
Szenenkonstruktion aus stereoskopischen Bildquellen gleicher und verschiedener Auflösungen

Erarbeitet von Studenten und Studentinnen
im Rahmen der Abschlussarbeit

Masterarbeit der Fakultät

Studiengang Semester Max Mustermann 222222

Betreut von: Prof. Dr. Mustermann

Disclaim here



Fakultät XYZ der Hochschule Furtwangen
Sommersemester - Wintersemester 2015

Inhaltsverzeichnis

1 Einleitung	5
2 Model der Bildaufnahme mit einer Kamera	7
2.1 Lochkameramodell zur Abbildung eines Punktes auf Bildebene	7
2.2 Koordinatentransformation	8
2.3 Aufname mit einer willkürlichen Kameraorientierung	12
3 Geometrische Beziehungen zwischen Punktekorrespondenzen	14
3.1 Korrespondenzanalyse für Punkte auf einer Ebene (Homographie)	14
3.2 Korrespondenzanalyse für willkürliche Punkte ?im Raum? (Epipolare Geometrie)	16
3.3 geometrische Erläuterung der Epipolargeometrie	17
3.4 Bestimmung von Homographie und Fundamentalmatrix aus Punktekorrespondenzen .	18
4 Virtuelle Rekonstruktion	22
4.1 Erstellen eines virtuellen Stereoaufbaus	22
4.2 Bildanalyse	25
4.2.1 Bestimmung der Abbildungsvorschriften	25
4.2.2 Bestimmung der externen Kameraparameter	26
4.2.3 Szenenrekonstruktion durch Triangulation	28
5 Auswirkungen von unterschiedlichen Kameraauflösungen	33
5.1 Abbildungsunterschiede	33
5.2 Auswirkungen auf die Epipolargeometrie	35
5.3 Minimalbeispiel mit unterschiedlichen Kameraauflösungen	36
6 Relle Rekonstruktion	40
6.1 Arbeitsprozess	40
6.2 Normalized-eight-Point-Algorithm	42
6.2.1 Singularity-Constraint der Fundamentalmatrix	44
6.2.2 Singularity- Constraint der essentiellen Matrix	46
6.3 Szenenrekonstruktion	46
6.3.1 Minimieren der Kostenfunktion durch Sampson-approximation	47
6.4 Ergebnisse einer Stereoanalyse mit Kameras unterschiedlicher Auflösung	51
7 Vergleich entwickelter Rekonstruktions Algorithmus mit bereits vorhandenen (Matlab)	56
7.0.1 Projektive Transformation	60
7.0.2 Ähnlichkeitstransformation	65
7.0.3 Scherungstransformation	67
8 Punktesortierung in Schachbrettmustern	71
8.1 Algorithmus zur Punktesortierung in verzeichneten Schachbrettbildern	71
8.1.1 Vorläufiges Klassendiagramm	72
8.1.2 Beispiele	76
9 Fazit - Conclusion	79
10 Nächste Schritte - next steps	80
11 Protocol - 10.11.2015	81

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, welcher auch die Möglichkeit von unterschiedlichen Bildauflösungen zwischen den Bildquellen in Betracht zieht. 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]. 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 Relle 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 mir 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.

(aktualisieren schreiben!!!) Die Kapitel 1 bis 5 umfassen die theoretischen mathematischen Hintergründe, welche für das Verständnis der implementierten Kalibrierungs- und Rekonstruktionsalgorithmen vorhanden sein müssen. Kapitel 6 befasst sich dann umfassend mit dem für die virtuelle Rekon-

struktion entwickelten Algorithmus für die Kamerakalibrierung und Szenenrekonstruktion. Kapitel 7 befasst sich dann mit der Kalibrierung und Rekonstruktion, wenn die virtuellen Kameras unterschiedliche Auflösungen aufweisen. In Kapitel 8 wird der entwickelte Algorithmus auf reale Bilddaten angewandt. In Kapitel 9 werden die Auflösungen der realen stereoskopischen Bilder künstlich verändert und die Ergebnisse verifiziert. Kapitel 10 beschreibt einen Algorithmus zur Sortierung von zuvor detektierten Eckpunkten eines Schachbretts, welcher im Zuge der Korrespondenzanalyse von Bildpunkten in Stereoskopischen Aufnahmen entstanden ist.

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.

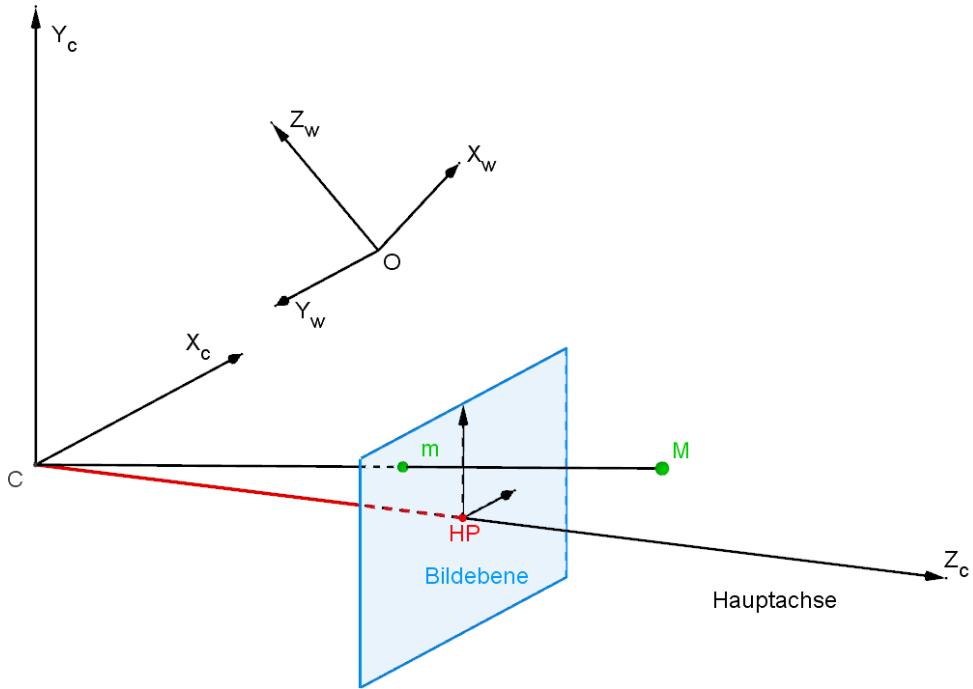


Abbildung 2.1: Schematik eines Abbildenden Systems. Ein Punkt M im Weltkoordinatensystem O wird durch eine Kamera C aufgenommen. Diese Aufnahme wird durch eine Projektion, die als Verbindungslinie von M zu C dargestellt ist und M auf m abbildet, beschrieben.

Ein Punkt M in einem dreidimensionalen Weltkoordinatensystem wird mit Hilfe der Kamera, die in einem eigenen dreidimensionalen Kamerakoordinatensystem beschrieben wird, auf die Bildebene I projiziert, welches durch ein zweidimensionales Bildkoordinatensystem beschrieben ist. 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 Bildebene

Mit Hilfe des Lochkameramodells wird die Abbildung eines Objektes auf die Bildebene beschrieben. Das Modell beruht ausschließlich auf der geometrischen Optik und vernachlässigt physikalische Effekte wie Beugung oder die Auswirkung der Linse[8]. 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 zur Hauptachse und der Schnittpunkt der Hauptachse mit der Bildebene

bildet den Hauptpunkt HP . Der Hauptpunkt ist der Ursprung des Bildebenekoordinatensystems. 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 .

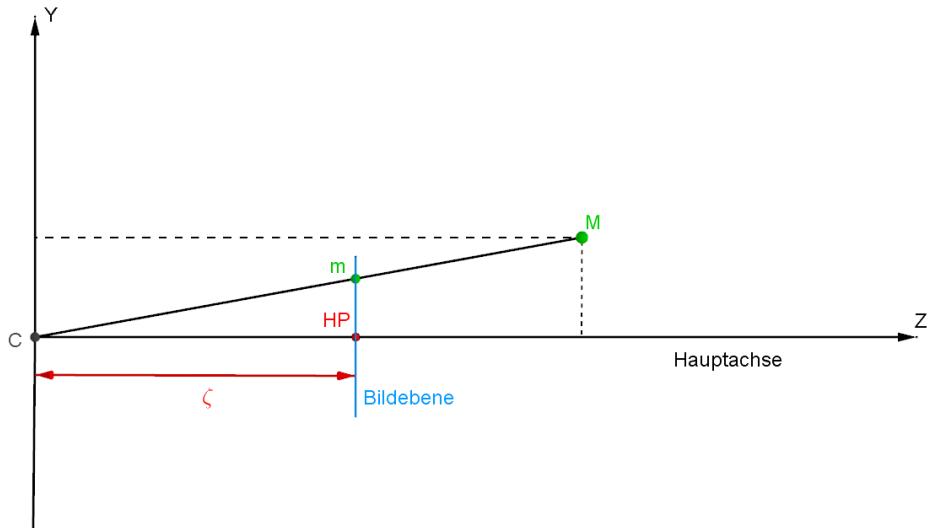


Abbildung 2.2: Die Abbildung zeigt einen Querschnitt des beschriebenen Lochkameramodells. Zu sehen ist das Projektionszentrum C der Kamera. C ist gleichzeitig das Kamerazentrum und bildet den Ursprung für das Kamerakoordinatensystem. ζ beschreibt den Abstand des Projektionszentrums zur Bildebene. Die 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 Bildebenekoordinatensystem. 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 beschrieben.

