 und 5.16 zeigen die aus den Bildern 5.4 und 5.3 rekonstruierten Punkte im Raum. Der rote Punkt steht für die Position von C , der grüne für die Position von C' . Die blauen Punkte sind die rekonstruierten Punkte, die beiden Bilder 5.4 und 5.3.

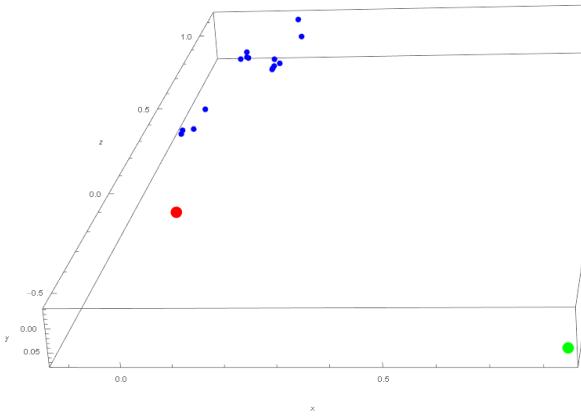


Abbildung 5.15: Rekonstruierte Szene, unskaliert in Pixeleinheiten

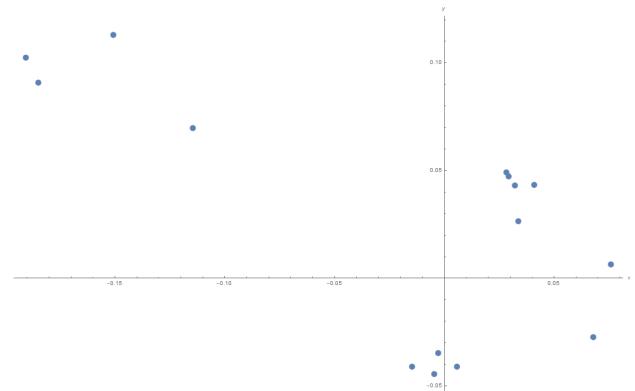


Abbildung 5.16: Rekonstruierte Szene, unskaliert, in Pixeleinheiten und in einem 2D-Plot angezeigt

5.5 Ergebnisse einer Stereoanalyse mit Kameras unterschiedlicher Auflösung

Für den Test, ob Szeneriekonstruktion im Realbeispiel auch mit unterschiedlichen Kameraauflösungen funktioniert, wurde die Kameramatrix K' von C' künstlich skaliert. Wie aus Kapitel 2, hat diese die Form

$$K' = \begin{bmatrix} k_x\zeta & s & V_{\sigma,x} \\ 0 & k_y\zeta & V_{\sigma,y} \\ 0 & 0 & 1 \end{bmatrix} \quad (5.45)$$

Um die Auflösung von K' zu verändern, werden die Matrixeinträge $k_x\zeta$ und $k_y\zeta$ jeweils noch um eine beliebige Skalierung erweitert. Als Beispiel wurde K' fünf mal unterschiedlich skaliert und zwar mit den Verhältnissen $[2 : 2]$, $[5 : 2]$, $[1 : 2]$ und $[1.2 : 2.3]$.

$$\begin{aligned} K'_{[2:2]} &= \begin{bmatrix} k_x\zeta \cdot 2 & s & V_{\sigma,x} \cdot 2 \\ 0 & k_y\zeta \cdot 2 & V_{\sigma,y} \cdot 2 \\ 0 & 0 & 1 \end{bmatrix} \\ K'_{[5:2]} &= \begin{bmatrix} k_x\zeta \cdot 5 & s & V_{\sigma,x} \cdot 5 \\ 0 & k_y\zeta \cdot 2 & V_{\sigma,y} \cdot 2 \\ 0 & 0 & 1 \end{bmatrix} \\ K'_{[1:2]} &= \begin{bmatrix} k_x\zeta \cdot 1 & s & V_{\sigma,x} \cdot 1 \\ 0 & k_y\zeta \cdot 2 & V_{\sigma,y} \cdot 2 \\ 0 & 0 & 1 \end{bmatrix} \\ K'_{[1.2:2.3]} &= \begin{bmatrix} k_x\zeta \cdot 1.2 & s & V_{\sigma,x} \cdot 1.2 \\ 0 & k_y\zeta \cdot 2.3 & V_{\sigma,y} \cdot 2.3 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$

Von den detektierten korrespondierenden Bildpunkten des SURF-Algorithmus wurden die Bildpunkte des zweiten Bildes von C' auch um die selben Verhältnisse skaliert.

$$m_{\sigma[2:2]} = m_{\sigma} \cdot \begin{pmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5.46)$$

$$m_{\sigma[5:2]} = m_{\sigma} \cdot \begin{pmatrix} 5 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5.47)$$

$$m_{\sigma[1:2]} = m_{\sigma} \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5.48)$$

$$m_{\sigma[1.2:2.3]} = m_{\sigma} \cdot \begin{pmatrix} 1.2 & 0 & 0 \\ 0 & 2.3 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad (5.49)$$

Der Szenenrekonstruktionsalgorithmus wurde für jede Kameraauflösung getestet. Die vier Lösungen der Bestimmung der extrinsischen Parameter sind in den Abbildungen 5.17, 5.18, 5.19 und 5.20 zu sehen. Abbildung ?? zeigt die vier Lösungen bei gleicher Kameraauflösung. Zu beobachten ist, dass sich die Lösungen bei unterschiedlichen Auflösungen nicht unterscheiden. Die geringen Abweichungen in den Nachkommastellen sind auf die Ungenauigkeiten der Bilddaten zurückzuführen.

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

Abbildung 5.17: Zeigt die rekonstruierte Matrix T' bei unveränderter Auflösung. Die Auflösungen von C_δ und C'_δ sind die selben.

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

Abbildung 5.19: Zeigt die rekonstruierte Matrix T' wenn K' mit einem Verhältnis von $[1 : 2]$ skaliert wurde

Die Abbildungen 5.21, 5.22 und 5.23 zeigen einmal die Rekonstruierten Punkte in einem 3D-Plot und daneben den 2D-Plot. Auch hier kann beobachtet werden, dass es immer zu den gleichen Szenen. (Soll ich die Abweichungen noch irgendwie erwähnen????) In Abbildung 5.22 ist die Rekonstruktion in einem links drehendem Koordinatensystem geplottet, weshalb die Rekonstitution spiegelverkehrt wirkt.

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

Abbildung 5.18: Zeigt die rekonstruierte Matrix T' wenn K' mit einem Verhältnis von $[5 : 2]$ 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 5.20: Zeigt die rekonstruierte Matrix T' wenn K' mit einem Verhältnis von $[1.2 : 2.3]$ skaliert wurde

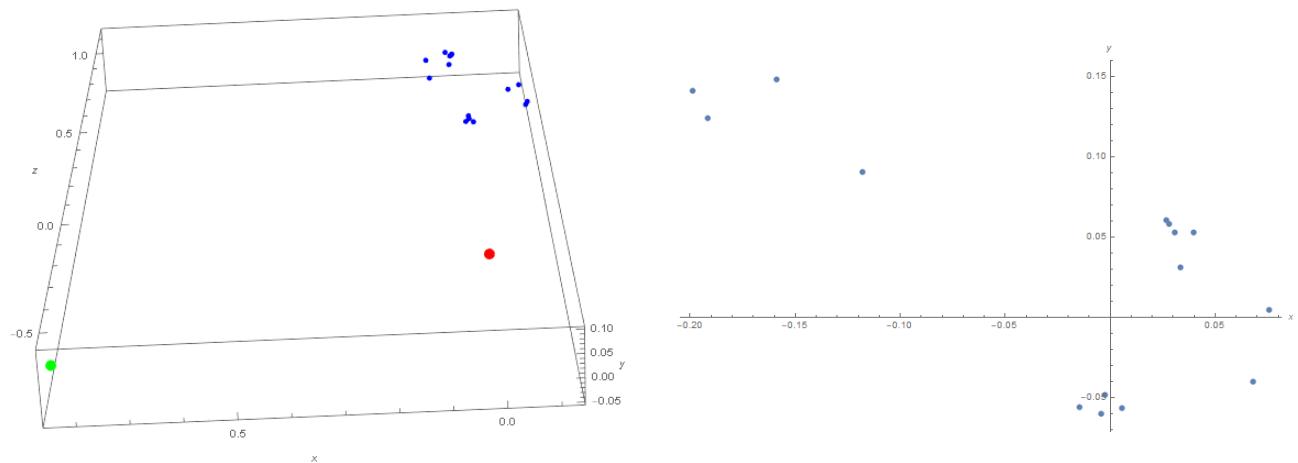


Abbildung 5.21: Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[5 : 2]$ skaliert wurde

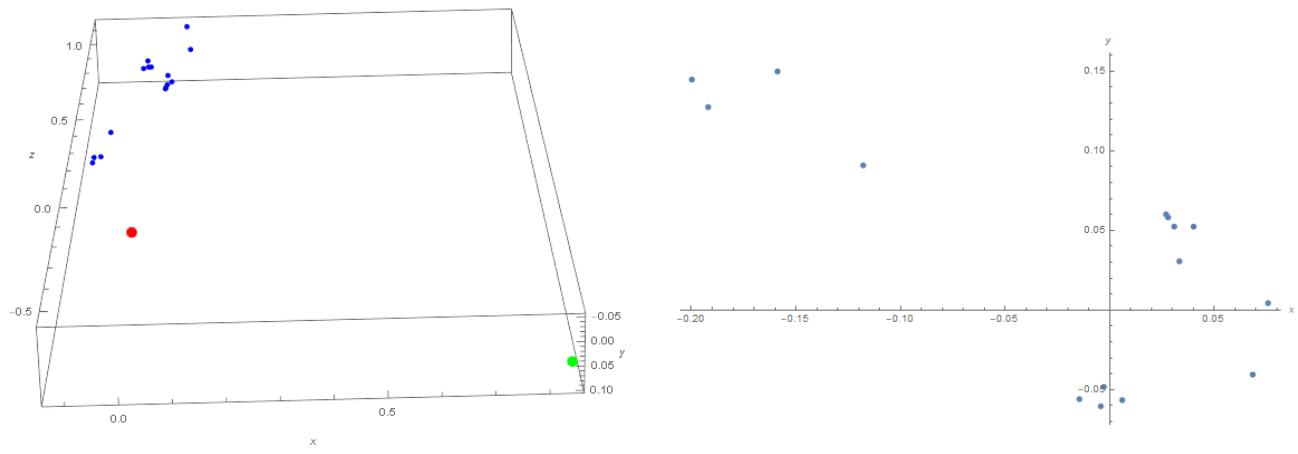


Abbildung 5.22: Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[1 : 2]$ skaliert wurde

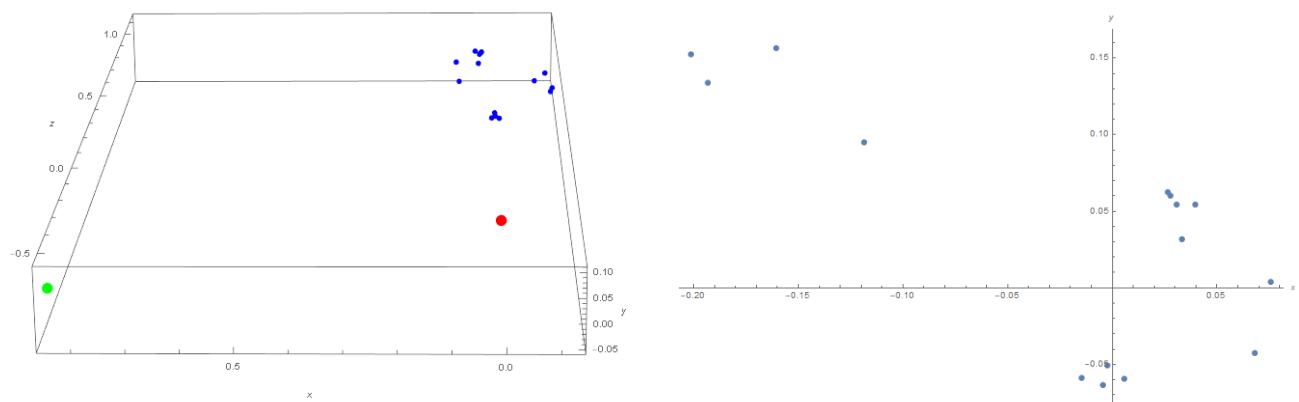


Abbildung 5.23: Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[1.2 : 2.3]$ skaliert wurde

6 Vergleich mit anderen Computer Vision Applikationen

Ein weit verbreiteter Ansatz der Stereobildanalyse basiert auf zuvor Rektifizierten Bildern. Rektifizierte Bilder zeichnen sich dadurch aus, dass die jeweiligen Epipole der Bilder ins unendliche projiziert werden, was dazu führt, dass die jeweiligen Epipolarlinien der Bilder parallel zueinander verlaufen. Anschließend werden die Epipole noch so rotiert, dass die Epipolarlinien in einheitlichen Scanlinien über beide Bilder verlaufen. In Abbildung 6.1 ist das graphisch dargestellt [28, 29, 30, 31, 15].

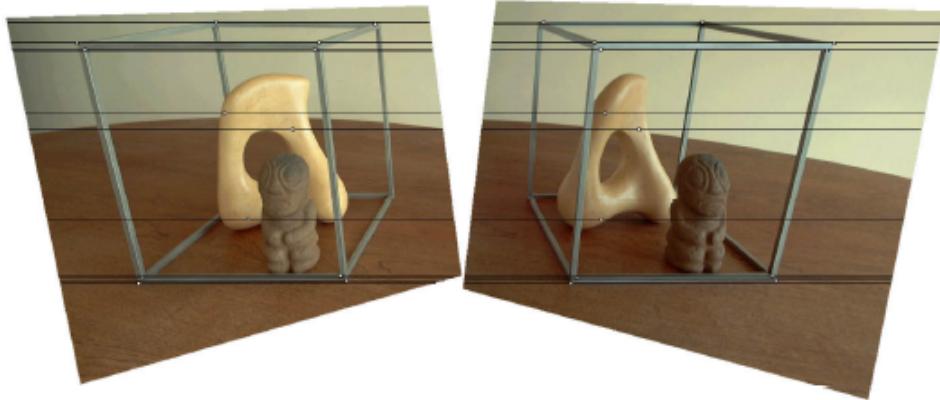


Abbildung 6.1: Beispiel eines rektifizierten Bildes. Die Epipole beider Bilder sind ins unendliche projiziert worden. Des Weiteren wurden die Epipole so rotiert, dass die Epipolarlinien zu einheitlichen Scanlinien über beide Bilder verlaufen. Quelle: [30]

Anhand der so entstandenen Scanlinien, wird die Suche nach weiteren korrespondierenden Punkten auf eine eindimensionale Suche beschränkt, da nur noch entlang der entsprechenden korrespondierenden Epipolarlinie gesucht wird. Die Rektifizierung, macht es somit möglich die Bildpunkte kompletter Bilder mit relativ geringem Rechenaufwand zu rekonstruieren [30, 28, 29]. Jedoch verlangen viele Rektifizierungsansätze als Voraussetzung, dass das verwendetet Stereobildpaar die selbe Auflösung besitzt.

Der in dieser Arbeit entwickelte Szenenrekonstruktionsalgorithmus wurde so entwickelt, dass eine Rektifizierung der Bilder, nicht notwendig wird. Der Grund dafür ist, dass ein Algorithmus entsteht, welcher auch mit Bildern unterschiedlicher Kameraauflösungen eine erfolgreiche Rekonstruktion vollbringt.

Im Verlauf des Kapitels wird zunächst der Arbeitsprozess der Stereoanalyse auf Basis von Bildrektifizierungen erläutert. Anschließend wird ein Rektifizierungsalgorithmus nach [30] vorgestellt werden, welcher nur durch Vorwissen von 9 korrespondierenden Punkten und der daraus resultierenden Fundamentalmatrix eine Rektifizierung zweier Bilder ermöglicht. Der Rektifizierungsalgorithmus wurde anhand des synthetischen Stereoaufbaus aus Kapitel 4 implementiert. Mit dem so entstandenen Algorithmus werden zunächst zwei Bilder gleicher Auflösung rektifiziert und danach werden zwei Tests mit Bildern unterschiedlicher Auflösung aufgezeigt und analysiert.

6.1 Szenenrekonstruktion mit Rektifizierung

Da bestimmte Formen der Rektifizierung keine vorherige Kalibrierung der Kameras benötigen, wird diese Methode in den meisten gängigen Echtzeit-Szenenrekonstruktionen eingesetzt. [28, 29, 32]. Der

Arbeitsprozess für die Szenenrekonstruktion auf Basis von rektifizierten Bildern, welcher in Abbildung 6.2 schematisch dargestellt ist, unterscheidet sich etwas zu den bereits bekannten Arbeitsprozessen, welche in den Abbildungen 4.1 und 5.1 zusammengefasst sind.

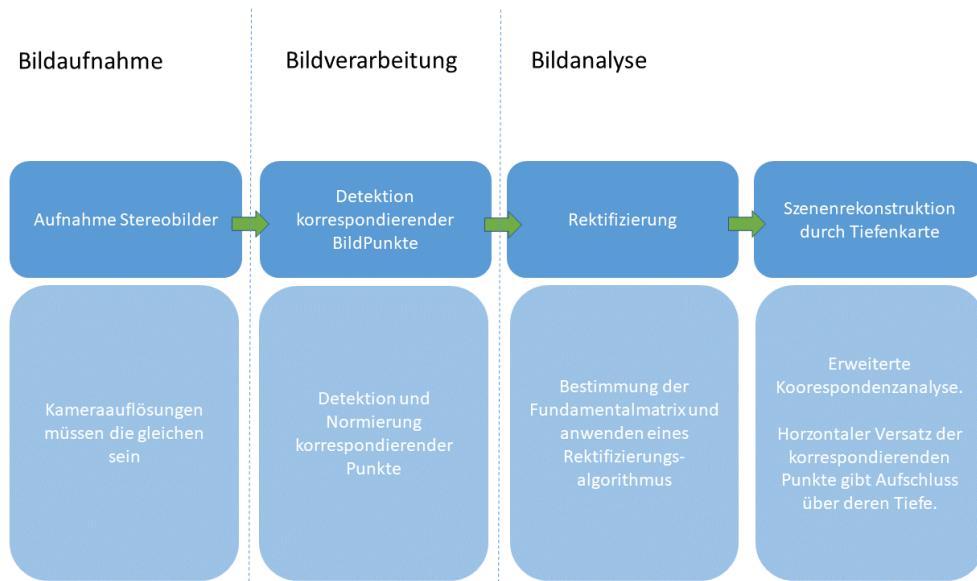


Abbildung 6.2: Ablaufdiagramm der Szenenrekonstruktion basierend auf einem Rektifizierungsansatz

Der entscheidende Unterschied zwischen den beiden Ansätzen liegt im Abschnitt der Bildanalyse. Es wird davon ausgegangen, dass die intrinsischen Kameraparameter im einzelnen nicht bekannt sind. Der Ansatz beinhaltet auch keinen Kalibrierungsschritt, in welchem intrinsische oder extrinsische Kameraparameter bestimmt werden sollen. Das Ziel, des in Abbildung 6.2 gezeigten Ansatzes, ist es eine 3D-Szene mit möglichst wenig Informationen aus einem Stereobildpaar zu rekonstruieren [30, 29, 28, 15].

Zunächst werden acht bis neun korrespondierende Punkte in einem Stereobildpaar detektiert. Aus den zuvor noch, wie in Kapitel 5 beschriebenen, normierten korrespondierenden Punkten wird eine Fundamentalmatrix F bestimmt. Aus den korrespondierenden Punkten und der Fundamentalmatrix F wird ein Rektifizierungsalgorithmus, wie beispielsweise in Kapitel 6.2, angewandt, welcher zwei Stereobilder so transformiert, dass ihre Epipolarlinien zu horizontal ausgerichteten Scanlinien werden, wie in Abbildung 6.1 zu sehen ist.

Mit Hilfe dieser entstandenen Scanlinien, wird die Suche nach weiteren korrespondierenden Punkten von einer zweidimensionalen auf eine eindimensionale Suche runter gebrochen, da sich diese jeweils entlang der zum Punkte entsprechenden korrespondierenden Epipolarlinien befinden [30, 28, 29].

Es wird davon ausgegangen, dass die Bilder aus zwei verschiedenen Blickwinkeln aufgenommen wurden. Nach der Rektifizierung ist zwischen zwei korrespondierenden Punkten ein horizontaler Versatz zu verzeichnen, wie in Abbildung 6.3 in den ersten beiden Bildern zu sehen ist. Dieser Versatz entsteht durch den Abstand der beiden Kameras zueinander. Der Versatz zwischen den beiden korrespondierenden Punkten wird als Disparität bezeichnet. Je weiter ein Objekt von den Kameras weg ist, desto geringer ist die zu verzeichnende Disparität [29].

Bei der Disparität handelt es sich um einen numerischen Wert. Diesem Wert kann je nach Größe ein Grauwert zugeordnet werden. Je größer eine Disparität zwischen zwei Punkten ist, desto heller wird der entsprechende Grauwert gewählt und je geringer die Disparität desto dunkler wird der Grauwert. Anhand der zugeordneten Grauwerte kann ein Bild generiert werden, welches auch als Disparitätskarte oder Tiefenkarte bezeichnet wird [29, 33]. Diese Disparitätskarte gibt ein Maß für die Tiefe der Objekte in der Szene an [29, 33].



Abbildung 6.3: In den oberen beiden Bildern sind in blau zwei korrespondierende Epipolarlinien zu sehen, die zusammen eine Scanlinie bilden. Im Bild darunter ist eine aus den Disparitäten zweier Bilder zueinander entstandene Disparitäts beziehungsweise Tiefenkarthe zu sehen. Quelle: [29]

Voraussetzung für dieses Verfahren ist es jedoch, dass beide Bilder mit Kameras gleicher Auflösung aufgenommen wurden. Bei unterschiedlichen Auflösungen ist es mit dem in 6.2 aufgezeigten Rektifizierungsalgorithmus prinzipiell möglich Bilder unterschiedlicher Auflösung zu rektifizieren. Dabei wird je nach dem das größere Bild gestaucht oder das kleinere Bild aufgezogen, so dass die Epipolarlinien beider Bilder wieder zu einheitlichen Scanlinien werden.

Jedoch kann es dadurch gerade bei starken Auflösungsunterschieden zu Ungenauigkeiten in der erweiterten Korrespondenzanalyse kommen, da ein Pixel in einem Bild je nach dem deutlich größer definiert ist als im anderen Bild. Mögliche Lösung hierfür wäre es zum Beispiel die Pixelgrößen vor der erweiterten Korrespondenzanalyse über Nachbarschaftsoperationen so anzugeleichen, dass mehrere benachbarte Pixel des größeren Bildes mit einem Pixel des kleineren korrespondiert.

6.2 Rektifizierung mit Homographien

Im Folgenden wird ein Rektifizierungsalgorithmus nach *Zhang*[30] vorgestellt. Diese recht aufwendige Art der Rektifikation zeichnet sich durch minimale Voraussetzungen an die Ursprungsbilder aus. Alle notwendigen Informationen zur Rektifikation werden aus der Fundamentalmatrix gewonnen. Zusätzlich zur Fundamentalmatrix muss noch die Lage der jeweiligen Epipole bekannt sein[15]. Anhand des entstandenen Algorithmus wird dann im Nachhinein getestet, ob Bilder unterschiedlicher Auflösung richtig rektifiziert werden können und was die Ergebnisse für den weiteren Verlauf der Szenenrekonstruktion bedeuten könnten.

Für den Rektifizierungsansatz wird pro Bild eine Homographiematrix H und H' aufgestellt. Die Rektifizierung aller Bildpunkte m_σ und $m'_{\sigma'}$ erfolgt durch Gleichung 6.2

$$\bar{m}_\sigma = H m_\sigma \quad (6.1)$$

$$\bar{m}'_{\sigma'} = H' m'_{\sigma'} \quad (6.2)$$

Die Fundamentalmatrix, welche aus den Rektifizierten korrespondierenden Punkte resultiert, wird mit \bar{F} bezeichnet[30, 15]:

$$\bar{m}'_{\sigma'}^T \bar{F} \bar{m}_\sigma = 0 \quad (6.3)$$

$$\rightsquigarrow m'^T_{\sigma'} H'^T \bar{F} H m_\sigma = 0 \quad (6.4)$$

$$\rightsquigarrow F = H'^T [i] \times H \quad (6.5)$$

Die Homographien H und H' werden in die projektiven Komponenten H_p und H'_p und die affinen Komponente H_a und H'_a unterteilt. Die affine Komponenten wird wiederum in zwei weitere Komponenten unterteilt. H_r steht für eine Ähnlichkeitstransformation und H_s bezeichnet eine Scherungstransformation[30, 15].

$$H = H_a H_p \quad \rightsquigarrow H = H_s H_r H_p \quad (6.6)$$

$$H' = H'_a H'_p \quad \rightsquigarrow H' = H'_s H'_r H'_p \quad (6.7)$$

$$(6.8)$$

H_p bezeichnet die projektiven Komponenten und H_a steht für die affine Komponente. Die affine Komponenten wird wiederum in zwei weitere Komponenten unterteilt. H_r steht für eine Ähnlichkeitstransformation und H_s beinhaltet einen Scherungstransformation. H_p beinhaltet sämtliche projektiven Transformationen, welche dafür sorgen, dass der Epipol e ins Unendliche projiziert wird und die Epipolarlinien parallel zueinander, jedoch noch nicht parallel zur horizontalen Achse sind[30, 15]. H_r ist eine Rotationsmatrix, welche die Epipolarlinien parallel zu horizontalen Achse ausrichtet. H_s ist eine Scherungsmatrix, welche durch Minimierung versucht die durch die Rektifizierung entstandenen projektiven Verzerrungen bestmöglich auszugleichen[30, 15].

Die Reihen der Homographiematrizen H und H' beschreiben drei Linien u , v und w , welche jeweils durch den Epipol verlaufen. Die Linien v und v' sowie w und w' sind korrespondierende Epipolarlinien. Durch diese geometrische Bedingung wird eine Verbindung der beiden Bilder zueinander.

$$H = \begin{bmatrix} u^T \\ v^T \\ w^T \end{bmatrix} = \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & w_c \end{bmatrix} \quad (6.9)$$

$$H' = \begin{bmatrix} u'^T \\ v'^T \\ w'^T \end{bmatrix} = \begin{bmatrix} u'_a & u'_b & u'_c \\ v'_a & v'_b & v'_c \\ w'_a & w'_b & w'_c \end{bmatrix} \quad (6.10)$$

Bevor die Matrizen H und H' in ihre projektiven und affinen Komponenten zerlegt werden, wird die letzte Komponenten w_c und w'_c durch Division eliminiert um somit skaleninvariante Matrizen H und H' zu bekommen[30, 15].

$$H = \begin{bmatrix} u^T \\ v^T \\ w^T \end{bmatrix} = \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & 1 \end{bmatrix} \quad (6.11)$$

$$H' = \begin{bmatrix} u'^T \\ v'^T \\ w'^T \end{bmatrix} = \begin{bmatrix} u'_a & u'_b & u'_c \\ v'_a & v'_b & v'_c \\ w'_a & w'_b & 1 \end{bmatrix} \quad (6.12)$$

Die Matrizen H_p und H'_p beschreiben den projektiven Teil von H und H' . Sie wirken sich auf den projektiven Teil eines Punktes aus und werden dazu verwendet die Epipole ins unendliche zu projizieren[30, 15].

$$H_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ w_a & w_b & 1 \end{bmatrix} \quad (6.13)$$

Für die affinen Komponenten H_a und H'_a gilt:

$$H_a = H \cdot H_p^{-1} = \begin{bmatrix} u_a - v_c w_b & v_c w_a - v_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad (6.14)$$

Des Weiteren gilt für H_a und H'_a , dass jeweils nochmal in eine Rotationsmatrix H_r und H'_r und eine Scherungsmatrix H_s und H'_s zerlegt werden[30, 15].

$$H_a = H_s \cdot H_r \quad (6.15)$$

$$H_r = \begin{bmatrix} v_b - v_c w_b & v_a - v_c w_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad (6.16)$$

$$H_s = \begin{bmatrix} u_a & u_b & u_c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.17)$$

6.2.1 Projektive Transformation

Im Folgenden wird die Herleitung der Matrizen H_p und H'_p beschrieben. Die projektiven Matrizen H_p und H'_p werden von den Linien w und w' , welche durch den Epipol verlaufen bestimmt. w und w' sind nicht willkürlich. Definiert werden sie durch eine Richtung $z = [\lambda \ \mu \ 0]^T$. z soll dabei so gewählt werden, dass die durch die Rektifizierung entstehenden Bildverzerrungen in beiden Bildern minimal bleibt. Die Linien w und w' werden wie folgt definiert.

$$w = [e]_x \cdot z \quad (6.18)$$

$$w' = F \cdot z \quad (6.19)$$

Unter der Minimierung versteht man in diesem Falle, dass versucht wird ein z zu finden, welches $w = (w_a, w_b, w_c)^T$ und $w' = (w'_a, w'_b, w'_c)^T$ so definiert, dass die Einträge w_a und w_b und auch w'_a und w'_b in H_p und H'_p nahezu null sind. Anders ausgedrückt es wird versucht die projektive Matrizen H_p und H'_p so affin wie möglich zu machen[30].

$$H_p = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \\ w_a & w_b & 1 \end{bmatrix} \quad (6.20)$$

Sollte es der Fall sein, dass beispielsweise Epipole e bereits im unendlichen wären, so wären $w_a = 0$ und $w_b = 0$. In diesem Fall wäre eine projektive Transformation für diesen Epipol nicht mehr nötig.

Für die Minimierung wird die Methode der kleinsten Quadrate auf angewandt. Die Methode der kleinsten Quadrate ist ein mathematisches Standardverfahren für eine Ausgleichsrechnung, mit deren Hilfe aus der Menge der Bildpunkten beider Bilder ein Wert für z ermittelt werden soll[34]. Für z gilt mit $z = [\lambda \ \mu \ 0]^T$ bereits die Bedingung, dass es sich um einen Punkt im unendlichen handeln soll. λ und μ , sollen dabei einen Wert annehmen, welcher am nächsten an den Punkteansammlungen beider Bilder ist.

Um ein solchen z zu ermitteln, werden zunächst die Gewichtungen der Punkte beider Bilder benötigt. p_i beinhaltet alle Punkte von Bild eins und p_j beinhaltet alle Punkte von Bild zwei. Ein Punkt $p_{i1} = [p_{i1,u} \ p_{i1,v} \ 1]^T$ aus Bild eins soll zu einem Punkt $p_{i1} = \left[\frac{p_{i1,u}}{w_i} \ \frac{p_{i1,v}}{w_i} \ 1\right]^T$ mit w_i gleich der Gewichtung

$$w_i = w^T p_i \quad (6.21)$$

transformiert werden. Dasselbe soll auch für die Punkte p_j im zweiten Bild geschehen. Sind die Gewichtungen beider Bilder gleich, so ergibt sich keine projektive Verzerrung und sowohl H_p also auch H'_p sind in dem Fall affine Transformationen und die Epipole wären bereits im unendlichen. Angenommen beide Epipole befinden sich noch nicht im unendlichen, so können die Gewichtungen der Punkte beider Bilder nicht gleich sein[30].