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

Die Koordinaten auf der 2 dimensionale Bildebene werden häufig als homomogene Koordinaten angegeben. Dazu werden die Koordinaten mit Z normiert und somit die Koordinaten auf die Ebene $(x, y, 1)^T$ projiziert. Zur Vereinfachung wird zuletzt nur die x,y Koordinaten des entstandenen Bildes angegeben. Gleichung ?? beschreibt somit die Abbildung eines Punktes auf die Bildebene.

2.2 Koordinatentransformation

Um ein Punkt von einem übergeordneten Weltkoordinatensystem in ein bestimmtes zum Weltkoordinatensystem verdrehtes 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 $(C, \beta) = \beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3, C)$ beschrieben.

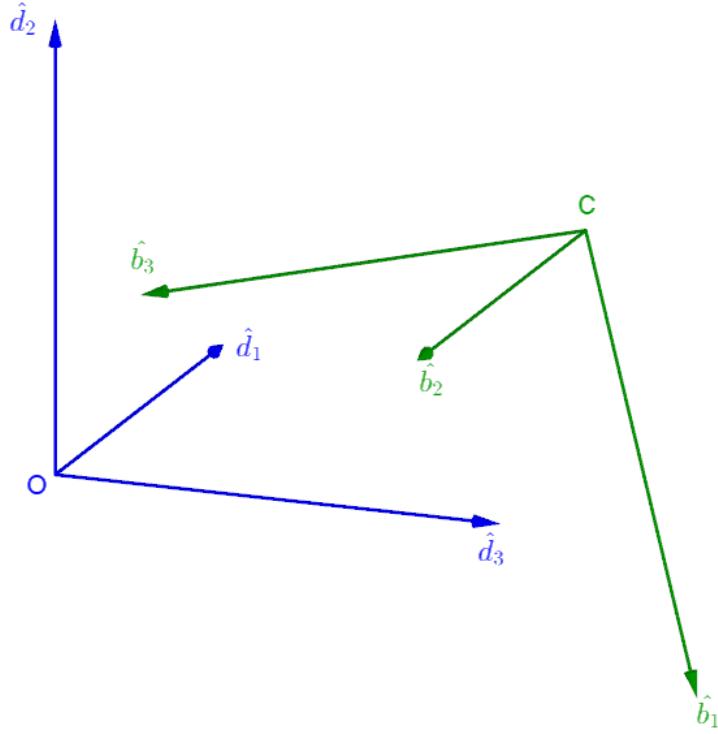


Abbildung 2.3: Ein Weltkoordinatensystem (O, δ) mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O)$ wird zu einem dazu verschobenen und rotiertem Kamerakoordinatensystem (C, β) mit $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3, C)$ transformiert

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 inversen Matrix D_β^{-1} kann verwendet werden um P_β aus P_δ zu berechnen.

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

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

Handelt es sich um ein kartesisches Koordinatensystem, so gilt $D_\beta^{-1} = D_\beta^T$ und die transponierten Matrix kann für die Koordinatentransformation benutzt werden.

Es werden hier kompakte vierdimensionale Vektoren eingeführt um die Transformation von Welt- in Kamerakoordinaten zu beschreiben. Der bekannte dreidimensionale Vektor wird mit dem Ursprung des Koordinatensystems erweitert.

$$\delta = \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \\ C \end{pmatrix} \quad (2.13)$$

Einmal wird wie bereits angefangen mit Spaltenvektoren gearbeitet, das selbe Verfahren wird dann noch einmal mit Zeilenvektoren dargestellt. Beide Ansätze funktionieren nach dem selben Prinzip. Der Unterschied ist die Darstellung der Matrizen. Beim arbeiten mit Programmen wie Beispielsweise *Matlab* ist es wichtig zu wissen, welche Darstellung benutzt wird. *MatLab* arbeitet mit Spaltenvektoren, während im entstandenen Algorithmus dieser Arbeit mit Zeilenvektoren gearbeitet wurde. Zunächst wird also weiter mit Spaltenvektoren verfahren. Wir nehmen an, dass der Punkt $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3, C)$ im Kamerakoordinatensystem und der Punkt $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O)$ im Weltkoordinatensystem identisch sind. Die Koordinatensysteme sind zueinander Verdreht und der Ursprung des Kamerakoordinatensystems ist bezüglich des Ursprungs des Weltkoordinatensystems verschoben. Der Translationsvektor entspricht den Koordinaten des Ursprungs des Kamerakoordinatensystem $\vec{V} = C_\beta = (C_{\beta,1}, C_{\beta,2}, C_{\beta,3})^T$. Die Transformation lässt sich in diesem vierdimensionalen Raum ausdrücken als

$$\begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \\ C \end{pmatrix} = \begin{bmatrix} b_{11} & b_{21} & b_{31} & 0 \\ b_{12} & b_{22} & b_{32} & 0 \\ b_{13} & b_{23} & b_{33} & 0 \\ C_{\beta,1} & C_{\beta,2} & C_{\beta,3} & 1 \end{bmatrix} \begin{pmatrix} \hat{d}_1 \\ \hat{d}_2 \\ \hat{d}_3 \\ O \end{pmatrix} = R_\delta \quad (2.14)$$

Die Transformationsmatrix R_δ setzt sich aus den Komponenten der Rotationsmatrix aus Gleichung ?? und dem Translationsvektor \vec{V} zusammen. Diese Parameter beschreiben die Kameraposition und Kameraorientierung, welche als extrinsische Kameraparameter bekannt sind.

Für eine Rücktransformation von Kamera in Weltkoordinaten muss die Inverse von R gebildet werden. Diese Inverse lässt sich in karthesischen Koordinaten durch die transponierten Rotationsmatrix D und der Inversen des Translationsvektor \vec{V} wie folgt ausdrücken.

$$\rightsquigarrow \begin{pmatrix} \hat{d}_1 \\ \hat{d}_2 \\ \hat{d}_3 \\ O \end{pmatrix} = \begin{bmatrix} b_{11} & b_{21} & b_{31} & 0 \\ b_{12} & b_{22} & b_{32} & 0 \\ b_{13} & b_{23} & b_{33} & 0 \\ -(C_{\beta,1}, C_{\beta,2}, C_{\beta,3})C^{-1} & 1 \end{bmatrix} \begin{pmatrix} \hat{b}_1 \\ \hat{b}_2 \\ \hat{b}_3 \\ C \end{pmatrix} \quad (2.15)$$

Das selbe Verfahren mit Zeilenvektoren führt zu den Gleichungen 2.16 und 2.17.

$$(\hat{b}_1, \hat{b}_2, \hat{b}_3, C) = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O) \cdot \begin{bmatrix} b_{11} & b_{21} & b_{31} & C_{\beta,1} \\ b_{12} & b_{22} & b_{32} & C_{\beta,2} \\ b_{13} & b_{23} & b_{33} & C_{\beta,3} \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.16)$$

Daraus folgt, dass für den Fall der Rücktransformation gilt:

$$\rightsquigarrow (\hat{d}_1, \hat{d}_2, \hat{d}_3, O) = (\hat{b}_1, \hat{b}_2, \hat{b}_3, C) \begin{bmatrix} b_{11} & b_{12} & b_{13} & \\ b_{21} & b_{22} & b_{23} & - \begin{pmatrix} C_{\beta,1} \\ C_{\beta,2} \\ C_{\beta,3} \end{pmatrix} C^{-1} \\ b_{31} & b_{32} & b_{33} & \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.17)$$