Das Ziel ist, die Abweichung der Gewichtungen der Punkte beider Bilder zueinander so gering wie möglich zu machen. Die Rektifizierung wurde anhand des synthetischen Beispiels in Kapitel 4 implementiert. Dem entsprechend bilden die Abbildungen der Eckpunkte des Quaders auf den Sensoren der virtuellen Kameras, die Punkte in p_i und p_j anhand welcher die Rektifizierung durchgeführt werden soll.

Der Wert p_c ergibt sich aus der Mittelung aller verwendeten Punkte eines Bildes $p_c = \frac{1}{n} \sum_{i=1}^n p_i$ und gibt den Bildmittelpunkt an. w_c ist die Gewichtung am Bildmittelpunkt und wird berechnet mit

$$w_c = w^T p_c \quad (6.22)$$

Die Abweichung der Gewichtungen wird bezüglich der Gewichtung des Bildzentrums w_c gemessen.

$$\sum_{i=1}^n \left[\frac{w_i - w_c}{w_c} \right]^2 \quad (6.23)$$

$$\rightsquigarrow \sum_{i=1}^n \left[\frac{w^T(p_i - p_c)}{w^T p_c} \right]^2 \rightsquigarrow \sum_{i=1}^n \left[\frac{w^T(p_i - p_c)(p_i - p_c)^T w}{w^T p_c p_c^T w} \right] \quad (6.24)$$

Vereinfacht lässt sich das auch in einer Matrixgleichung in Form von

$$\frac{w^T P P^T w}{w^T p_c p_c^T w} \quad (6.25)$$

angeben, in welcher für P gilt:

$$P = \begin{bmatrix} p_{1,u} - p_{c,u} & p_{2,u} - p_{c,u} & \dots & p_{i,u} - p_{c,u} \\ p_{1,v} - p_{c,v} & p_{2,v} - p_{c,v} & \dots & p_{i,v} - p_{c,v} \\ 0 & 0 & \dots & 0 \end{bmatrix} \quad (6.26)$$

Für die Punkte p_j in Bild zwei ergibt sich dann entsprechend die Matrixgleichung:

$$\frac{w'^T P' P'^T w'}{w'^T p'_c p'_c^T w'} \quad (6.27)$$

w und w^T werden nun noch mit ihren Definitionen aus den Gleichungen 6.18 und 6.19 ersetzt und die Gleichungen 6.25 und 6.27 als Summiert und als eine Funktion von z ausgedrückt[30].

$$\frac{z^T [e]_x^T P P^T [e]_x z}{z^T [e]_x^T p_c p_c^T [e]_x z} + \frac{z^T F^T P' P'^T F z}{z^T F^T p'_c p'_c^T F z} \quad (6.28)$$

Gleichung 6.28 wird noch vereinfach mit:

$$A = [e]_x^T P P^T [e]_x \quad (6.29)$$

$$B = [e]_x^T p_c p_c^T [e]_x \quad (6.30)$$

$$A' = F^T P' P'^T F \quad (6.31)$$

$$B' = F^T p'_c p'_c^T F \quad (6.32)$$

$$\rightsquigarrow \frac{z^T A z}{z^T B z} + \frac{z^T A' z}{z^T B' F z} \quad (6.33)$$

Da bereits fest gesetzt ist dass die dritte Komponente von z gleich 0 sein wird, wird z im Folgenden als zweidimensionaler vektor mit $z = \begin{pmatrix} \lambda \\ \mu \end{pmatrix}$ dargestellt. A, B, A' und B' sind 3x3-Matrizen, von denen dann nur noch der erste 2×2 - Block wichtig ist.

Für eine 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[30].

$$z = \frac{\frac{\hat{z}_1}{\|\hat{z}_1\|} + \frac{\hat{z}_2}{\|\hat{z}_2\|}}{2} \quad (6.34)$$

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 werden als eine Funktion von $f(z)$ definiert. Matrix A wird mit der Choleskyzerlegung[35] in zwei höhere Dreiecksmatrizen zerlegt $A = D^T D$. Dies geht nur da

A nachweislich eine symmetrische und positiv-definite Matrix ist[36, 35]. Des Weiteren wird definiert, dass $y = Dz$ ist und $f(z)$ wird dann zu $\hat{f}(y)$ [30].

$$A = D^T D \quad (6.35)$$

$$y = Dz \rightsquigarrow z = D^{-1}y \quad (6.36)$$

$$f(z) = \frac{z^T B z}{z^T A z} \quad (6.37)$$

$$\rightsquigarrow f(z) = \frac{z^T B z}{z^T D^T D z} \quad (6.38)$$

$$\hat{f}(y) = \frac{y^T D^{-T} B D^{-1} y}{y^T y} \quad (6.39)$$

Da y bis auf einen Skalierungsfaktor bestimmt ist, kann angenommen werden, dass $\|y\| = 1$ gilt. $\hat{f}(y)$ ist maximiert, wenn y gleich dem Eigenvektor von $D^{-T} B D - 1$ ist, welcher mit dem größten Eigenwert von $D^{-T} B D - 1$ assoziiert wird[30]. Für \hat{z}_1 ergibt sich $\hat{z}_1 = D^{-1}y$. Exakt das selbe Verfahren wird für die Bestimmung von z_2 mit $\frac{z^T B' z}{z^T A' z}$ angewandt[30].

Sind z_1, z_2 und durch Mittelung der beiden eine erste Schätzung für z gefunden, so kann ein Wert für z gesucht werden, welcher noch näher an ein optimales Ergebnis heranreicht. Beide Lösungen z_1 und z_2 , werden in die Funktion $f(z)$ eingesetzt und es wird ein gemeinsames Minimum gesucht[30]. So kann iterativ eine optimale Lösung für z gefunden werden. Ist der Wert für z bestimmt, so kann dieser die Gleichungen 6.18 und 6.19 eingesetzt werden und w beziehungsweise w' bestimmt werden. Die Einträge der Matrizen H_p und H'_p wären mit $w = (w_a, w_b, w_c)^T$ und $w' = (w'_a, w'_b, w'_c)^T$ bestimmt[30].

Auf diese Weise wurden im synthetischen Beispiel die Matrizen H_p und H'_p bestimmt und auf die Abbildungen des Quaders angewandt. Abbildung 6.4 zeigt die unrektifizierten Abbildungen der Quader in den Kameras. In grün ist die Abbildung des Quaders in C zu sehen und in rot ist die Abbildung des Quaders in C' . Abbildung 6.5 zeigt in blau die Epipolarlinien beider Abbildungen, welche sich in den jeweiligen Epipolen treffen.

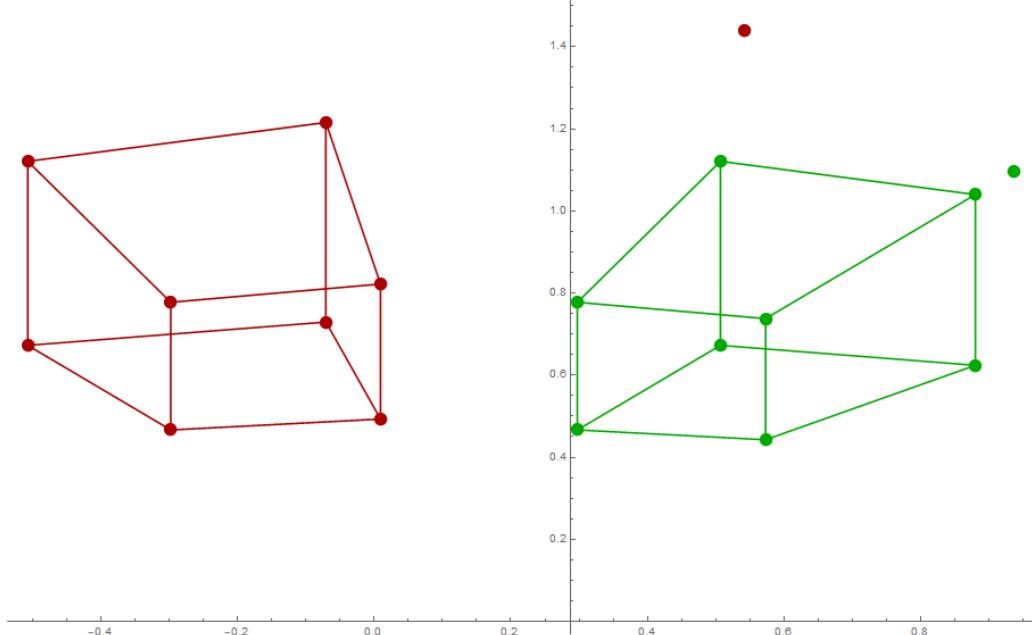


Abbildung 6.4: Aufnahmen zweier Kameras mit den selben Auflösungen, Kamera eins(Grün) und Kamera(rot) zwei gelten jeweils $\zeta = 1$

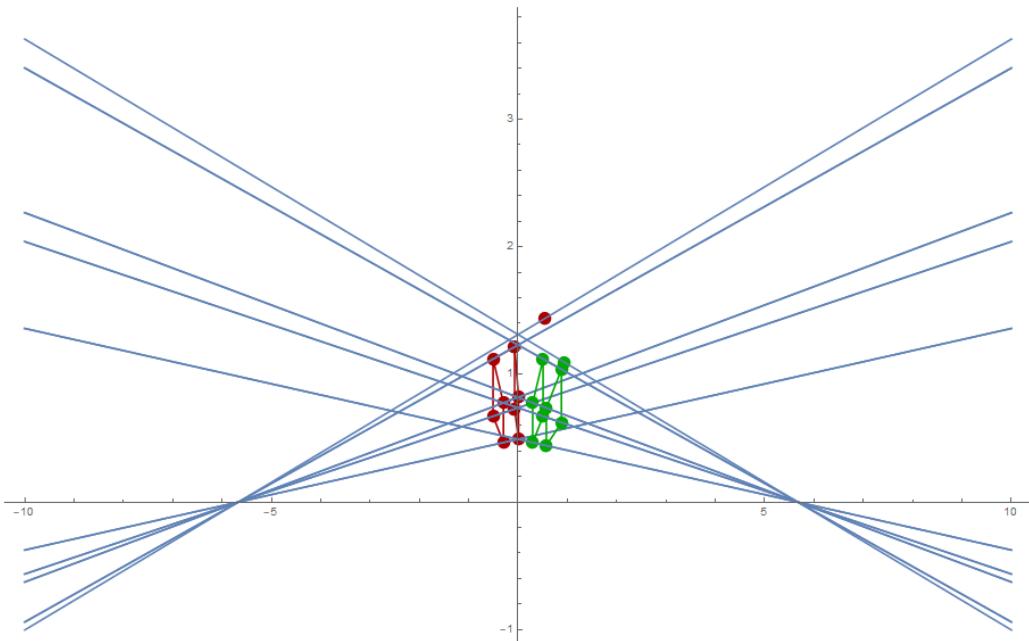


Abbildung 6.5: Epipole für Kamera eins und Kamera zwei vor der Rektifizierung

Nach Transformieren der Abbildungspunkte mit den Matrizen H_p und H'_p wurden die Epipole ins unendliche transformiert. Die Epipolarlinien verlaufen jetzt parallel zueinander, jedoch noch nicht zwingend parallel zur horizontalen Achse. In den Abbildungen 6.6 und 6.7 ist das Ergebnis der mit H_p und H'_p transformierten Bildpunkte zu sehen.

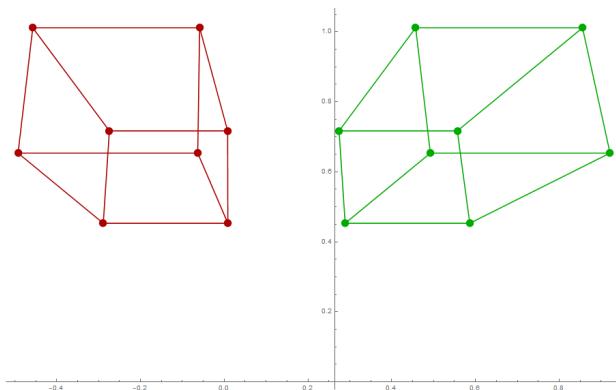


Abbildung 6.6: Abbildungen der Quader nach der Transformation mit H_p und H'_p

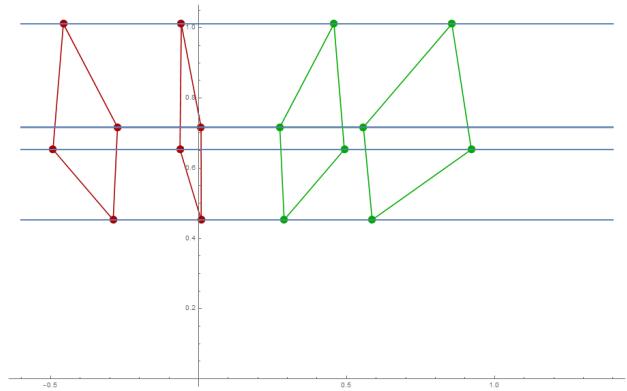


Abbildung 6.7: Abbildungen der Quader nach der Tranfromation mit H_p und H'_p mit eingezeichneten Epipolarlinien

6.2.2 Ähnlichkeitstransformation

Die Epipole der jeweiligen Bilder befinden sich nach der projektiven Transformation im unendlichen. Die daraus resultierenden Epipolarlinien sind, wie in Abbildung ?? zu sehen, parallel zueinander angeordnet. Im Folgenden sollen die Epipole rotiert werden, so dass sie ihre Richtungen $i = [1 \ 0 \ 0]$ betragen. Die Epipolarlinien würden dann parallel zur horizontalen Achse verlaufen[30]. H_r und H'_r , welche aus der Zerlegung von H_a und H'_a resultieren, sollen die Rotation ausführen. Für H_r und H'_r gelten wie aus Gleichung 6.16 bekannt folgende Gleichungen.

$$H_r = \begin{bmatrix} v_b - v_c w_b & v_a - v_c w_a & 0 \\ v_a - v_c w_a & v_b - v_c w_b & v_c \\ 0 & 0 & 1 \end{bmatrix} \quad (6.40)$$

$$H'_r = \begin{bmatrix} v'_b - v'_c w'_b & v'_a - v'_c w'_a & 0 \\ v'_a - v'_c w'_a & v'_b - v'_c w'_b & v'_c \\ 0 & 0 & 1 \end{bmatrix} \quad (6.41)$$

w und w' sind bereits bekannt. Mit Hilfe von F , können v_a und v_b ersetzt werden. Zur Erinnerung eine Voraussetzung für den hier aufgezeigten Rektifizierungsansatz ist, dass sowohl die Bildpunkte als auch die Fundamentalmatrix bekannt sein müssen. Die Fundamentalmatrix welche aus den rektifizierten Punkten entsteht hätte dann die Form

$$F = H'^T[i] \times H \quad (6.42)$$

$$F = \begin{bmatrix} u'_a & v'_a & w'_a \\ u'_b & v'_b & w'_b \\ u'_c & v'_c & 1 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} u_a & u_b & u_c \\ v_a & v_b & v_c \\ w_a & w_b & 1 \end{bmatrix} \quad (6.43)$$

$$F = \begin{bmatrix} v_a w'_a - v'_a w_a & v_b w'_a - v'_a w_b & v_c w'_a - v'_a \\ v_a w'_b - v'_b w_a & v_b w'_b - v'_b w_b & v_c w'_b - v'_b \\ v_a - v'_c w_a & v_b - v'_c w_b & v_c - v'_c \end{bmatrix} \quad (6.44)$$

$$(6.45)$$

Da bisher nur w und w' bekannt sind und die Einträge der Fundamentalmatrix F aus den nicht-rektifizierten Punkten, werden die Einträge so umgestellt. Für v_a , v_b und v_c werden jeweils die Einträge der letzten Zeile umgeformt. Für v'_a , v'_b und v'_c dient die letzte Spalte von F .

$$v_a = F_{31} + v'_c w_a \quad (6.46)$$

$$v_b = F_{32} + v'_c w_b \quad (6.47)$$

$$v_c = F_{33} + v'_c \quad (6.48)$$

$$v'_a = v_c w'_a - F_{13} \quad (6.49)$$

$$v'_b = v_c w'_b - F_{23} \quad (6.50)$$

$$v'_c = v_c - F_{33} \quad (6.51)$$

F_{31} , F_{32} , F_{33} , F_{13} und F_{23} stehen für die Einträge der Matrix F , welche aus den nicht-rektifizierten Punkten bestimmt wurde.

Werden die Gleichungen 6.46 bis 6.51 in H_r und H'_r eingesetzt, so ergeben sich für H_r und H'_r

$$H_r = \begin{bmatrix} F_{32} - w_b F_{33} & w_a F_{33} - F_{31} & 0 \\ F_{31} - w_a F_{33} & F_{32} - w_b F_{33} & F_{33} + v'_c \\ 0 & 0 & 1 \end{bmatrix} \quad (6.52)$$

$$H'_r = \begin{bmatrix} w'_b F_{33} - F_{23} & F_{13} - w'_a F_{33} & 0 \\ w'_a F_{33} - F_{13} & w'_b F_{33} - F_{23} & v'_c \\ 0 & 0 & 1 \end{bmatrix} \quad (6.53)$$

Die gemeinsame unbekannte Variable v'_c beschreibt die geometrische Verbindung beider Bilder in ihrer Verschiebung entlang ihrer vertikalen Richtung[30]. F_{33} ist somit der horizontale Versatz, welcher benötigt wird, um die horizontalen Epipolarlinien beider Bilder zueinander auszurichten, so dass die

gewünschten Scanlinien über beide Bilder entstehen. v' wird dabei so gewählt, dass die kleinste Koordinate der rektifizierten Bilder in vertikaler Achsenrichtung gleich null ist.[30].

Die Abbilder des Quader sehen nach der Transformation mit H_rH_p und $H'_rH'_p$ aus wie in den Abbildungen 6.8 und 6.9 dargestellt. Sollten die Epipolarlinien noch nicht parallel zur horizontalen Achse gewesen sein, so sind sie es nach der Transformation mit H_p und H'_p . Des Weiteren wurden beide Bilder durch v'_c so verschoben, dass der in vertikaler Richtung kleinste Punkt bei auf der Horizontalen Achse liegt.

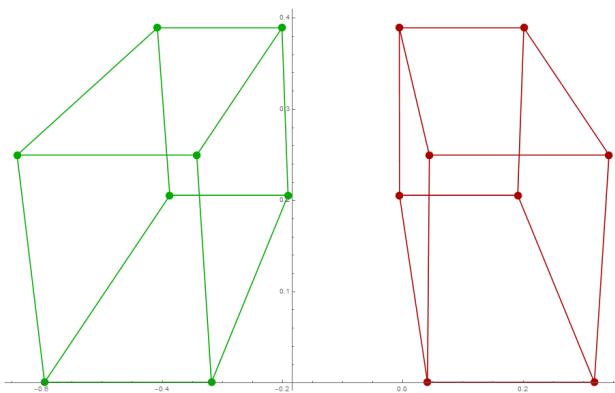


Abbildung 6.8: Abbildungen der Quader nach der Tranfromation mit H_rH_p und $H'_rH'_p$

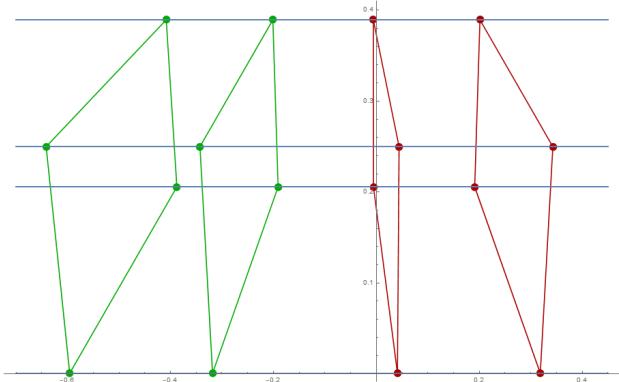


Abbildung 6.9: Abbildungen der Quader nach der Tranfromation mit H_rH_p und $H'_rH'_p$ mit eingezeichneten Epipolarlinien

6.2.3 Scherungstransformation

Im letzten Schritt soll die durch die Transformation mit H_p und H'_p entstandene horizontale Verzerrung in beiden Bildern reduziert werden. Aus diesem Grund werden die Scherungsmatrizen H_s und H'_s , aus der Zerlegung der affinen Matrix H_a und H'_a benötigt. Die Einträge der ersten Zeile in H_s und H'_s haben nur noch Auswirkungen auf die horizontalen Koordinaten der Bildpunkte.

$$H_s = \begin{bmatrix} u_a & u_b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.54)$$

$$H'_s = \begin{bmatrix} u'_a & u'_b & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.55)$$

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.

Da es nicht möglich ist die horizontale Verzerrung gänzlich zu reduzieren, wird stattdessen versucht die Orthogonalität und das Seitenverhältnis der Verbindungslien \overline{bd} und \overline{ca} wieder herzustellen. In Abbildung 6.10 ist das Ziel nochmal grafisch dargestellt.

Für den nächsten Schritt werden im synthetischen Beispiel für die Abbildungen des Quaders jeweils Bildweite w und Bildhöhe h definiert. Da zunächst von gleichen Kameraauflösungen ausgegangen wird, sind die Bildbreiten und Höhen beider Kameras gleich. Danach werden die vier Mittelpunkte a, b, c und d der Bildkanten definiert mit $a = [\frac{w-1}{2} \ 0 \ 1]^T, b = [w - 1 \ \frac{h-1}{2} \ 1]^T, c = [\frac{w-1}{2} \ h - 1 \ 1]^T, d = [0 \ \frac{h-1}{2} \ 1]^T$.

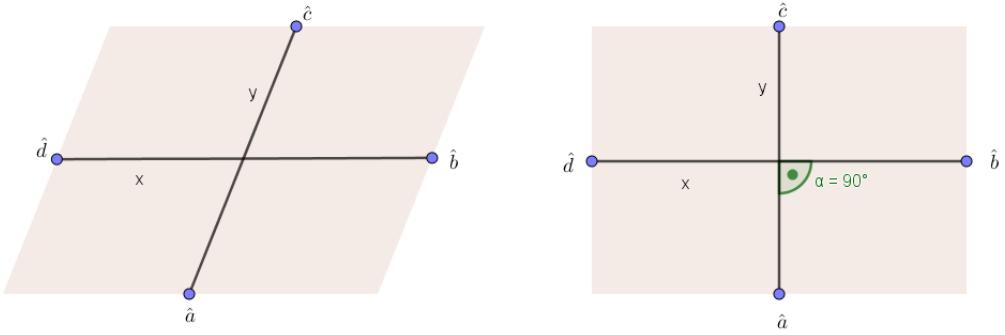


Abbildung 6.10: Die Verbindungslien \overline{bd} und \overline{ca} sollen so ausgerichtet werden, dass sie orthogonal zueinander stehen.

Die Punkte a, b, c, d und auch a', b', c', d' sind die Mittelpunkte der noch nicht rektifizierten Bildkanten. Diese werden mit den Transformationsmatrizen $H_s H_r H_p$ und $H'_s H'_r H'_p$ verrechnet, um so die Position der Kantenmitten nach der Rektifizierung zu erhalten.

$$\begin{aligned}\hat{a} &= H_r \cdot H_p \cdot a \\ \hat{b} &= H_r \cdot H_p \cdot b \\ \hat{c} &= H_r \cdot H_p \cdot c \\ \hat{d} &= H_r \cdot H_p \cdot d \\ \hat{a}' &= H'_r \cdot H'_p \cdot a' \\ \hat{b}' &= H'_r \cdot H'_p \cdot b' \\ \hat{c}' &= H'_r \cdot H'_p \cdot c' \\ \hat{d}' &= H'_r \cdot H'_p \cdot d'\end{aligned}$$

Danach können die Vektoren \vec{x} und \vec{y} aus den Differenzen der sich ursprünglich gegenüberliegenden Punkte gebildet werden.

$$x = \hat{b} - \hat{d} \quad (6.56)$$

$$y = \hat{c} - \hat{a} \quad (6.57)$$

$$x' = \hat{b}' - \hat{d}' \quad (6.58)$$

$$y' = \hat{c}' - \hat{a}' \quad (6.59)$$

x und y sind Vektoren der euklidischen Bildebene[30]. Das heißt, sie sind genau dann orthogonal zueinander wenn gilt:

$$(H_s x)^T (H_s y) = 0 \quad (6.60)$$

$$(H'_s x')^T (H'_s y') = 0 \quad (6.61)$$

Für die Erhaltung der Seitenverhältnisse gilt dann:

$$\frac{(H_s x)^T (H_s x)}{(H_s y)^T (H_s y)} = \frac{w^2}{h^2} \quad (6.62)$$

$$\frac{(H'_s x')^T (H'_s x')}{(H'_s y')^T (H'_s y')} = \frac{w'^2}{h'^2} \quad (6.63)$$

Anhand der in den Gleichungen 6.60, 6.61, 6.62 und 6.63 können die folgenden Gleichungen für die Matirixeinträge für H_s und H'_s aufgestellt werden[30, 37].

$$u_a = \frac{h^2 x_v^2 + w^2 + y_v^2}{hw(x_v y_u - x_u y_v)} \quad (6.64)$$

$$u_b = \frac{h^2 x_u x_v + w^2 y_u y_v}{hw(x_u y_v - x_v y_u)} \quad (6.65)$$

Die selben Gleichungen werden auch für u'_a und u'_b aufgestellt. Das Ergebnis der gesammten Transformation H mit $H_s H_r H_p$ und H' mit $H'_s H'_r H'_p$ ist in den Abbildungen 6.11 und 6.12 zu sehen.

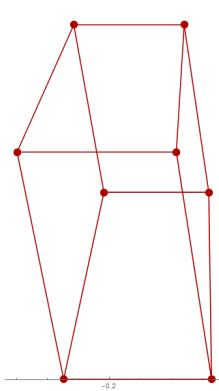


Abbildung 6.11: Abbildungen der Quader nach der Tranfromation mit $H_s H_r H_p$ und $H'_s H'_r H'_p$

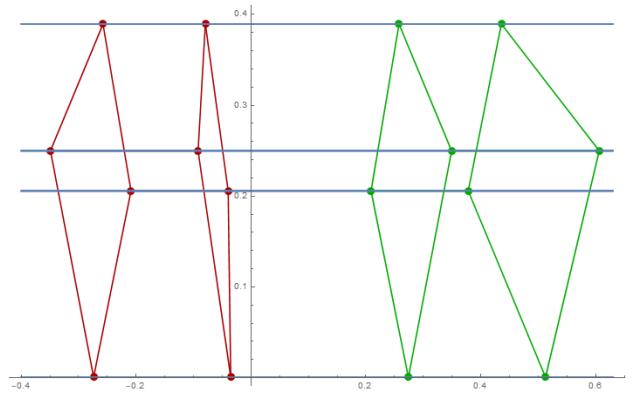
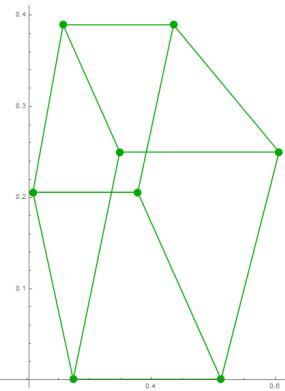


Abbildung 6.12: Abbildungen der Quader nach der Tranfromation mit $H_s H_r H_p$ und $H'_s H'_r H'_p$ mit eingezeichneten Epipolarlinien

6.3 Rektifizierung mit unterschiedlichen Kameraauflösungen

Im Folgenden wird der entstandenen Rektifizierungsalgorithmus auf Bilder unterschiedlicher Auflösung angewandt. Im ersten Beispiel wird für C die Auflösung $\zeta_x = \zeta_y = 1$ gewählt. Die Kameramatrix K lautet wie folgt

$$K = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.66)$$

Für C' wird eine kleinere Auflösung mit $\zeta'_x = \zeta'_y = 0.5$ definiert. Die Kameramatrix K' lautet somit

$$\begin{bmatrix} 0.5 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.67)$$

Die entstehenden Bilder des Quaders sind in Abbildung 6.13 zu sehen. Da K' nur eine halb so große Auflösung wie K besitzt, ist das resultierende Bild des roten Quaders auch nur halb so groß. Der grüne Quader zeigt das entstehende Bild von C' .

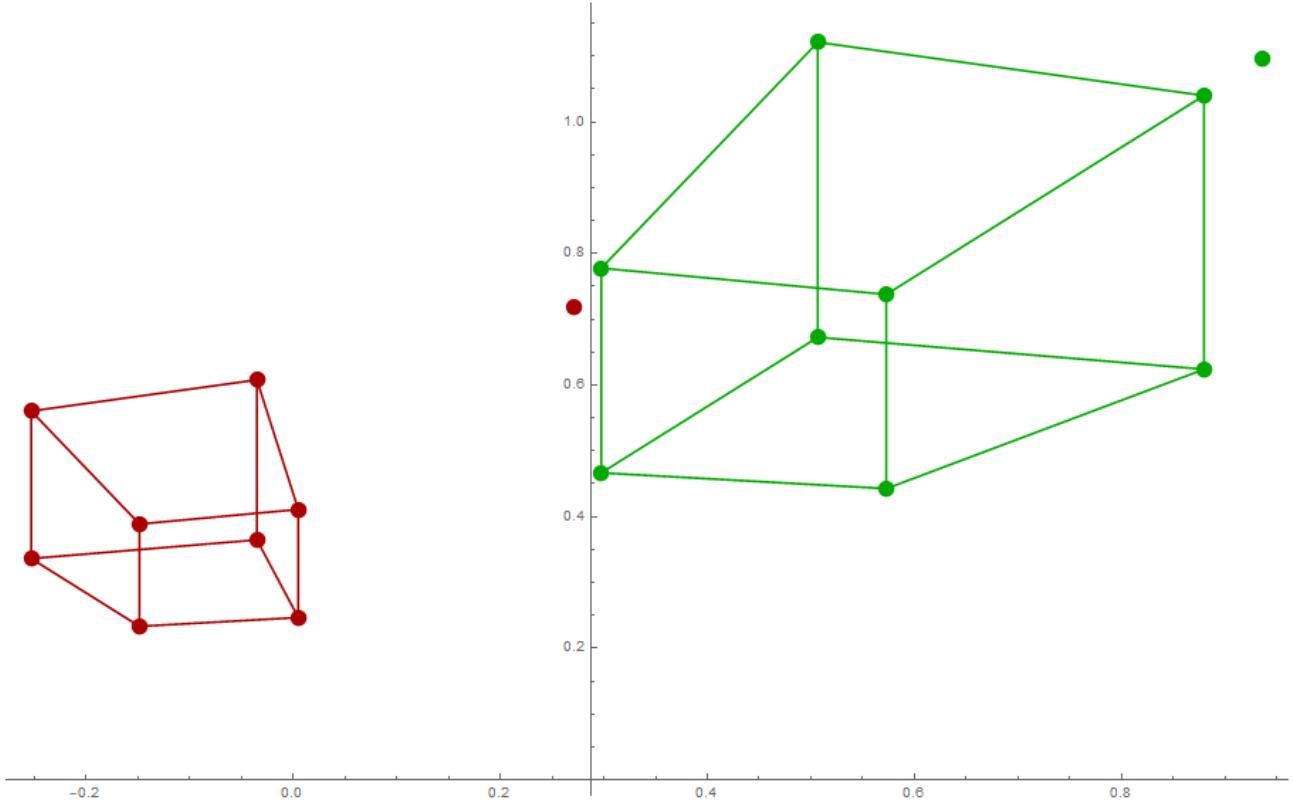


Abbildung 6.13: Aufnahmen zweier Kameras mit unterschiedlichen Auflösungen. Für Kamera eins(Grün) gilt $\zeta_x = \zeta_y = 1$ und für Kamera zwei(rot) gilt $\zeta'_x = \zeta'_y = 2$.