2.3 Aufname mit einer willkürlichen Kameraorientierung

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

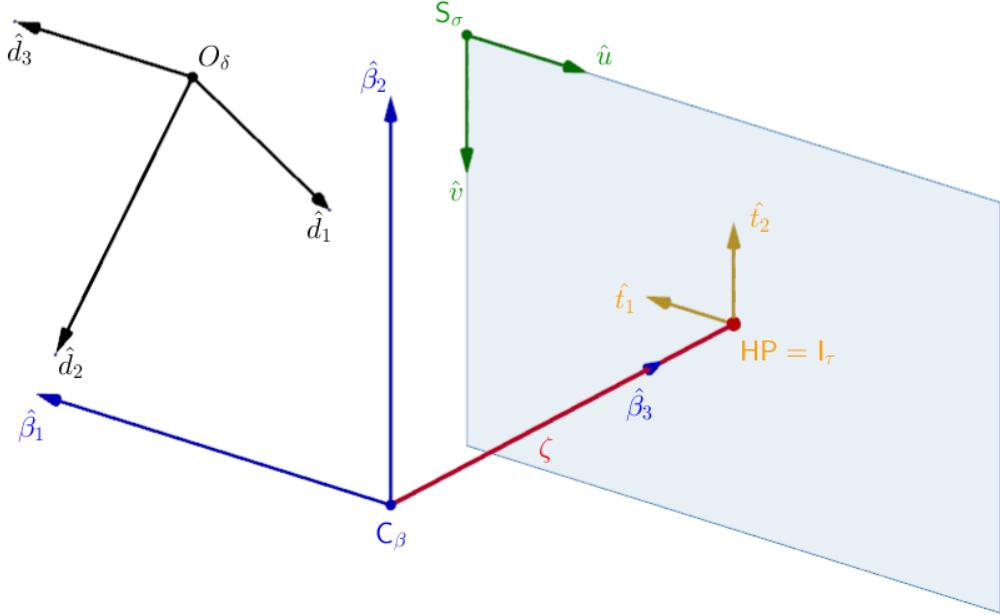


Abbildung 2.4: Das Schaubild zeigt die einzelnen Koordinatensysteme in einem Lochkameramodell. Das Weltkoordinatensystem (O, δ) mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O)$, das Kamerakoordinatensystem (C, β) mit $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3, C)$, das Bildebenenkoordinatensystem (I,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

Für die Projektion eines Punktes $M_\delta = (M_{\delta x} M_{\delta y} M_{\delta z} O_\delta)$ bezüglich des Weltkoordinatensystems in einen Punkt m_σ bezüglich des Sensorkoordinatensystems wird eine Projektionsmatrix P definiert. P entsteht durch die Vereinigung von Transformationsmatrix R und der Kameramatrix K [3].

Für die Transformation der Weltkoordinaten in Kamerakoordinaten gilt Gleichung ??:

$$(M_{\beta x} M_{\beta y} M_{\beta z} C_\beta) = (M_{\delta x} M_{\delta y} M_{\delta z} O_\delta) \cdot R \quad (2.18)$$

Nach der Transformation eines Punkts M_δ zu M_β im Kamerakoordinatensystem, erfolgt die Projektion in die 2D-Bildecke entsprechend Gleichung 2.20. Die Kameramatrix lässt den Ursprung eines Koordinatensystems unverändert und die Kameramatrix K_0 lässt sich erweitern:

$$K_0 = \begin{bmatrix} \zeta & 0 & 0 & 0 \\ 0 & \zeta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.19)$$

Ein Punkt $m_\tau = (m_{\tau x}, m_{\tau y}, m_{\tau z}, C)$ auf der Bildecke lässt sich aus M_β abbilden mit

$$(m_{\tau x}, m_{\tau y}, m_{\tau z}, C) = (M_{\beta x} M_{\beta y} M_{\beta z} C_\beta) \cdot K_0 = (\zeta M_{\beta x} \zeta M_{\beta y} M_{\beta z} C_\beta). \quad (2.20)$$

Um Bildpunkt m_τ bezüglich eines zweidimensionalen Bildebenenkoordinatensystems anzugeben, wird die Bildecke mit der Tiefekomponente $M_{\beta z}$ normiert, sodass m_τ auf den zweidimensionalen Raum

der Bildebene gemappt wird und der Bildpunkt m_τ mit $m_\tau = [\zeta \frac{M_{\beta x}}{M_{\beta z}}, \zeta \frac{M_{\beta y}}{M_{\beta z}}, 1] = [X_\tau, Y_\tau]$ entsteht.

Zuletzt folgt die Transformation der Bildebenenkoordinaten auf den Sensorchip. Der Sensorchip besteht aus einer Ansammlung von Sensorelementen. Diese Sensorelemente können verschiedenste Formen annehmen. Die meisten Sensorchips bestehen aus rechtwinkligen, rechteckigen Sensorelementen. Deswegen 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 Sensorkoordinatensystem in der Regel an einer Ecke des Sensorchips definiert, sodass die Transformation von Bildebenenkoordinaten in Sensorkoordinaten auch eine Translation $(V_{\sigma x}, V_{\sigma y})$ aufweist. Für einen Punkt $m_\sigma = (u, v, w, S)$. u auf dem Sensorkoordinatensystem lassen sich die folgenden Bedingungen herleiten.

$$u = m_{\tau x} k_x \quad (2.21)$$

$$v = m_{\tau y} k_y \quad (2.22)$$

$$w = m_{\tau z} \quad (2.23)$$

$$S_\sigma = C_\beta + V_{\sigma, x} + V_{\sigma, y} \quad (2.24)$$

$k_x = \frac{1}{lx}$ und $k_y = \frac{1}{ly}$ ist die Pixeldichte in $\frac{\text{Pixel}}{m}$. Anzumerken ist, dass die Bildebene und die Sensorebene die Punkte ausschließlich im zweidimensionalen Raum definiert und die z-Komponente keine zusätzlich Information trägt. Deswegen wurde keine Umskalierung der z-Achse vorgenommen. Diese Bedingung lassen sich in folgender Koordinatentransformationsmatrix ausdrücken:

$$m_\sigma = (u, v, w, S) = (m_{\tau x}, m_{\tau y}, m_{\tau z}, C) \begin{bmatrix} k_x & 0 & 0 & V_{\sigma, x} \\ 0 & k_y & 0 & V_{\sigma, y} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.25)$$

$$= (m_{\tau x}, m_{\tau y}, m_{\tau z}, C) R_\sigma \quad (2.26)$$

Mit Hilfe der Transformationsmatrix R_σ kann ein Punkt von der Bildebene auf das Sensorelement projiziert werden. Setzt man alle Transformation zusammen so kann eine Projektionsmatrix P gebildet werden, welche die Projektion von einem Punkt im Weltkoordinatensystem auf die Sensorebenen beschreibt.

$$m_\sigma = m_\tau \cdot R_\sigma = M_\beta \cdot K_0 R_\sigma = M_\delta \cdot R K_0 R_\sigma = M_\delta \cdot P \quad (2.27)$$

Die Projektionsmatrix $P = R K_0 R_\sigma$ setzt sich aus einer Koordinatentransformationsmatrix R , der Kameramatrix K_0 und einer weiteren Transformation zusammen. Häufig werden die letzten beiden Matrizen zu einer erweiterten Kameramatrix $K = K_0 R_\sigma$ zusammengefasst. Die erweiterte Kameramatrix beinhaltet die Pixeldichte k_x, k_y und die Brennweite ζ . Diese Parameter werden im folgenden als intrinsische Kameraparameter bezeichnet. Die Koordinatentransformationsmatrix R wird im Gegensatz zu den sogenannten extrinsischen Kameraparameter, der Kameraposition und Orientierung, definiert. Sind sowohl die intrinsischen wie auch extrinsischen Kameraparameter vorbestimmt kann P aufgestellt werden und ein Punkt auf die Sensorebene der entsprechenden Kamera projiziert werden.

3 Geometrische Beziehungen zwischen Punktekorrespondenzen

Das Ziel dieser Masterarbeit ist es Punkte im dreidimensionalen Raum aus einer Stereoskopischen Aufnahme zweier Kamerä zu rekonstruieren. Bis jetzt wurde die Bildaufnahme einer einzigen Kamera betrachtet. Jedoch kann eine Kamera allein nicht räumlich sehen. Um Räumlichkeit 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 ein und des selben projektiven Punkts im Raum sind. Für diese Punkte muss dann eine gemeinsame Abbildungsvorschrift aufgestellt werden. Mit diesen Abbildungsvorschriften ist es möglich eine Beziehung zwischen zwei projizierten Bildpunkten herzuleiten, ohne dass intrinsischen und extrinsischen Kameraparameter bekannt sind. 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 dann wiederum Schlüsse auf die Kameraparameter der beiden Kamerä gezogen werden.