Abbildung 6.14 zeigt die projektive Transformation der beiden Bilder mit H_p und H'_p . Die Epipolarlinien der beiden Bilder sind nach dieser Transformation jeweils parallel zueinander. In Abbildung 6.15 ist das Resultat zu sehen wenn die Transformation H_r und H'_r dazukommen. Die Epipolarlinien sind jetzt parallel zur horizontalen Achse und die Epipolarlinien von Bild eins und Bild zwei sind so zueinander ausgerichtet, dass sie zu einheitlichen Linien über zwei Bilder werden. Der rote Quader, welches das Bild mit der niedrigeren Auflösung repräsentiert, ist nach dieser Transformation stark horizontaler und vertikaler Richtung vergrößert wird.

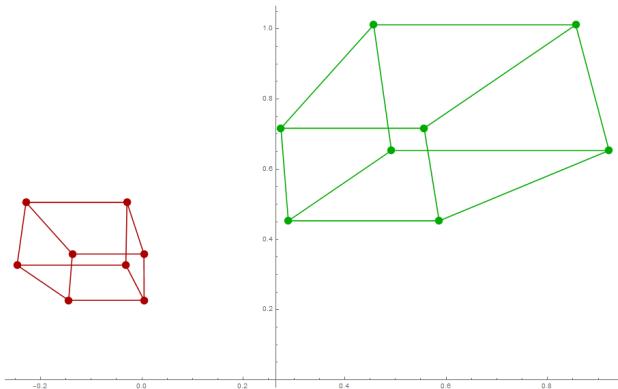


Abbildung 6.14

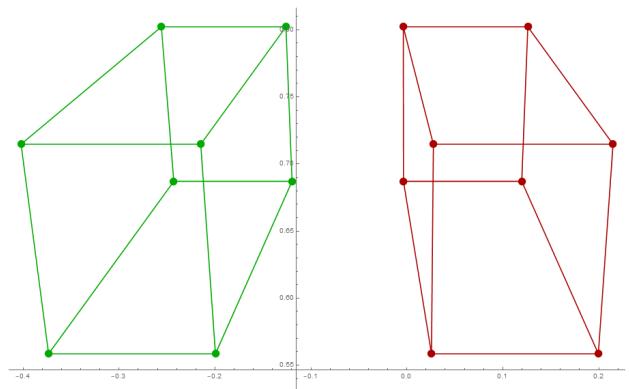


Abbildung 6.15

Die Abbildungen 6.16 und 6.17 zeigen die letzt Transformation mit H_s und H'_s einmal mit und einmal ohne Epipolarlinien. Die Bilder scheinen richtig rektifiziert geworden zu sein. Dieser Test wurde noch mit weiteren Vielfachen der Kameramatrix K ausprobiert, immer mit dem selben Ergebnis wie in den Abbildungen 6.16 und 6.17 zu sehen. Somit ist es wohl prinzipiell möglich aus den Rektifizieriten Bilder eine Tiefenkarte zu erstellen und somit die 3D-Szene zu rekonstruieren.

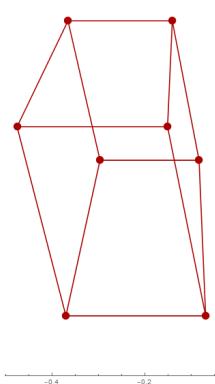


Abbildung 6.16

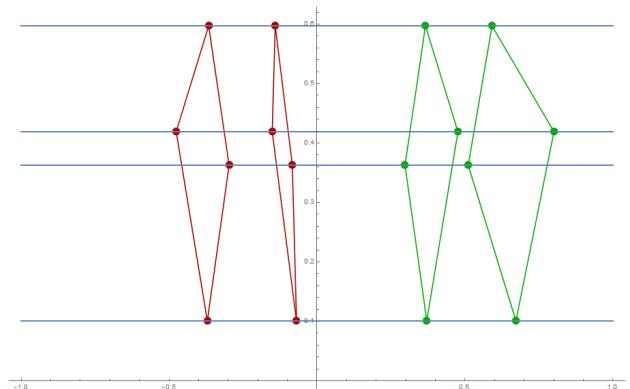
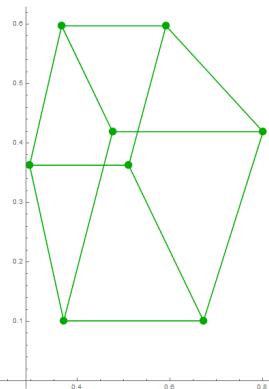


Abbildung 6.17

Als nächstes wurden die Auflösung von C' so verändert, dass $\zeta'_x \neq \zeta'_y$ gilt. Es werden $\zeta'_x = 2.3$ und $\zeta'_y = 3.2$ definiert. Somit folgt für K' die folgende Matrix.

$$K' = \begin{bmatrix} 2.3 & 0 & 0 \\ 0 & 3.2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.68)$$

Für K von C gilt weiterhin $\zeta_x = \zeta_y = 1$. Die Entstehenden Bilder sind in Abbildung 6.18 zu sehen. Die horizontale Kantenlänge roten Quaders ist im Verhältnis kürzer als ihre vertikale Kantenlänge. Wie in Abbildung 6.19 zu sehen ist, bleibt ungleiche Seitenverhältnis des roten Quaders nach der Rektifizierung erhalten, weshalb der Rote Quader schmäler ist als der der grüne.

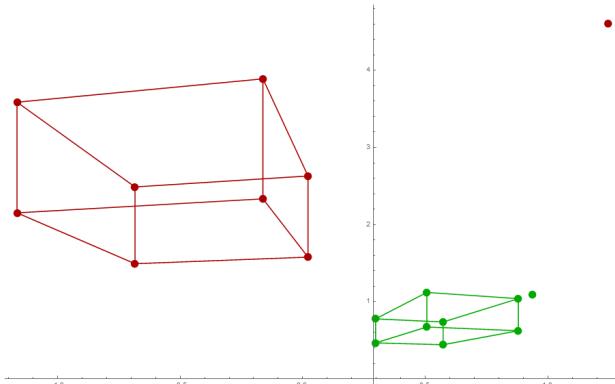


Abbildung 6.18: Aufnahmen zweier Kameras mit unterschiedlichen Auflösungen. Für Kamera eins(grün) gilt $\zeta_x = \zeta_y = 1$ und für Kamera zwei(rot) gilt $\zeta'_x = 2.3$ $\zeta'_y = 3.2$

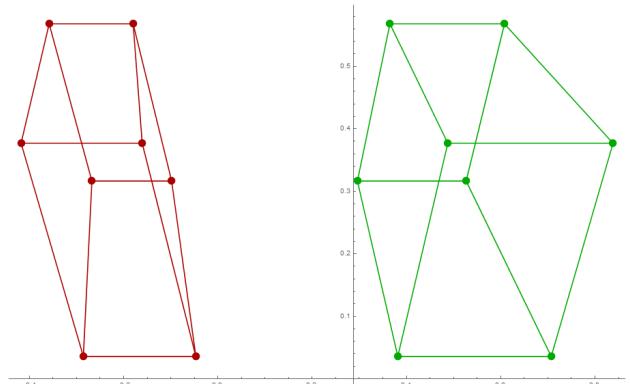


Abbildung 6.19: Nach der Rektifizierung stimmen die horizontalen Koordinaten nicht überein, es würde zu Fehlern in der Tiefenkarte und somit in der gesamten 3D-Rekonstruktion geben.

Die Beispiele zeigen, dass es möglich ist Bilder unterschiedlicher Beispiele zu rektifizieren, jedoch müssen für die nachfolgende Erstellung der Tiefenmaps....(was wäre jetzt ein guter Abschluss??)

7 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 und Spalten des Schachbretts sie sich befinden. Für die Sortierung der Punkte in ein Schachbrettmuster wurde ein Algorithmus entwickelt welcher nach folgendem Schema funktioniert.

Als erstes wird initial ein Startwert gesetzt
(Hier Grafik der Funktionsübersicht)

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.

7.1 Sortierungsalgorithmus

Für die Sortierung der Schachbrettpunkte, wird das Schachbrett in ein Koordinatensystem, mit vertikaler Achse i und horizontaler Achse j , gelegt. Um das Schachbrett herum wird ein Rahmen definiert. Die Punkte mit der maximale i -Koordinate und der minimalen i -Koordinate Begrenzen die oberen und unteren Kanten des Rahmens. Die Punkte mit der minimalen j -Koordinate und der Punkt mit der maximalen j -Koordinate Begrenzen die vertikalen Kanten des Rahmens. In Abbildung 7.1 sind die roten Begrenzungskanten um das Schachbrett zu sehen.

Der durch den Rahmen begrenzte Bereich wird in gleich viele Reihen wie Spalten unterteilt, so dass eine Gitter entsteht. Die so entstehenden Zellen des Gitters werden in i -Richtung sowie in j -Richtung durchgezählt und bekommen somit jeweils zwei Indizes (i_i, j_i) , welche sie eindeutig identifizieren.

In den angelegten *ConstantArrays* namens *JSplits* und *ISplits* werden die Begrenzungen Zellen in i - und j -Richtung gespeichert. Die Begrenzungen der Zellen werden über die Distanz des Punktes mit der minimalen j -Koordinate zum Punkt mit der maximalen j -Koordinate geteilt durch die gewünschte Anzahl der Zellen, definiert. Selbiges gilt für die Unterteilung der Zellen in i -Richtung.

Alle Punkte in den Zellen mit den Indizes $i = 1$ und $j \leq j_{max}$ werden als mögliche Startpunkte gekennzeichnet. In Abbildung 7.1 befinden sich die möglichen Startpunkte innerhalb des blauen Bereichs. Die Suche nach einem Startpunkt und auch nach den ersten Punkten in zwei Richtungen vom Startpunkt aus, ist relativ komplex. Der Grund hierfür ist, dass sämtliche Schachbretter wie in 7.2 gezeigt, in Betracht gezogen werden müssen. Um Anhand des Algorithmus später korrespondierende Punkte in zwei Aufnahmen des Schachbretts finden kann, sollte gewährleistet sein, dass der Startpunkt immer an der selben Ecke eines Schachbretts platziert wird.

Innerhalb der blau gefärbten Fläche wird nach einem Startpunkt gesucht. Die Suche nach einem Startpunkt wird pro i - und j -Richtung in zwei Abfragen unterteilt.

In der ersten Abfrage für die i -Richtung, wird innerhalb der ersten Zellenreihe entlang der j -Koordinatenachse nach dem Punkt mit der kleinsten i -Koordinate gesucht. Dieser Punkt wird als möglicher erster Startpunkt in der Variablen $VecI$ gespeichert.

Danach wird geprüft ob es innerhalb der ersten Zellen in i -Richtung einen weiteren Punkt gibt dessen j -Koordinate kleiner ist als die des momentan gesetzten Punktes $VecI$, wird ein solcher Punkt gefunden, so wird dieser Punkt als neuer $VecI$ gesetzt, sofern seine i -Koordinate nicht größer ist als die des zuvorigen $VecI$ plus einem Pufferwert. Durch den Puffer wird verhindert, dass ein Punkt als $VecI$ gesetzt wird, welcher sich möglicherweise schon in der zweiten Reihe der Schachbrettpunkte befinden würde. In Abbildung 7.1 ist $VecI$ mit als orangener Punkt abgebildet.

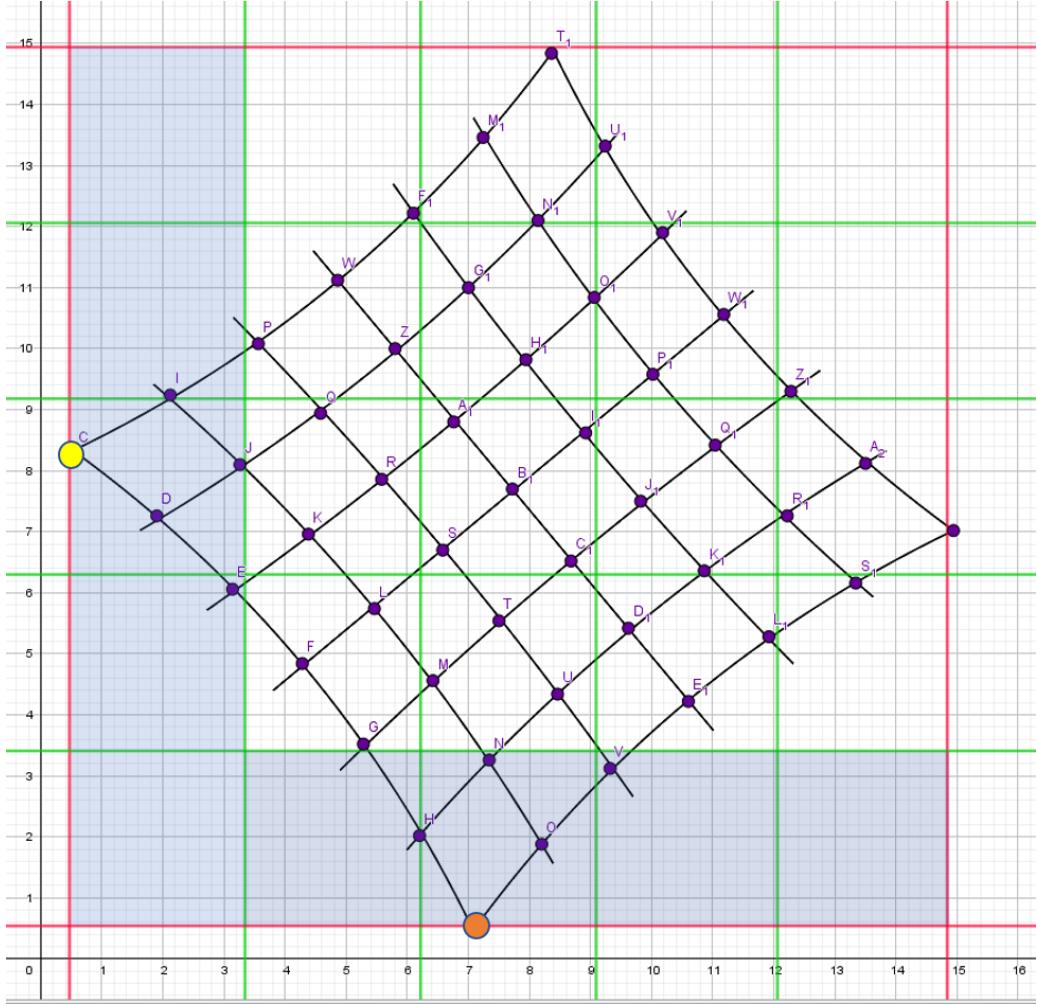


Abbildung 7.1: Die in blau markierten Bereiche beinhalten die möglichen Startpunkte. Der Bereich entlang der horizontalen j -Achse bildet die erste Suchfensterreihe in i -Richtung. Der blaue Bereich entlang der vertikalen i -Achse bildet die erste Suchfensterreihe in j -Richtung. Der gelbe Punkt steht für den Punkt welcher als $VecJ$ bezeichnet wurde und der orangefarbene Punkt ist derjenige Punkt, welcher als $VecI$ bestimmt wurde

Für die erste Abfrage in j -Richtung, werden die Zellen entlang der vertikalen i -Koordinatenachse mit den Indizes $j = 1$ und $i \leq i_{max}$, nach dem Punkt gesucht, welcher die geringste j -Koordinate besitzt. Dieser Punkt wird als vorläufiges $VecJ$ bestimmt.

Danach wird auch hier geprüft, ob es innerhalb der ersten Suchfensterreihe in j -Richtung einen weiteren Punkt gibt, dessen i -Koordinate kleiner ist als die des momentan gesetzten Punktes $VecJ$. Trifft das Weiteren zu, dass die j -Koordinaten des neu gefundenen Punktes kleiner ist als die j -Koordinate

plus einem Pufferwert des momentanen $VecJ$, so wird dieser als neuer Punkt als $VecJ$ festgelegt. In Abbildung 7.1 ist $VecJ$ als gelber Punkt zu sehen.

Je nachdem wie das Schachbrett liegt oder welche Verzerrungen es aufweist, kann es sein dass $VecI$ und $VecJ$ den selben Punkt ergeben haben, was den Startpunkt $StartPoint$ eindeutig identifiziert, wie es in Beispiel ?? oder auch ?? der Fall ist. Andererseits kann es auch wie in Abbildung 7.1 zu sehen ist, sein, dass $VecI$ und $VecJ$ sich unterscheiden. In solchen Fällen wird $VecJ$ als Startpunkt festgelegt wird. Somit ist gewährleistet, das je nach Lage des Schachbretts der linke obere oder die linke untere Ecke des Schachbretts als Startpunkt gesetzt werden

Die Koordinaten des $Startpoint$ werden in eine Liste aus *Associations* [22] hinzugefügt. *Associations* können Werten Schlüsselwörter zuweisen. Anhand dieser Schlüssel sind Werte eindeutig identifiziert. Die *Association*-Liste *SortedPoints* beinhaltet die Schlüssel *CoordJ* und *CoordI*, welche die jeweiligen i - und j - Koordinaten des Punktes speichern. Die Schlüssel *CellI* und *CellJ*, speichern die Information in welcher Zelle sich der Startpunkt befindet und die Schlüssel *NeighbourI* und *NeighbourJ* speichern die Indizes i und j ab, die definieren, der wievielte Punkt *StartPoint* in i -Richtung und in j -Richtung ist. Er bekommt als erster Sortierter Punkte die $NeighbourI = 1$ und $NeighbourJ = 1$ zugeordnet. Ein Eintrag für einen Punkt in der Liste *SortedPoints* sieht wie folgt aus

$$StartPoint = \{ < |CoordI \rightarrow i - \text{Koordinate}, CoordJ \rightarrow j - \text{Koordinat}, CellI \rightarrow i - \text{Zelle}, CellJ \rightarrow j - \text{Zelle}, NeighbourI \rightarrow 1, NeighbourJ \rightarrow 1| > \}$$

Anhand der Schlüssel *NeighbourI* und *NeighbourJ*, können Punkte eindeutig identifiziert werden, da die in den Schlüssel die Information steckt, in welcher Reihe von i und in welcher Spalte von j sich ein Punkt aufhält und welche Punkte sind mit ihm in der selben Reihe oder Spalte. In den Abbildungen ?? bis ?? wurden die Punkte angefragt welche sich in der dritten Reihe von i befinden. Besagte Punkte sind in den Grafiken grün eingefärbt.

Die Schlüssel *ICoord* und *JCoord* werden noch in eine weitere Liste namens *CheckList* gespeichert. Anhand der *CheckList* wird überprüft, ob die Koordinaten und somit ein Punkt selbst bereits Sortiert wurde. Damit wird verhindert, dass ein Punkt mehrmals in eingesortiert wird.

Nachdem der Startpunkt gefunden ist werden die nächsten Punkte in i - und j -Richtung gesucht. Diese Punkte werden später in die Variablen *NextI* und *NextJ* gespeichert. In Abbildung 7.2 ist *NextI* der violette Punkte und *NextJ* ist der rote Punkte.

Für die Bestimmung von *NextI* wird zu erst initial $NextI = < |CoordI \rightarrow 100000, CoordJ \rightarrow 100000| >$ gesetzt. Es werden Punkte gesucht, welche sich in der selben *CellI* wie *StartPoint* und in den Zellen eins darüber *CellI* + 1 und eine darunter *CellI* - 1 befinden. In Abbildung 7.2 ist der Bereich gelb eingefärbt.

Gibt es in diesem Bereich einen Punkt, dessen i -Koordinate größer ist als die i -Koordinate des *StartPoints* und dessen j -Koordinate kleiner ist als die des momentan gesetzten *NextI*, so wird dieser Punkt als neues *NextI* festgelegt. Danach wird geprüft, ob der neue *NextI* schon der letztendliche *NextI* ist. Um das zu prüfen wird zunächst überprüft ob es innerhalb des gelben Bereichs noch einen Punkt gibt, dessen j -Koordinatenabstand vom Startpunkt aus kleiner gleich dem j -Koordinatenabstand zwischen dem momentanen *NextI* zum Startpunkt ist. Gleichzeitig wird auch noch geprüft, ob der i -Koordinatenabstand zwischen *StartPoint* und *NextI* größer ist als der i -Koordinatenabstand zwischen einem Punkt innerhalb des Bereichs und *StartPoint*. Ist dies der Fall, so wird der gefundenen Punkt als neues *NextI* bestimmt. Für die Überprüfung mit *NextJ* wird nach dem gleichen Schema verfahren. in Abbildung ?? ist die eben beschriebene Prüfung nochmal grafisch dargestellt. *NextI* und *NextJ* werden dann mit den selben Schlüssel aber anderen zugeteilten Werten

wie *StartPoint* in die Liste *SortedPoints* geschrieben. Die *CheckList* wird um die Punkte *NextI* und *NextJ* erweitert.

$$\begin{aligned} \text{NextI} &= \{<|\text{CoordI} \rightarrow i - \text{Koordinate}, \text{CoordJ} \rightarrow j - \text{Koordinat}, \\ &\quad \text{CellI} \rightarrow i - \text{Zelle}, \text{CellJ} \rightarrow j - \text{Zelle}, \text{NeighbourI} \rightarrow 2, \text{NeighbourJ} \rightarrow 1|>\} \end{aligned}$$

$$\begin{aligned} \text{NextJ} &= \{<|\text{CoordI} \rightarrow i - \text{Koordinate}, \text{CoordJ} \rightarrow j - \text{Koordinat}, \\ &\quad \text{CellI} \rightarrow i - \text{Zelle}, \text{CellJ} \rightarrow j - \text{Zelle}, \text{NeighbourI} \rightarrow 1, \text{NeighbourJ} \rightarrow 2|>\} \end{aligned}$$

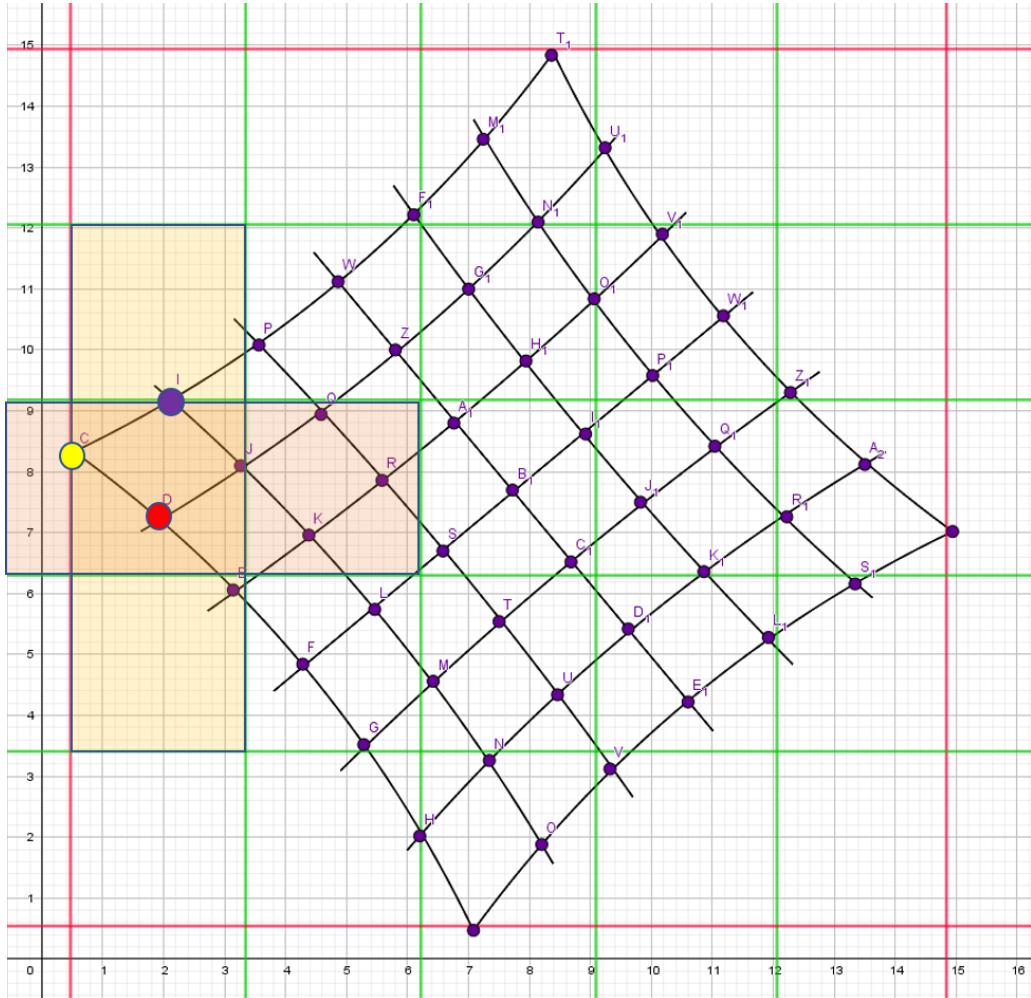


Abbildung 7.2: Die in blau markierten Bereiche beinhalten die möglichen Startpunkte. Der Bereich entlang der horizontalen j -Achse bildet die erste Suchfensterreihe in i -Richtung. Der blaue Bereich entlang der vertikalen i -Achse bildet die erste Suchfensterreihe in j -Richtung. Der gelbe Punkt steht für den Punkt welcher als VecJ bezeichnet wurde und der orange Punkt ist derjenige Punkt, welcher als VecI bestimmt wurde

(HIER GRAFIK EINFÜGEN SIEHE NOTIZ)

Mit den Punkten *StartPoint*, *NextI* und *NextJ* können die ersten Richtungsvektoren gebildet werden, anhand welcher das Schachbrett Gitter rekonstruiert werden kann. Für beide Richtungen wird die Anzahl der möglichen Punkte begrenzt. Es werden zwei Listen *IList* und *JList* angelegt. In *IList* werden alle Punkte innerhalb der Zellen CellI_{min} bis CellI_{max} und CellJ_1 und CellJ_2 gespeichert. In Abbildung 7.3 entspricht das dem Vertikalen blauen Bereich. In *JList*, kommen alle Punkte innerhalb von CellJ_{min} bis CellJ_{max} und CellI_1 und CellI_2 . Was dem horizontalen blauen Bereich in Abbildung 7.3 entspricht.

Zuerst wird die erste Kante in j -Richtung vervollständigt. Es wird ein Richtungsvektor $NextJDir$ aus $StartPoint$ und $NextJ$ gebildet. Des Weiteren wird ein Pufferwert $ProportionI$ deklariert, welcher den i -Koordinatenabstand zwischen $StartPoint$ und $NextJ$ beinhaltet. Anhand des Richtungsvektors $NextJDir$ und des Pufferwertes $ProportionI$ wird der nächste Punkt von $NextJ$ aus gesucht.

$$NextJDir = NextI - StartPoint$$

$$ProportionI = NextI[CoordI] - StartPoint[CoordI]$$

In Abbildung 7.3 auf dem linken Bild wird der Richtungsvektor $NextJDir$ als Pfeil vom gelben $StartPoint$ zum roten $NextJ$ dargestellt, der Richtungsvektor wird noch um einen Pufferwert vergrößert, da durch Perspektive und Bildverzerrungen die Abstände zwischen den einzelnen Punkten nicht mehr einheitlich sein muss. Die Pfeile welche in i -Richtung von $NextJ$ aus ausgehen, ist der Abstand $ProportionI$, welcher auf die i -Koordinate von $NextJ$ einmal addiert und einmal subtrahiert wird. Der Bereich den die Vektoren aufspannen bildet den Suchbereich für den auf $NextJ$ folgenden Punkt.

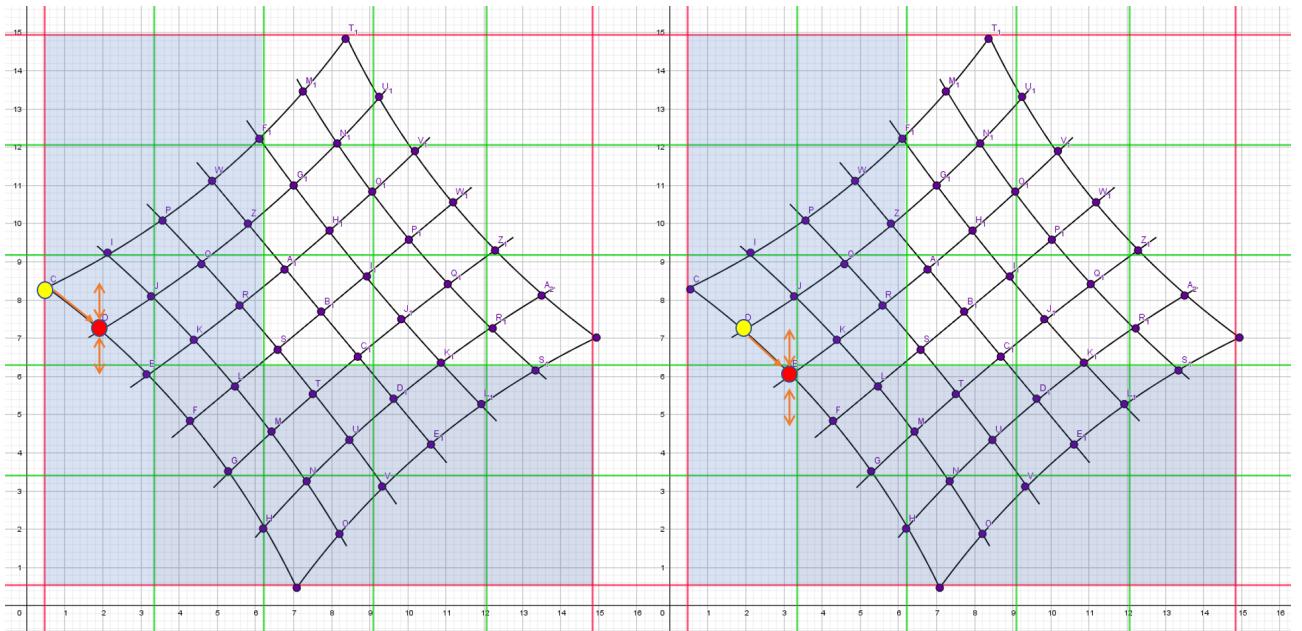


Abbildung 7.3: Klassendiagramm

$NextJ$ wird zum neuen $StartPoint$. Vom neuen $StartPoint$ ausgehen, wird der zuvor definierte Suchbereich nach einem neuem $NextJ$ abgesucht. Wird ein Punkt in diesem Bereich entdeckt, so wird dieser zum neuen $NextJ$, sofern er sich noch nicht in der $CheckList$ befindet. Tritt der Fall ein, dass ein Punkt bereits in der $CheckList$ sich befindet, so wird dieser ignoriert. Die Position und der Platz im Schachbrettgitter werden dem neu entdeckten Punkt, wie die Punkten zuvor, durch Schlüsselwerte festgehalten und in die Liste $SortedList$ gespeichert. Des Weiteren wird auch dieser in die $CheckList$ aufgenommen.