Es seien $m_\tau = \begin{pmatrix} m_{\tau,1} \\ m_{\tau,2} \\ m_{\tau,3} \end{pmatrix}$ die homogenen Koordinaten eines Punktes auf der Bildebene (I, τ) und $m'_{\tau'} = \begin{pmatrix} m'_{\tau',1} \\ m'_{\tau',2} \\ m'_{\tau',3} \end{pmatrix}$

ein Punkt der projektiv transformierten Bildebene (I', τ') . Dann gilt

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

$$Hm_\tau = \begin{bmatrix} h_1^T \cdot m_{\tau,1} \\ h_2^T \cdot m_{\tau,2} \\ h_3^T \cdot m_{\tau,3} \end{bmatrix} \quad (3.2)$$

$$\rightsquigarrow m'_{\tau'} = Hm_\tau = \begin{bmatrix} h_{11}m_{\tau,1} + h_{12}m_{\tau,2} + h_{13}m_{\tau,3} \\ h_{21}m_{\tau,1} + h_{22}m_{\tau,2} + h_{23}m_{\tau,3} \\ h_{31}m_{\tau,1} + h_{32}m_{\tau,2} + h_{33}m_{\tau,3} \end{bmatrix} \quad (3.3)$$

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

Im folgenden werden zunächst zwei Fälle für die Beschreibung von Abbildungsvorschriften unterschieden. Bei den Fällen wird angenommen, dass sowohl intrinsische und extrinsische Parameter bekannt sind. Im ersten Fall wird davon ausgegangen, dass die 3D-Punkte im Raum auf einer Ebene liegen und auf die Bildebene von zwei zueinander verschobenen und rotierten Bildebene abgebildet werden. Im zweiten Fall werden die Punkte eines komplexeren 3D-Objektes auf die Bildebene abgebildet. Danach werden die Herleitungen beider Beispiele für den Fall von unbekannten Kameraparameter aufgezeigt.

3.1 Korrespondenzanalyse für Punkte auf einer Ebene (Homographie)

Es wird der direkte Zusammenhang zwischen zwei Bildpunkten auf unterschiedlichen Kamerabildebene aus einem un demselben Punkt hergeleitet.

In diesem Abschnitt wird davon ausgegangen dass die z-Komponente des Ursprungspunktes $M_{\delta z}$ bekannt ist. Dies wird in Abbildung XX schematisch dargestellt.

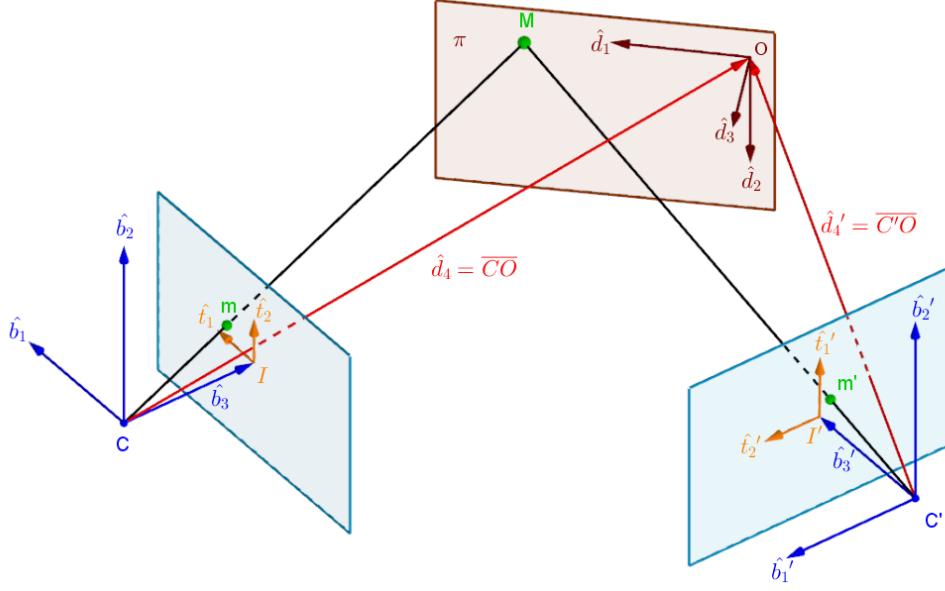


Abbildung 3.1: Veranschaulichung der Homographie bei zwei verschiedenen translatierten und rotierten Kameras.

Ein Punkt auf der normierten Bildebene $m_{n,\tau} = (m_{n,\tau x}, m_{n,\tau y}, 1, I)$ lässt sich somit eindeutig auf die Bildebene erweitern mit

$$m_\tau = (m_{\tau x}, m_{\tau y}, m_{\tau z}, I) = M_{\delta z}(m_{n,\tau x}, m_{n,\tau y}, 1, I) = M_{\delta z}m_{n,\tau} \quad (3.5)$$

Für 2 korrespondierende Punkte m_τ und m'_τ auf 2 unterschiedlichen Bildebenen gilt der folgende Zusammenhang

$$m_\tau = M_{\delta z}m_{n,\tau} = M_\delta \cdot P \quad (3.6)$$

$$m'_\tau = M_{\delta z}m'_{n,\tau} = M_\delta \cdot P' \quad (3.7)$$

Die Projektionsmatrix P beschreibt in diesem Fall nur die Abbildung auf die Bildebene und wird durch $P = K_0 R$ mit der Kameramatrix K_0 und der Transformationsmatrix R aus Kapitel 2 zusammen. Durch das Anwenden der inversen von P und P' von rechts auf die Gleichung, werden diese zu

$$m_\tau \cdot P^{-1} = M_{\delta z}m_{n,\tau}P^{-1} = M_\delta \quad (3.8)$$

$$m'_\tau P'^{-1} = M_{\delta z}m'_{n,\tau}P'^{-1} = M_\delta \quad (3.9)$$

Auf der rechten Seite der Gleichung steht schließlich derselbe Ursprungspunkt, sodass wir beide Gleichungen zusammenfassen können.

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

$$m_{n,\tau}P^{-1} = m'_{n,\tau}P'^{-1} \quad (3.11)$$

Multiplizieren wir nun P von rechts auf die Gleichung erhalten wir

$$m_{n,\tau} = m'_{n,\tau} P'^{-1} P = m'_{n,\tau} H \quad (3.12)$$

die sogenannte Homographiematrix $H = P'^{-1}P$. Diese Matrix beschreibt die direkte, eindeutige Abbildung von einem Bildpunkt auf der 2 dimensionalen Bildebene I' auf den Bildpunkt auf der Bildebene I. Mittels der inversen H^{-1} kann diese Abbildung auch rückwärts gemacht werden.

Mann könnte das gleichungssystem hier ausformulieren um auf die bestimmung der Kameraparameter einzugehn, wenn du willst (denke das hilft hier zwar nicht aber schaden tut warhscheinlich auch nicht so sehr)

3.2 Korrespondenzanalyse für willkürliche Punkte ?im Raum? (Epipolare Geometrie)

In diesem Kapitel wird der Fall einer Korrespondenzanalyse für Punkte behandelt, bei denen die z-Komponente des abzubildenden Punktes $M_{\delta z}$ nicht bekannt ist. Die Punkte können somit überall im Raum liegen und wir definieren $M_{\delta z} = k$, mit einer freien Variablen k. Für die Transformation eines Punktes im normalisierten Bildkoordinatensystem in die nicht normalisierten Bildkoordinaten gilt somit:

$$m_\tau = (m_{\tau x}, m_{\tau y}, m_{\tau z}, I) = k(m_{n,\tau x}, m_{n,\tau y}, 1, I) = M_{\delta z} m_{n,\tau} \quad (3.13)$$

(im nicht normalisierten Koordinatensystem befindet sich nun der Punkt auf der durch C und m(Tau) gehenden Geraden, durch die Art wie die normierung funktioniert befindet sich der Objektpunkt auf geraden durch)

Für 2 korrespondierende Punkte m_τ und m'_τ auf 2 unterschiedlichen Bildebenen kann folgende gleichung hergeleitet werden

$$m_\tau = km_{n,\tau} = M_\delta \cdot P \quad (3.14)$$

$$m'_\tau = k'm'_{n,\tau} = M_\delta \cdot P' \quad (3.15)$$

Wie in Kapitel 3.1 lässt sich folgende Beziehung herleiten

$$km_{n,\tau} P^{-1} = k'm'_{n,\tau} P'^{-1}. \quad (3.16)$$

(diese gleichung repräsentiert die such nach dem Schnittpunkt der beiden Geraden wie oben gemeint)
Dies Gleichung können wir mit der Homographimatrix H wiederum in

$$m_{n,\tau} = \lambda m'_{n,\tau} H \quad (3.17)$$

mit $\lambda = k/k'$. λ beschreibt somit den relativen Unterschied der oben definierten $M_{\delta z}$. Wenn $(\lambda m' H = entspricht der epipollinie)$

(WICHTIG: dann noch zeigen was es mit den normierten Bildpunkten auf sich hat!!! sobald die rechnung wieder da ist)

3.3 geometrische Erläuterung der Epipolargeometrie

Auf Abbildung 3.2 wird die oben beschriebene mathematische Herleitung der Epipolargeometry grafisch dargestellt. (Kurz darlegen was ist was in der Grafik in bezug zu den Formeln)

Ein Objektpunkt M_δ wird auf die Bildebene I und I' der beiden Kameras C und C' projiziert. Es entstehen die zueinander korrespondierenden Bildpunkte m_τ und $m'_{\tau'}$. Die durch die Bildpunkte m_τ und $m'_{\tau'}$ und den entsprechenden Epipolen e und e' verlaufenden Epipolarlinien, sind zueinander korrespondierende Epipolarlinien. Die zum Punkt m_τ korrespondierende Epipolarlinie l' beinhaltet alle zu m_τ möglichen korrespondierenden Punkte, darunter eben auch der eindeutig korrespondierende Punkt $m'_{\tau'}$.

Die Epipolargeometrie beschreibt weiterhin eine Beziehung zwischen einem Bildpunkt m und dessen korrespondierender Epipolarlinie l' . Hier kann eine Verbindung zu den willkürlichen Tiefen hergestellt werden und anhand des Bildes gezeigt werden und anhand von Abbildung 3.3

Diese Beziehung ist der sogenannte *Epipolar-Constraint*[3, 11, 1]. (beschreibt der nicht quasi das selbe wie die Abbildungsvorschrift, also bzw hier könnte man die grafik in bezug zur gleichung bringen) Der *Epipolar-Constraint* sagt aus, dass wenn ein 3D-Bildpunkt M_δ sich entlang seiner Verbindungsgeraden $\overline{CM_\delta}$ auf die Bildebene I zu bewegt, so ändert sich die Position des Bildpunktes m_τ auf I nicht, während der korrespondierende Punkt $m'_{\tau'}$ sich entlang seiner Epipolarlinie bewegt. In Abbildung 3.3 ist m_τ mit $m_{\tau,i}$ bezeichnet.

Ist der Constraint erfüllt, dann sagt der *Epipolar-Constraint* aus, dass Bildpunkt $m'_{\tau'}$, auf der zu m_τ korrespondierenden Epipolarlinie l' liegt und somit ein möglicher korrespondierender Punkt zu m_τ [3, 11, 5, 1].

Ist der *Epipolar-Constraint* erfüllt, so wird gleichzeitig der Suchaufwand nach weiteren Korrespondenzen reduziert, da somit nur noch eine eindimensionale Suche, entlang der Epipolarlinie, anstatt einer zweidimensionalen durchgeführt werden muss. Dieser neue *Constraint* wird auch als *Coplanarity-Constraint* oder Koplanaritätsbeschränkung bezeichnet. Er sagt aus, dass die Projektionszentren der Kameras und die korrespondierenden Bildpunkte auf einer und der selben Epipolarebene liegen müssen [11].

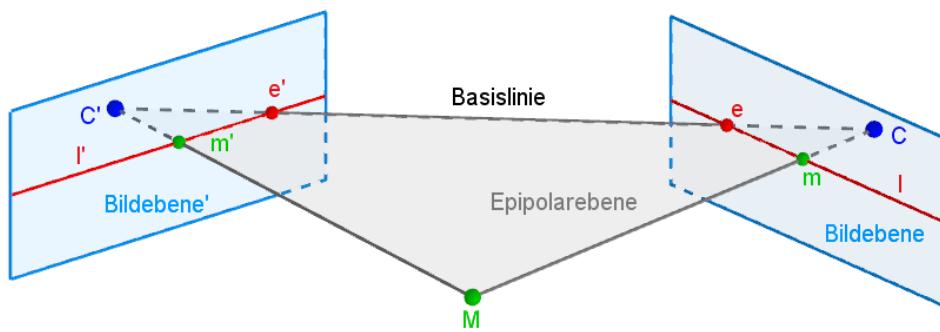


Abbildung 3.2: C und C' sind die Projektionszentren zweier Kameras. Beide Kameras besitzen jeweils eine Bildebene. Die Basislinien verbindet die Projektionszentren der Kameras. Der Punkt an welchem die Basislinie die Bildebene schneidet, wird als Epipol bezeichnet. Durch den Epipol verlaufen alle Epipolarlinien des Bildes. M ist der Objektpunkt im 3D-Raum und m_1 und m_2 sind die jeweiligen Abbildungen dieses Punktes auf den Bildebene. Die Verbindungsvektoren zwischen C, C' und M bilden die sogenannte Epipolarebene[5, 3, 1].

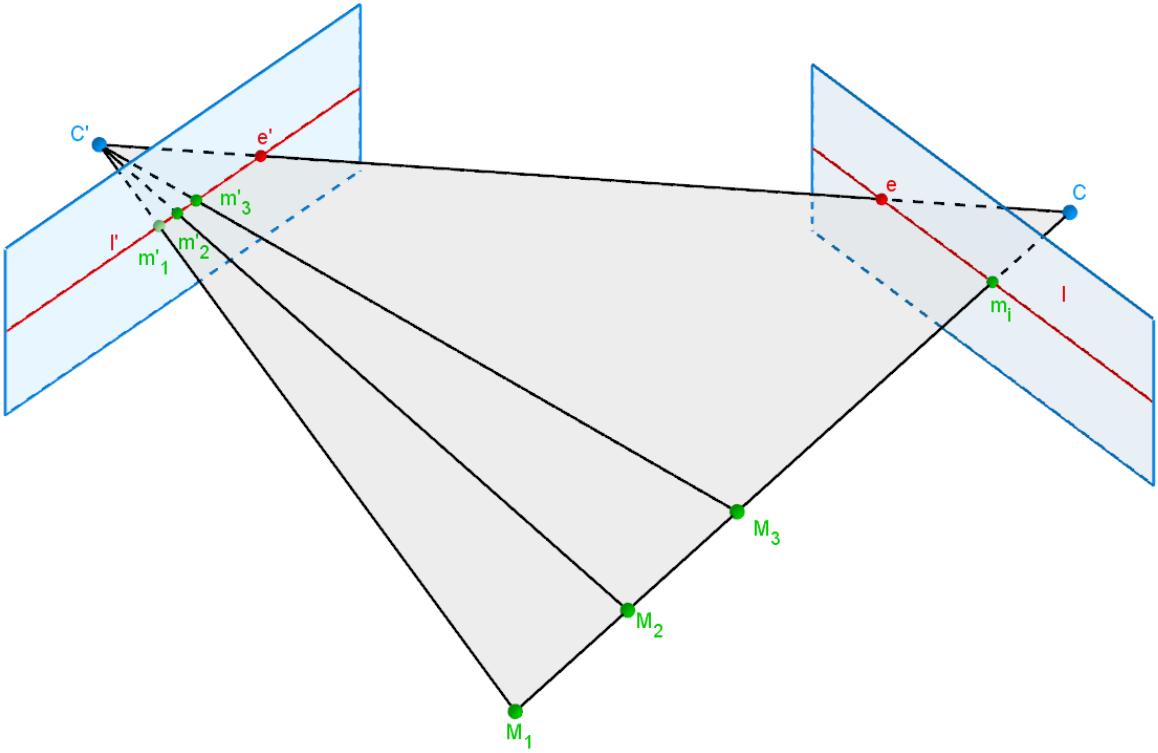


Abbildung 3.3: Die Objektpunkte M_1, M_2 und M_3 werden in I' als m'_1, m'_2 und m'_3 abgebildet, während sie in I immer den selben Bildpunkt m_1 ergeben.

3.4 Bestimmung von Homographie und Fundamentalmatrix aus Punktekorrespondenzen

(sollte vllt zwischen rein) Hier sagen wo man die Punkte herbekommt

Im folgenden soll nun gezeigt werden, wie Homographien und Fundamentalmatrizen aus Punktekorrespondenzen gewonnen werden können. Für essentielle Matrizen gilt das selbe Verfahren wie für die Fundamentalmatrizen nur sind hier die Punktekorrespondenzen in normierten Koordinaten gegeben. Es wird davon ausgegangen, dass die Transformationsmatrizen R und R' sowie die Kameramatrizen K und K' nicht bekannt sind. Des Weiteren gehen wir davon aus, dass zuvor mindestens acht korrespondierende Punkte aus den jeweiligen Bildpaaren detektiert wurden.