Anhand des neuen $StartPoints$ und des neuen $NextJ$ werden der Richtungsvektor $NextJDir$ und $ProportionI$ neu berechnet. $NextJ$ wird wieder zum neuen $StartPoint$ und die Suche wird dementsprechend weiter geführt, bis kein Punkt mehr gefunden wird. Alle Punkte werden als $Asssociation$ mit entsprechenden Schlüsselwerten in $SortedList$ und $CheckList$ eingetragen.

Bevor die Suche jedoch beendet wird, treten noch zwei Sicherheitsfunktionen in Kraft. Die erste wird als *Safetylist* bezeichnet. Wird beispielsweise kein weiterer Punkt innerhalb von $JList$ gefunden, so erweitert *Safetylist* den Suchbereich um den letzten noch detektierten $NextJ$ um alle Zellen um diesen Punkt herum und überprüft, dann nochmal, ob es einen weiteren Punkt innerhalb des letzten

definierten Suchbereiches gibt. Ist dies der Fall, so wird dieser Punkt auch noch mit aufgenommen. Ist dies nicht der Fall ist die Suche für die Reihe beendet. Die selbe Sicherheitsfunktion gibt es auch für die Suche in i -Richtung.

In Abbildung 7.5 auf dem rechten Bild ist ein Beispiel für den Einsatz der *SaftyList*-Funktion zu sehen. Der rote Bereich vom gelben Punkt aus gehend ist definiert als der Bereich potentieller weiterer Punkte außerhalb der *IList*.

Ist die erste j -Kante bestimmt, so wird die nächste Reihe vervollständigt, beginnend am Punkt *NextI*. Danach muss in i -Richtung zunächst der nächste *NextI* bestimmt werden, damit aufbauend auf diesem die nächste Reihe in j -Richtung vervollständigt werden kann. Wie in den Abbildung 7.4 und 7.5 zu sehen, basiert die Suche nach *NextI* auf dem gleichen Verfahren wie die Suche nach *NextJ*.

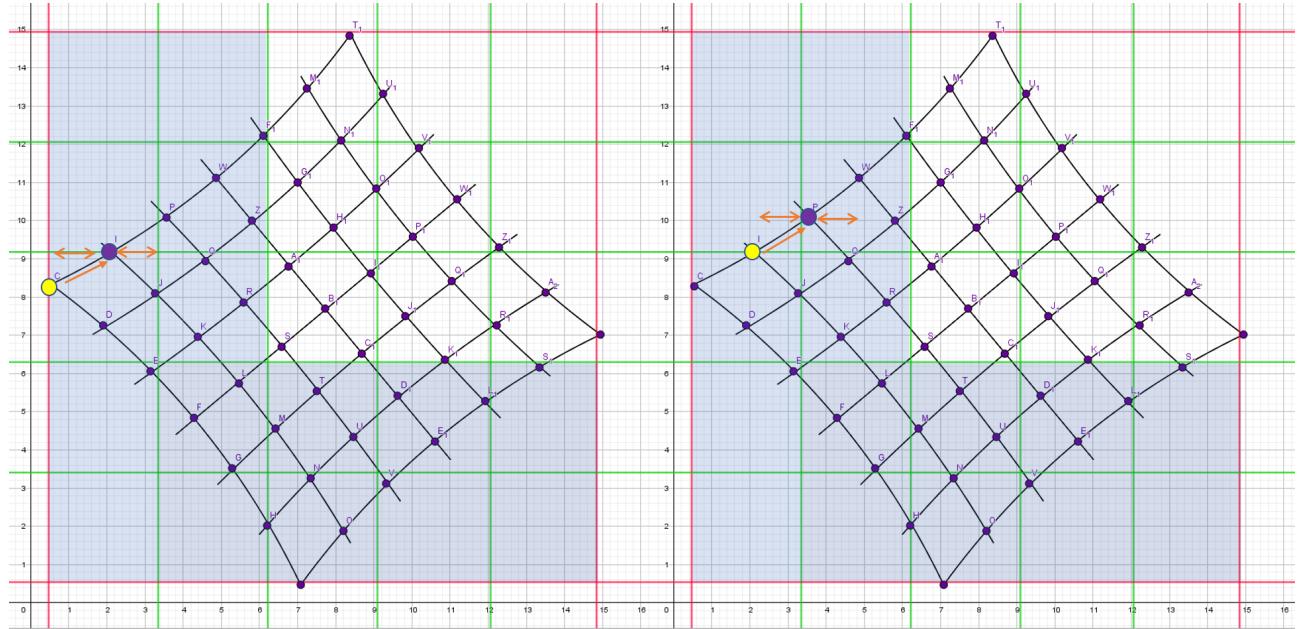


Abbildung 7.4: Klassendiagramm

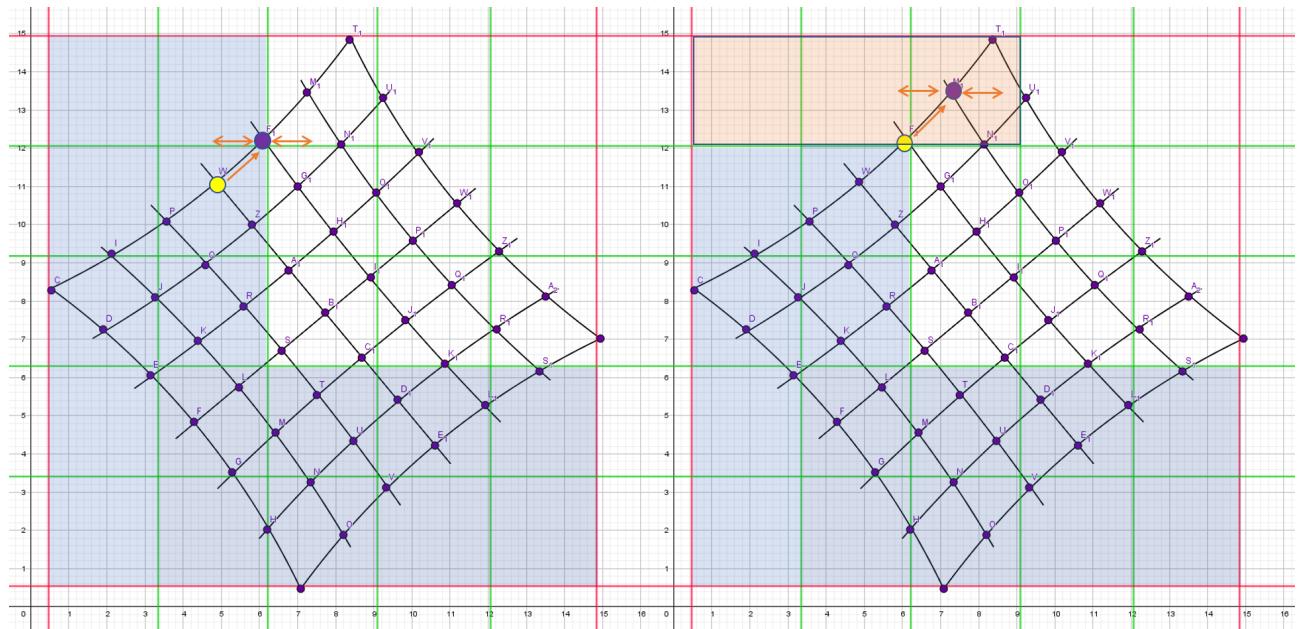


Abbildung 7.5: Klassendiagramm

Zuvor wurde erwähnt, dass es neben *SaftyList* noch eine andere Funktion, welche die Detektion aller Punkte absichert. Hierzu sei nochmal erwähnt, dass der hier beschriebene Sortierungsalgorithmus auf

einem bereits bestehenden Algorithmus aufbaut. Dieser Algorithmus detektiert die Eckpunkte eines Schachbretts. Bei der Detektion kann es vorkommen, dass Punkte nicht erkannt werden. Ist dies der Fall entstehen Lücken im Schachbrett-Muster und der Sortierungsalgorithmus, würde nicht alle Punkte richtig sortieren können.

Um das zu verhindern, wurde eine Sicherheitsfunktion eingebaut. Sollte es vorerst kein Punkt im Suchbereich entdeckt werden, so setzt die Sicherheitsfunktion einen synthetischen Punkt und sucht ausgehend von diesem synthetischen weiter nach Punkten. Sollte ein Punkt nach dem synthetischen Punkt gefunden werden, so bleibt der synthetische Punkt bestehen und die Suche wird normal fortgesetzt. Wird nach dem setzen des synthetischen Punktes kein weiterer Punkt gefunden, gilt die Suche in der Reihe beziehungsweise Spalte für beendet.

(GRAFIK EINBAUEN FÜR SYNTETISCHEN PUNKT)

7.2 Extrembeispiele

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.

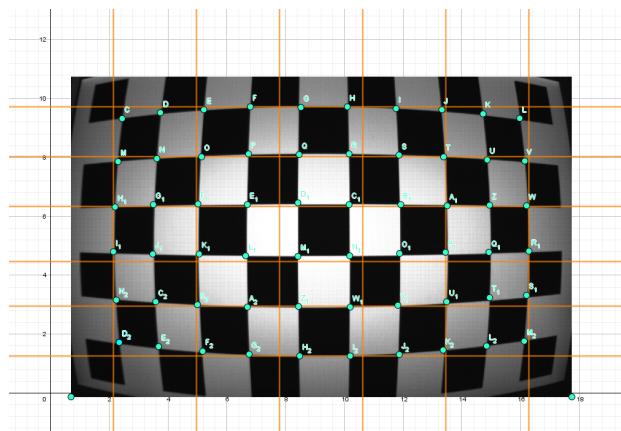


Abbildung 7.6: Bild eines tonnenförmig verzeichneten Schachbretts

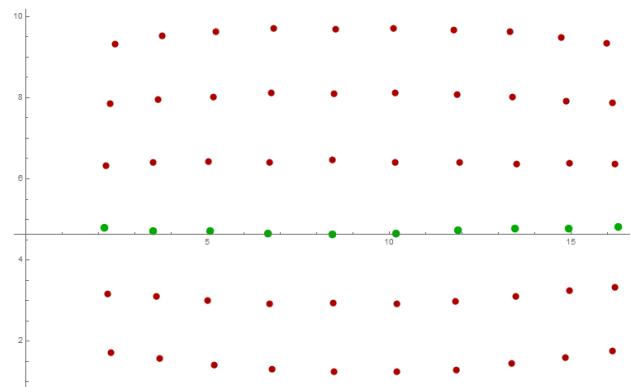


Abbildung 7.7: Algorithmisch detektierte Linie der dritten i-Reihe

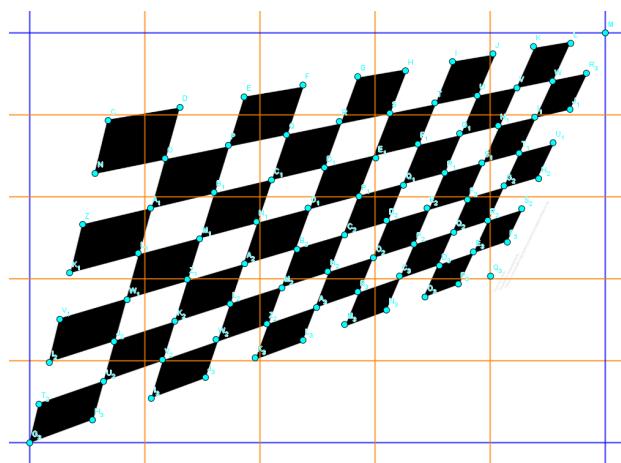


Abbildung 7.8: Bild eines perspektivisch verzerrten Schachbretts

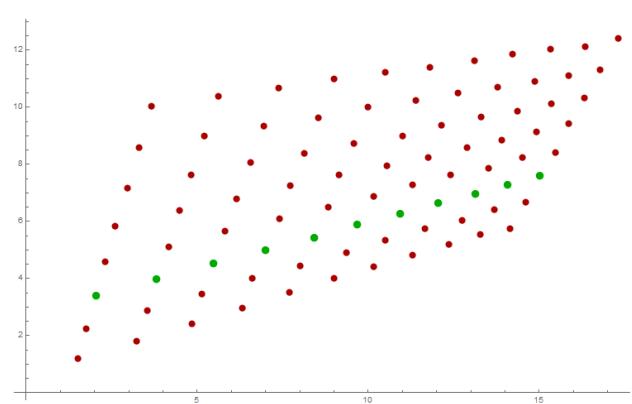


Abbildung 7.9: Algorithmisch detektierte Linie der dritten i-Reihe

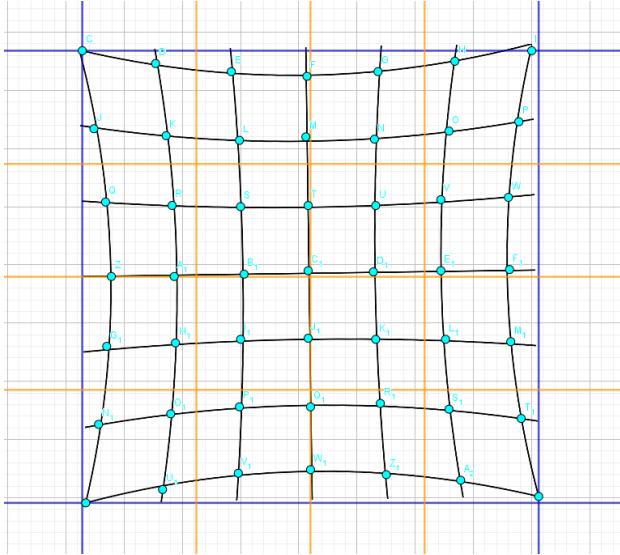


Abbildung 7.10: Bild eines Kissenförmig verzeichnetem Schachbretts

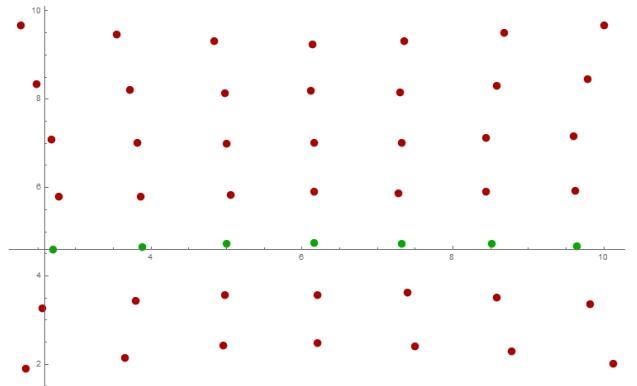


Abbildung 7.11: Algorithmisch detektierte Linie der dritten i-Reihe

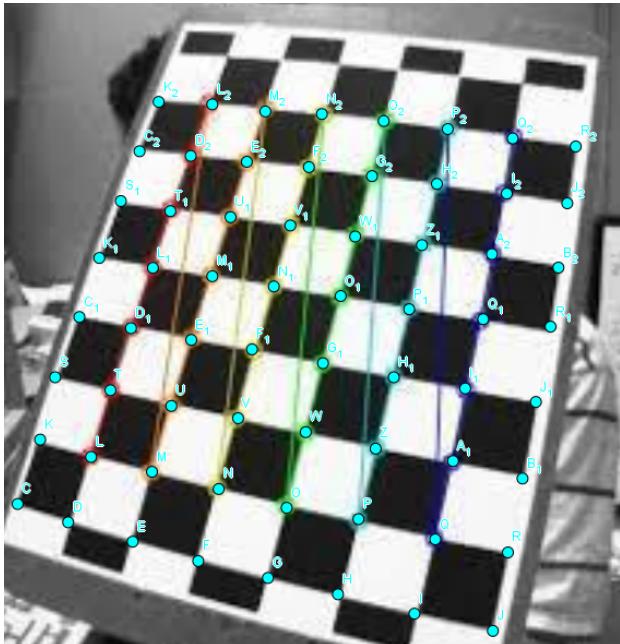


Abbildung 7.12: Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts (GRFIK AUSTAUSCHEN BILD IS KACKE)