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.

$$H \cdot m_\tau = m'_\tau \quad (3.18)$$

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} m_\tau \end{bmatrix} = \begin{bmatrix} m'_{\tau'} \end{bmatrix} \quad (3.19)$$

$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \cdot \begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} \quad (3.20)$$

Aus Gleichung 3.16 lässt sich das folgende Gleichungssystem aufstellen.

$$h_{11}x + h_{12}y + h_{13}z = \lambda x' \quad (3.21)$$

$$h_{21}x + h_{22}y + h_{23}z = \lambda y' \quad (3.22)$$

$$h_{31}x + h_{32}y + h_{33}z = \lambda z' \quad (3.23)$$

Da mit zweidimensionalen homogenen Bildkoordinaten gearbeitet wird und somit z und $z' = 1$ sind, ergibt sich für die letzte Zeile $h_{31}x + h_{32}y + h_{33}z = \lambda$. Setzt man das anstelle von λ ein, so ergeben sich die folgenden Gleichungen:

$$h_{11}x + h_{12}y + h_{13}z = (h_{31}x + h_{32}y + h_{33}z) \cdot x' \quad (3.24)$$

$$h_{21}x + h_{22}y + h_{23}z = (h_{31}x + h_{32}y + h_{33}z) \cdot y' \quad (3.25)$$

Für den Aufbau von A werden beide Ausdrücke noch nach Null aufgelöst, so dass sich die Gleichungen 4.34 und 4.35 aus 4.32 und 4.33 ergeben. Pro korrespondierendem Punktpaar m_τ und $m'_{\tau'}$ ergeben sich somit zwei Gleichungen:

$$h_{11}x + h_{12}y + h_{13}z - (h_{31}x + h_{32}y + h_{33}z) \cdot x' = 0 \quad (3.26)$$

$$h_{21}x + h_{22}y + h_{23}z - (h_{31}x + h_{32}y + h_{33}z) \cdot y' = 0 \quad (3.27)$$

$$\rightsquigarrow h_{11}x + h_{12}y + h_{13}z - h_{31}x \cdot x' - h_{32}y \cdot x' - h_{33}z \cdot x' = 0 \quad (3.28)$$

$$\rightsquigarrow h_{21}x + h_{22}y + h_{23}z - h_{31}x \cdot y' - h_{32}y \cdot y' - h_{33}z \cdot y' = 0 \quad (3.29)$$

Die entstandenen Gleichungen werden dann nach folgendem Schema in die Koeffizientenmatrix A eingetragen.[5, 3, 14, 8]

$$\begin{pmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & x_1x'_1 & y_1x'_1 & 1 \cdot x'_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & x_1y'_1 & y_1y'_1 & 1 \cdot y'_1 \\ & & & \ddots & & & & & \\ & & & & \ddots & & & & \\ x_i & y_i & 1 & 0 & 0 & 0 & x_ix'_i & y_ix'_i & 1 \cdot x'_i \\ 0 & 0 & 0 & x_i & y_i & 1 & x_iy'_i & y_iy'_i & 1 \cdot y'_i \end{pmatrix} \cdot \begin{pmatrix} h_1 \\ h_2 \\ \vdots \\ h_i \end{pmatrix} = 0 \quad (3.30)$$

Gesucht wird nun ein Vektor \vec{x} , für den gilt das $A \cdot \vec{x} = 0$. Der gesuchte Vektor \vec{x} entspricht dem Kern der Koeffizientenmatrix und ist ein Spaltenvektor mit insgesamt neun Einträgen, welche in die 3x3-Homographiematrix eingetragen werden können[3, 14].

Das Verfahren mit welchem sowohl F als auch E geschätzt werden können, ähnelt in seinem Ablauf dem der Homographiebestimmung. Das Verfahren wird hier allgemein als der *eighth-Point-Algorithm* bezeichnet. Der *8-Point-Algorithm* ist eine lineare Technik die angewandt wird, um die Fundamentalmatrix, sowie auch die essentielle Matrix, aus $n \geq 8$ Punkten schätzen zu können [11, 3]. Der Algorithmus benötigt $n \geq 8$ Punkte, um ein valides Ergebnis zu liefern [3, 4]. Das Ergebnis und jedes seiner Vielfachen ist eine mögliche Lösung für F . Da auch jedes Vielfache eine gültige Lösung ist, kann aus der Herleitung der Epipolareometrie geschlossen werden. (**sagen warum aber ich weiß nicht wie genau man das sagt ohne wieder auszuschweifen**). Der Algorithmus wird am Beispiel für die Bestimmung von F veranschaulicht. Zunächst wird auch hier eine Koeffizientenmatrix aus den detektierten Punktekorrespondenzen gebildet.

$$m'^T_{\sigma'} \cdot F \cdot m_{\sigma} = 0 \quad (3.31)$$

$$F = \begin{bmatrix} f_{11} & f_{122} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \quad (3.32)$$

$$\begin{bmatrix} x'_n & y'_n & 1 \end{bmatrix} \cdot \begin{bmatrix} f_{11} & f_{122} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix} \cdot \begin{bmatrix} x_n \\ y_n \\ 1 \end{bmatrix} = 0 \quad (3.33)$$

$$f_{11}x_nx'_n + f_{12}y_nx'_n + f_{13}x'_n + f_{21}x_ny'_n + f_{22}y_ny'_n + f_{23}y'_n + f_{31}x_n + f_{32}y_n + f_{33} = 0 \quad (3.34)$$

$$(x_nx'_n, y_nx'_n, x'_n, x_ny'_n, y_ny'_n, y'_n, x_n, y_n, 1) \cdot f = 0 \quad (3.35)$$

$$\begin{bmatrix} x_1x'_1 & y_1x'_1 & x'_1 & x_1y'_1 & y_1y'_1 & y'_1 & x_1 & y_1 & 1 \\ x_2x'_2 & y_2x'_2 & x'_2 & x_2y'_2 & y_2y'_2 & y'_2 & x_2 & y_2 & 1 \\ \vdots & \vdots \\ x_nx'_n & y_nx'_n & x'_n & x_ny'_n & y_ny'_n & y'_n & x_n & y_n & 1 \end{bmatrix} \cdot \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0 \quad (3.36)$$

Gesucht wird wieder ein Vektor \vec{f} , für den gilt das $A \cdot f = 0$. Der gesuchte Vektor \vec{f} entspricht 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 dazukommen ,dass das Gleichungssystem, welches für die Kernbestimmung beider Koeffizientenmatrizen gelöst wird, zu einem überbestimmten System wird. Ein System gilt als überbestimmt, wenn es durch mehr Gleichungen als Unbekannte beschrieben wird. Für die Koeffizientenmatrix für F und H hätte, dass zur Folge, dass die Koeffizientenmatrix in ihrem Rang steigt. Die Bestimmung des Kerns würde in beiden Fällen kein eindeutiges Ergebnis mehr liefern[3].

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} beziehungsweise gesucht, so dass $\|A \cdot x\|$ minimal wird [3, 15, 14]. Die Singulärwertzerlegung von A ist eine Faktorisierung der Matrix $A \in \mathbb{R}^{m \times n}$ der Form $A = U \cdot S \cdot V^T$ mit orthogonalen Matrizen $U \in \mathbb{R}^{m \times n}$ und $V \in \mathbb{R}^{m \times n}$ sowie mit einer Diagonalmatrix S .

$$S = \begin{pmatrix} s_1 & & \dots & 0 & 0 & \dots & 0 \\ & \ddots & & \ddots & \ddots & & \ddots \\ & & \ddots & \ddots & \ddots & & \ddots \\ & & & \ddots & \ddots & & \ddots \\ 0 & & \dots & s_r & 0 & \dots & 0 \\ 0 & & \dots & 0 & 0 & \dots & 0 \\ & & & \ddots & \ddots & & \ddots \\ & & & & \ddots & & \ddots \\ 0 & & \dots & 0 & 0 & \dots & 0 \end{pmatrix} \quad (3.37)$$

Die Diagonalmatrix S beherbergt die Singulärwerte der Matrix. Für die Singulärwerte in S soll für s_1 bis s_r gelten, dass $s_1 \geq s_2 \geq \dots \geq s_r \geq 0$ [15]. Dabei soll für die diagonalen Singulärwerte in S mit s_1 bis

s_r gelten, dass $s_1 \geq s_2 \geq \dots \geq s_r \geq 0$ [15]. Die Spalte von V^T , welche mit dem kleinsten Singulärwert von S korrespondiert, ergibt den Vektor \vec{x} , für den $\| A \cdot x \|$ minimal wird.

4 Virtuelle Rekonstruktion

(WICHTIG σ zu τ umschreiben, da ja gesagt wird das $S = I$, sonst kommt Verwirrung auf)

Für den in dieser Arbeit benutzen Ansatz des Szenenrekonstruktionsalgorithmus, wird davon ausgegangen, dass die intrinsischen Kameraparameter bekannt sind. Der Grund dafür ist, dass in dieser Arbeit ein Ansatz entwickelt werden sollte, welcher die Möglichkeit von unterschiedlichen Auflösungen der Kameras mit einbeziehen sollte. Aus diesem Grund wurde ein Ansatz für die Szenenrekonstruktion entwickelt, welcher keine Rektifizierung der Bilder mit einschließt.

Im folgenden sollen die Funktionsweisen und Abläufe des entstandenen Szenenrekonstruktionsalgoritmus aufgezeigt werden. Hierzu wurde ein virtuelle Beispielszene gebaut mit einem 3D-Objekt und zwei simulierten zueinander rotierten und verschobenen Kameras. Das Objekt wird mathematisch auf die virtuellen Sensoren der beiden Kameras abgebildet. In Abbildung 4.1 werden die einzelnen Schritte des Algoirthmus nocheinmal schematisch kurz zusammengefasst. Die Schritte eins bis vier beschreiben den Aufbau der virtuellen Szene. In schritt drei der Einzelkalibrierung werden die intrinsischen Kameraparameter der beiden Kameras bestimmt. Ab hier setzt dann der Szenenrekonstruktionsalgoritmus an.

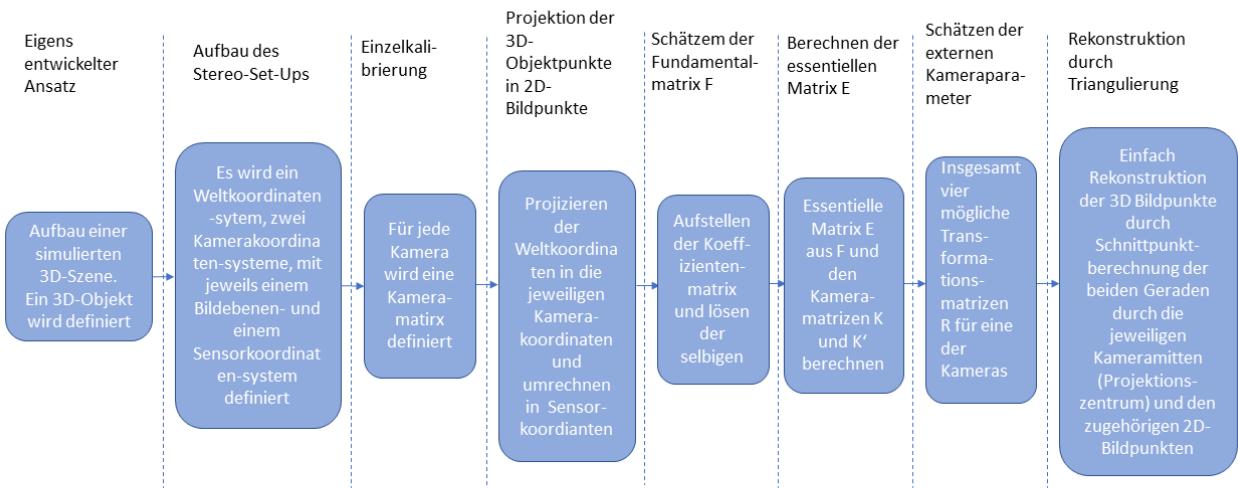


Abbildung 4.1: Arbeitsprozesse der Stereoanalyse bei Verwendung von eigens erstellten synthetischen Bilddaten

4.1 Erstellen eines virtuellen Stereoaufbaus

Als 3D-Objekt wurde ein Quader, in ein Weltkoordinatensystem (O, δ) mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3)$ positioniert. Des Weiteren wurden 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. Im weiteren Verlauf werden die beiden Kameras jeweils mit C oder C' bezeichnet, um

die Unterscheidung beider klar zu machen. C' ist von C entlang der x-Achse verschoben worden und um 45° Richtung C' rotiert. Die zwei Bildebene (I, τ) mit $\tau = (\hat{t}_1, \hat{t}_2, \hat{t}_3, I)$ und (I, τ) mit $\tau = (\hat{t}_1, \hat{t}_2, \hat{t}_3, I)$ befinden sich vor C und C' in positiver z-Richtung. Anzumerken ist, dass für das virtuelle Beispiel angenommen wurde, dass Bildebenekoordinatensystem und Sensorkoordinatensystem gleich sind und deshalb mit der vereinfachten Kameramatrix K_0 , siehe Kapitel Model der Bildaufnahme mit einer Kamera. Der schematische Aufbau der Szenen ist in den Abbildungen 4.2, 4.3 und 4.4 dargestellt.

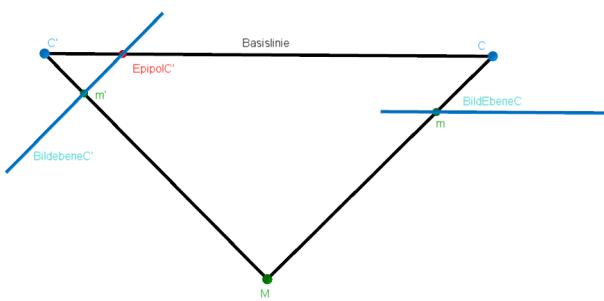


Abbildung 4.2: In der Abbildung ist der vereinfachte Stereoaufbau in einer Top-Down-Ansicht zu sehen

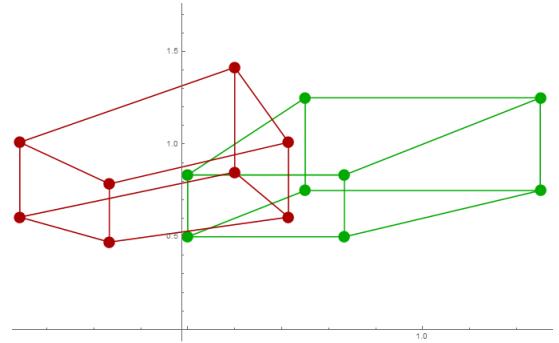


Abbildung 4.3: In Grün ist die Abbildung des Quaders auf der Bildebene I von C und in rot ist die Abbildung des Quaders auf der Bildebene I' von C' zu sehen

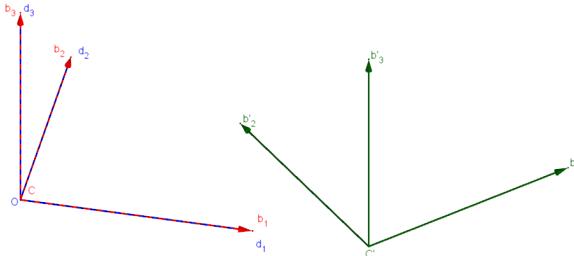


Abbildung 4.4: **Achsen falsch!!!** In Blau und Rot sind jeweils das Welt- und Kamerakoordinatensystem von Kamera eins zu sehen. In grün ist das gedrehte Koordinatensystem von Kamera 2 zu sehen.

Zunächst werden die extrinsischen Kameraparameter von C und C' definiert. C ist gegenüber O weder rotiert noch verschoben, C' dagegen ist entlang \hat{d}_1 verschoben und um \hat{b}_2 rotiert. Somit ergibt sich für $R = [D|V]$ von C :

$$V = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (4.1)$$

$$R = [I|V] \quad (4.2)$$

$$R = \begin{bmatrix} 1 & 0 & 0 & v_1 \\ 0 & 1 & 0 & v_2 \\ 0 & 0 & 1 & v_3 \end{bmatrix} \quad (4.3)$$

Und für $R' = [D'|V']$ von C' gilt:

$$R' = D'^T \cdot [I|V] \quad (4.4)$$

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

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

$$R' = \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) \\ 0 & 1 & 0 \\ \sin(\alpha) & 0 & \cos(\alpha) \end{pmatrix} \cdot \begin{pmatrix} 1 & 0 & 0 & -v'_1 \\ 0 & 1 & 0 & -v'_2 \\ 0 & 0 & 1 & -v'_3 \end{pmatrix} \quad (4.7)$$

$$R' = \begin{pmatrix} \cos(\alpha) & 0 & -\sin(\alpha) & -v'_1 \cos(\alpha) + v'_3 \sin(\alpha) \\ 0 & 1 & 0 & -v'_2 \\ \sin(\alpha) & 0 & \cos(\alpha) & -v'_1 \sin(\alpha) - v'_3 \cos(\alpha) \end{pmatrix} \quad (4.8)$$

Die Eckpunkte des Quaders $A_\delta, B_\delta, C_\delta, D_\delta, A'_\delta, B'_\delta, C'_\delta, D'_\delta$ sind bekannt. Neben den Eckpunkten des Quaders wird noch ein neunter Punkt E_δ außerhalb des Quaders platziert und zwar so, dass es zu keinen linearen Abhängigkeiten zwischen E_δ und den anderen Punkten kommt. So kann vermieden werden, dass die später aufgestellt Koeffizientenmatrix, zum berechnen der Fundamentalmatrix, einen Rang kleiner als acht bekommt und somit zwei linear unabhängige Lösungen ausgibt[3].