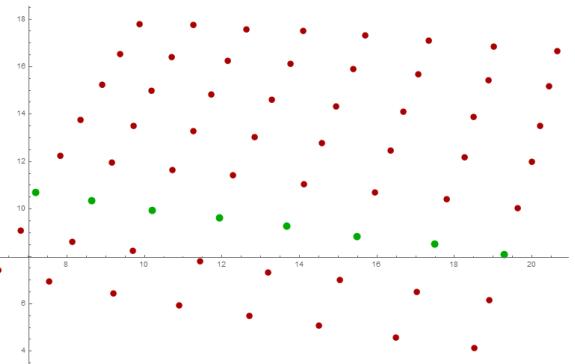


Abbildung 7.13: Algorithmisch detektierte Linie der dritten i-Reihe

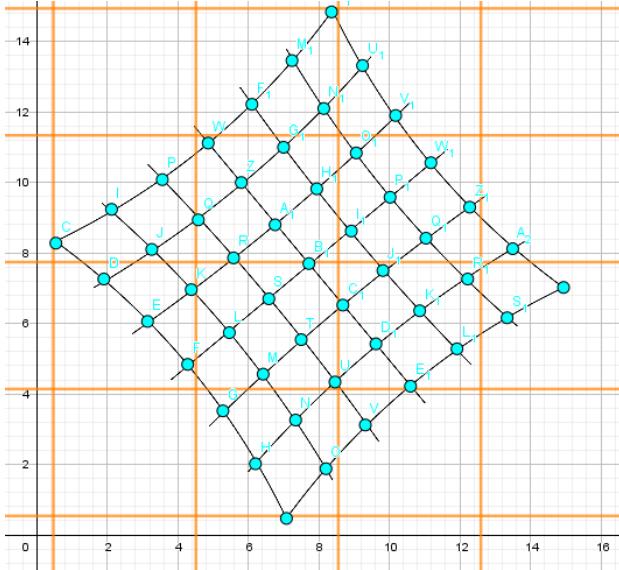


Abbildung 7.14: Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts

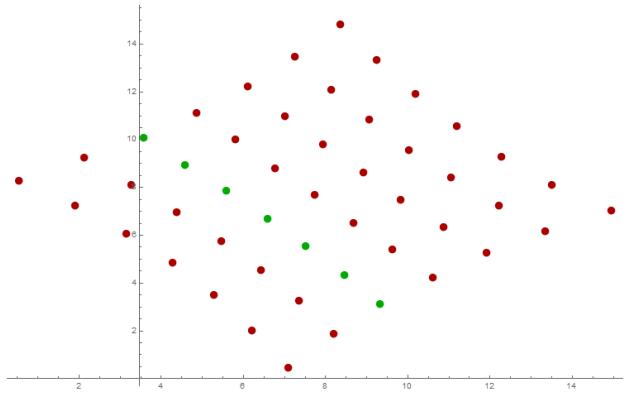


Abbildung 7.15: Algorithmisch detektierte Linie der dritten i-Reihe

7.3 Modulübersicht

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

Abbildung 7.16: Klassendiagramm

FindStartVectors	ConvexHull	PointCloud={}, StartPointCloudKeys=< >, VecI,VecJ,countI,countJ, Start, nextI,nextJ,	<ul style="list-style-type: none"> Die Punkte der Zellen ($i = 1, j = \text{All}$) und ($i = \text{all}, j = 1$) werden in eine neue Liste namens StartPointCloud gespeichert. Die Liste wird zweimal durchlaufen <ul style="list-style-type: none"> Alle Punkte welche sich in den Zellen $j = 1$ und $i = \text{all}$ aufhalten. Aus ihnen wird der geringste i-Wert ermittelt (VecI) Alle Punkte welche sich in den Zellen $j = \text{all}$ und $i = 1$ aufhalten. Aus ihnen wird der geringste j-Wert ermittelt (VecJ) Die Punkte mit den geringsten Werten werden in VecI und VecJ gespeichert. Jetzt wird die Liste nochmals zweimal durchgegangen. <ul style="list-style-type: none"> Alle Punkte welche sich in den Zellen $j = 1$ und $i = \text{all}$ aufhalten werden durchgegangen. Aus ihnen wird derjenige Wert ermittelt, welcher einen Wert für J besitzt der kleiner ist als der 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 Alle Punkte welche sich in den Zellen $j = \text{all}$ und $i = 1$ aufhalten. Aus ihnen wird derjenige Wert ermittelt, welcher einen Wert für i besitzt der kleiner ist als der momentan i-Wert von VecJ und dessen j-Wert kleiner ist als der j-Wert von VecJ plus einem Offset. Dieser Wert ist das neue VecJ VecI und VecJ ergeben den gleichen Punkt und somit ist der Startwert gesetzt.
------------------	------------	---	--

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

Abbildung 7.18: Klassendiagramm

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

Abbildung 7.19: 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 7.20: Klassendiagramm

8 Fazit - Conclusion

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. ζ 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 Verbindungsgerade von C und M mit der Bildebene I	6
2.3	Ein Weltkoordinatensystem (O, δ) mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O)$ wird zu einem dazu verschobenen und rotiertem Kamerakoordinatensystem (C, β) 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, δ) mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3)$, das Kamerakoordinatensystem (C, β) mit $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3)$, das Bildebenenkoordinatensystem (I, τ) mit $\tau = (\hat{t}_1, \hat{t}_2)$ und das Sensorkoordinatensystem (S, σ) 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_δ bezüglich eines Weltkoordinatensystems (O, δ) befindet sich auf der Ebene π , welche durch die Achsen \hat{d}_1 und \hat{d}_2 aufgespannt wird. M_δ wird auf I und I' projiziert. Es entstehen die Bildpunkte m_τ und m'_τ	13
3.2	Auf der Geraden durch C und \vec{m}_τ , befinden sich alle möglichen Punkte für m_β . m_β wird auf Grund seiner Unbestimmtheit als γm_τ 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_δ ist der Objektpunkt im 3D-Raum und m_τ und m'_τ sind die jeweiligen Abbildungen dieses Punktes auf den Bildebenen. Die Verbindungsvektoren zwischen C, C' und M_δ 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	21
4.2	Synthetisches Beispiel Top-Down-Ansicht	22
4.3	Simulierte Abbildung eines Quaders auf zwei Kameras	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° gedreht	26

4.10	Optimale Triangulierung: Beide Geraden Treffen sich in einem Punkt im 3D-Raum	27
4.11	29
4.12	29
4.13	Veranschaulichung der Skaleninvarianz und dessen Auswirkung auf die geometrische Form und Größe der Objekte	29
4.14	30
4.15	30
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. Die Abbildungen entstand aus dem in <i>Mathematica</i> [22] implementierten Algorithmus.	30
4.17	Rechteckiger Bildsensor mit darauf sich befindendenden quadratischen Sensorelementen. Vergleiche [8]	30
4.18	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
4.19	C und C' haben die selbe Auflösung eingestellt	32
4.20	C und C' haben unterschiedliche Auflösungen eingestellt	32
4.21	C und C' haben die selbe Auflösung eingestellt	34
4.22	C und C' haben unterschiedliche Auflösungen eingestellt. C mit K und C' mit K'_1 . . .	34
4.23	C mit K und C' mit K'_2	34
4.24	C mit K und C' mit K'_3	34
4.25	Die Abbildung zeigt die rekonstruierte Szenen des synthetischen Beispiels mit K'_3 als intrinsische Parameter für C'	35
5.1	Ablaufdiagramm für die reelle Rekonstruktion	36
5.2	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
5.3	a	38
5.4	b	38
5.5	Epipolarlinien ohne <i>Epipolar-constraint</i> im Bild der Canon 6D	40
5.6	Epipolarlinien ohne <i>Epipolar-constraint</i> im Bild der Canon 60D	40
5.7	Die Abbildung zeigt, dass die Epipolarlinien auf der Aufnahme von C , nach dem Erzwingen der Singularität in der normierten \hat{F} , alle durch einen Epipol verlaufen	41
5.8	Die Abbildung zeigt, dass die Epipolarlinien auf der Aufnahme von C' , nach dem Erzwingen der Singularität in der normierten \hat{F} , alle durch einen Epipol verlaufen	41
5.9	Die Abbildung zeigt die Epipolarlinien in C nachdem die Fundamentalmatrix \bar{F} mit $F = T' \bar{F} T$ denormalisiert wurde	41
5.10	Die Abbildung zeigt die Epipolarlinien in C' nachdem die Fundamentalmatrix \bar{F} mit $F = T' \bar{F} T$ denormalisiert wurde	41
5.11	a)	42
5.12	b)	42
5.13	a) Die rückprojizierten Strahlen der ungenauen korrespondierenden Punkte m_σ und $m'_{\sigma'}$ sind Windschief zueinander und treffen sich nicht in einem Punkt M_δ im Raum. b) Die korrespondierenden Bildpunkte m_σ und $m'_{\sigma'}$ erfüllen nicht den <i>Epipolar-Constraint</i> . Die Epipolarlinie $l' = Fm$ ist die korrespondierende Epipolarlinie zu m_σ und $l = F^T m'$ ist die korrespondierende Epipolarlinie zu $m'_{\sigma'}$. Da weder m_σ noch $m'_{\sigma'}$ auf der Epipolarlinie zum jeweils korrespondierenden Punkt liegt, kommt es zu keinem Schnittpunkt der rückprojizierten Strahlen	42
5.14	Die Abbildung zeigt die zwei korrespondierenden Epipolarlinien \hat{l} und \hat{l}' mit den gesuchten Punkten \hat{m}_σ und $\hat{m}'_{\sigma'}$	43

5.15 Rekonstruierte Szene, unskaliert in Pixeleinheiten	47
5.16 Rekonstruierte Szene, unskaliert, in Pixeleinheiten und in einem 2D-Plot angezeigt	47
5.17 Zeigt die Die rekonstruierte Matrix T' bei unveränderter Auflösung. Die Auflösungen von C_δ und C'_δ sind die selben.	49
5.18 Zeigt die rekonstruierte Matrix T' wenn K' mit einem Verhältnis von $[5 : 2]$ skaliert wurde	49
5.19 Zeigt die rekonstruierte Matrix T' wenn K' mit einem Verhältnis von $[1 : 2]$ skaliert wurde	49
5.20 Zeigt die rekonstruierte Matrix T' wenn K' mit einem Verhältnis von $[1.2 : 2.3]$ skaliert wurde	49
5.21 Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[5 : 2]$ skaliert wurde	50
5.22 Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[1 : 2]$ skaliert wurde	50
5.23 Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[1.2 : 2.3]$ skaliert wurde	51
 6.1 Beispiel eines Rektifizierten Bildes mit Scanlinien	52
6.2 Arbeitsprozess der Szenenrekonstruktion mit Rektifizierung	53
6.3 Entstehung einer Disparitätskarte	54
6.4 Aufnahmen zweier Kameras mit den selben Auflösungen, Kamera eins(Grün) und Kamera(rot) zwei gelten jeweils $\zeta = 1$	59
6.5 Epipole für Kamera eins und Kamera zwei vor der Rektifizierung	60
6.6 H_p und H'_p Transformation	60
6.7 H_p und H'_p Transformation mit Epipolarlinien	60
6.8 $H_r H_p$ und $H'_r H'_p$ Transformation	62
6.9 $H_r H_p$ und $H'_r H'_p$ Transformation mit Epipolarlinien	62
6.10 Die Verbindungslien \overline{bd} und \overline{ca} sollen so ausgerichtet werden, dass sie orthogonal zueinander stehen.	63
6.11 $H_s H_r H_p$ und $H'_s H'_r H'_p$ Transformation	64
6.12 $H_s H_r H_p$ und $H'_s H'_r H'_p$ Transformation mit Epipolarlinien	64
6.13 Aufnahmen zweier Kameras mit unterschiedlichen Auflösungen.Für Kamera eins(Grün) gilt $\zeta_x = \zeta_y = 1$ und für Kamera zwei(rot) gilt $\zeta'_x = \zeta'_y = 2$	65
6.14 Transformation H_p und H'_p angewandt auf Bilder unterschiedlicher Auflösungen	65
6.15 Transformation $H_r H_p$ und $H'_r H'_p$ angewandt auf Bilder unterschiedlicher Auflösungen	65
6.16 Transformation $H_s H_r H_p$ und $H'_s H'_r H'_p$ angewandt auf Bilder unterschiedlicher Auflösungen	66
6.17 Transformation $H_s H_r H_p$ und $H'_s H'_r H'_p$ angewandt auf Bilder unterschiedlicher Auflösungen mit Epipolarlinien	66
6.18 Rektifizierung mit $\zeta'_x = 2.3$ und $\zeta'_y = 3.2$	66
6.19 Rektifizierung mit $\zeta'_x = 2.3$ und $\zeta'_y = 3.2$	66
 7.1 Startpunktsuche in Schachbrettpunkten	68
7.2 Finden der Startvektoren in Schachbrettpunkten	70
7.3 Klassendiagramm	71
7.4 Klassendiagramm	72
7.5 Klassendiagramm	72
7.6 Bild eines Tonnenförmig verzeichneten Schachbretts	73
7.7 Algorithmisch detektierte Linie der dritten i-Reihe	73
7.8 Bild eines perspektivisch verzerrtem Schachbretts	73
7.9 Algorithmisch detektierte Linie der dritten i-Reihe	73
7.10 Bild eines Kissenförmig verzeichnetem Schachbretts	74
7.11 Algorithmisch detektierte Linie der dritten i-Reihe	74
7.12 Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts(GRFIK AUSTAUSCHEN BILD IS KACKE)	74
7.13 Algorithmisch detektierte Linie der dritten i-Reihe	74
7.14 Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts	75
7.15 Algorithmisch detektierte Linie der dritten i-Reihe	75

7.16 Klassendiagramm	75
7.17 Klassendiagramm	75
7.18 Klassendiagramm	76
7.19 Klassendiagramm	76
7.20 Klassendiagramm	76

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. 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] Bernd KITT. *Effiziente Schätzung dichter Bewegungsvektorfelder*, volume Band 027. KIT Scientific Publishing, 2013.
- [22] Wolfram Research, Inc. Mathematica, Version 11.1.1. Champaign, IL, 2018.
- [23] Olaf Dössel. *Bildgebende Verfahren in der Medizin*, volume 2. Auflage. Springer-Verlag Berlin Heidelberg, 2016.
- [24] 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.
- [25] 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.
- [26] 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.
- [27] James Demmel Dinesh Manocha. Algorithms for intersecting parametric and algebraic curves 1. *ACM Transactions on Graphics (TOG)*, Volume 13 Issue 1, Pages 73-100, 1994.
- [28] Luca Irsara Andrea Fusiello. Euclidean epipolar rectification of uncalibrated images. Eurac researc, IT.
- [29] Carlos Villagrá Arnedor Antonio Javier Gallego Sanchez, Rafael Molina Carmona. *Scene reconstruction and geometrical rectification from stereo images*. Januar 2005, uploaded by Antonio Javier Gallego Sánchez on 21 May 2014, ResearchGate.
- [30] 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.
- [31] MathWorks. Mathworks documentation, rectify stereo images.
- [32] Richard I. Hartley. Euclidean reconstruction from uncalibrated views. G.E. CRD, Schenectady, NY, 12301.
- [33] MathWorks. Mathworks documentation, disparity.
- [34] Dongqing Li, editor. *Encyclopedia of Microfluidics and Nanofluidics*, pages 999–999. Springer US, Boston, MA, 2008.
- [35] Jürgen Adamy Christian Voigt. *Formelsammlung der Matrizenrechnung*. Oldenburg Verlag München Wien, 2007.
- [36] 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.
- [37] James Demmel Dinesh Manocha. Algorithms for interesting parametric and algebraic curves 1: Simple intersections. *ACM Transactions of Graphics*:73–100, 1994.