Um diese Punkte auf die Bildebenen (I, τ) und (I', τ) zu projizieren, müssen neben den Transformationsmatrizen R und R' noch die Kameramatrizen K und K' festgelegt werden.

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

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

Sind R, R', K und K' bekannt, können die Projektionsmatrizen P und P' gebildet werden.

$$P = K \cdot R \quad (4.11)$$

$$P = \begin{bmatrix} \zeta_C & 0 & 0 & 0 \\ 0 & \zeta_C & 0 & 0 \\ 0 & 0 & \zeta_C & 0 \end{bmatrix} \quad (4.12)$$

$$P' = K' \cdot R' \quad (4.13)$$

$$P' = \begin{bmatrix} \zeta_{C'} \cos(\alpha) & 0 & \zeta_{C'} \sin(\alpha) & -\zeta_{C'}(v'_1 \cos(\alpha) + v'_3 \sin(\alpha)) & 0 \\ 0 & 1 & 0 & \zeta_{C'} - v'_2 & 0 \\ \zeta_{C'} \sin(\alpha) & 0 & \zeta_{C'} \cos(\alpha) & -\zeta_{C'}(v'_1 \sin(\alpha) + v'_3 \cos(\alpha)) & 0 \end{bmatrix} \quad (4.14)$$

Die 3D-Punkte $A_\delta, B_\delta, C_\delta, D_\delta, A'_\delta, B'_\delta, C'_\delta, D'_\delta, E_\delta$, werden mit den Projektionsmatrizen P und P' auf die Bildebenen I und I' projiziert. Ein Beispiel für die entstehende Abbildung ist in Abbildung 4.3 zu sehen. Da festgelegt wurde dass Sensorkoordinatensystem und Bildebenenkoordinatensystem Deckungsgleich sind, ist somit das Objekt auf den Bildebenen der virtuellen Kameras projiziert und es kann der Szenenrekonstruktionsalgorithmus beginnen.

4.2 Bildanalyse

Der Szenenrekonstruktionsalgorithmus für die virtuelle Rekonstruktion ist in drei Abschnitte unterteilt. Zuerst wird aus den Punktekorrespondenzen die Fundamentalmatrix und die essentielle Matrix berechnet geschätzt. Aus der essentiellen Matrix werden die extrinsischen Kameraparameter extrahiert um so im letzten Schritt durch Triangulation die Szenenpunkte zu rekonstruieren.

4.2.1 Bestimmung der Abbildungsvorschriften

Zu aller erst wird die Fundamentalmatrix F aus den korrespondierenden Punkten bestimmt. Die korrespondierenden Punkte sind die jeweiligen Abbildungen der entsprechenden Eckpunkte. Anhand des in Kapitel 3 beschriebenen 8-Point-Algorithms, wird F bestimmt. Über F wird die essentielle Matrix E ermittelt. Voraussetzung dafür ist, dass die intrinsischen Kameraparameter bekannt sind.

Für die Rekonstruktion der externen Kameraparameter, gibt es verschiedene Ansätze [3]. Es ist zum einen möglich, ohne Vorwissen der Kameramatrizen oder Transformationsmatrizen, aus der Fundamentalmatrix über den sogenannten *Stratified approach* die komplette Projektionsmatrix P und P' zu ermitteln, jedoch ohne weitere Informationen über die 3S-Szene nur bis zu einem Projektiven Vieldeutigkeit[3].

Da in dieser Arbeit davon ausgegangen wird, dass die Kameras zuvor einzeln Kalibriert wurden und die intrinsischen Kameraparameter bereits bekannt sind, wird für die Rekonstruktion der externen Kameraparameter ein Ansatz verfolgt, in welchen die essentielle Matrix zum Einsatz kommt. Die essentielle Matrix ist eine Spezialform der Fundamentalmatrix und beschreibt den *epipolar constraint*, zwischen den normierten Bildebenenkoordinaten[3, 5, 1, 11, 4].

(vllt τ schreiben? keine verwirrung mit anfang, macht schlussendlich bis auf die werte keien unterschied)

$$\hat{m}'_{\sigma'}^T \cdot E \cdot \hat{m}_{\sigma} = 0 \quad (4.15)$$

Sind F und Kameramatrizen K und K' bekannt, kann die essentielle Matrix wie in Kapitel 3.3 gezeigt aus F bestimmt werden.

Die essentielle Matrix ist eine Fundamentalmatrix, welche zu einem paar normierter Projektionsmatrizen P mit $P = [I|0]$ und P' mit $P' = [R'|V']$ korrespondierend ist[3, 11, 1, 4]. Um eine Projektionsmatrix $P' = K'[R'|V']$ auf die normierte Form zu bringen, müssen die intrinsischen Parameter für K' bekannt sein.

$$m'_{\sigma} = P' \cdot M_{\delta} \quad (4.16)$$

$$m'_{\sigma} = K'[R'|V'] \cdot M_{\delta} \mid \cdot K'^{-1} \quad (4.17)$$

$$K'^{-1} \cdot m'_{\sigma} = K'^{-1} \cdot K'[R'|V'] \cdot M_{\delta} \quad (4.18)$$

$$\hat{m}'_{\sigma} = [R'|V']M_{\delta} \quad (4.19)$$

M_{δ} ist der ein Objektpunkt im 3D-Raum, welcher mit P' zum Bildebenenpunkt $m'_{\tau'}$ abgebildet wird. Nach der Normierung, wird \hat{m}'_{σ} als normierte Koordinate bezeichnet und die entstandene Projektionsmatrix $P' = [R'|V']$ als normierte Projektionsmatrix[3, 4]. Um E aus F zu gewinnen, werden die Kameramatrizen K und K' mit F multipliziert. Zu Erinnerung im Kapitel geometrische Erläuterung der Epipolargeometrie wurde genau diese Beziehung zwischen E und F hergestellt.

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

Das Ergebnis alle seine Vielfachen, sind mögliche Lösungen für die essentielle Matrix. Wie die Fundamentalmatrix muss auch die essentielle Matrix einen Rang von 2 haben. Des Weiteren darf eine

3x3-Matrix nur dann als essentielle Matrix bezeichnet werden, wenn die Singulärwerte, bestimmte Merkmale aufweisen. So müssen zwei der drei Singulärwerte gleich und die dritte null sein[3]. Des Weiteren muss für die Determinante gelten, dass $\det(E) = 0$ ist und die Quadratwurzel der Eigenwerte müssen wieder die Singulärwerte ergeben. Die essentielle Matrix kann, wie die Fundamentalmatrix auch, über den *8-Point-Algorithm* ermittelt werden. Jedoch müssen auch hier die intrinsischen Kameraparameter K und K' bekannt sein, außerdem müssen die Koordinaten in normierte Bildebenenkoordinaten umgerechnet werden.[3, 4].

4.2.2 Bestimmung der externen Kameraparameter

Mit der essentiellen Matrix ist es möglich die Transformationsmatrix R' zu ermitteln. Es wird davon ausgegangen, dass für R von C gilt $R = [I|0]$. Die aus E ermittelte Matrix R' beschreibt dann die Transformation von C' relativ zu C [3, 4]. Um die externen Kameraparameter zu bestimmen, wird zunächst die essentielle Matrix E mit Hilfe der Singulärwertszerlegung in drei Matrizen zerlegt.

$$E = U\Sigma V^T \quad (4.21)$$

Die Singulärwerte befinden sich in der mittleren Matrix $\Sigma = \text{diag}(\sigma_1, \sigma_2, \sigma_3)$ wieder. Damit die Matrix E sich essentielle Matrix nennen darf, muss für zwei der Diagonaleinträge für die Singulärwerte gelten das $\sigma_1 = \sigma_2$ und für σ_3 muss dann dementsprechend gelten, dass $\sigma_3 = 0$ anderes wäre sie keine singuläre Matrix.

Die Zerlegung der essentiellen Matrix in USV^T muss nicht zwingend eine eindeutige Lösung hervorbringen. Für E muss gelten, dass $\det(E) = 0$ ist. Es wird also vorausgesetzt, dass die Determinante der aus der SVD gewonnenen Matrizen $UV^T = 1$ ist. Angenommen aus der SVD von E ergibt sich für die Determinanten von $UV^T = -1$, so können die Determinanten der Matrizen U und V getrennt voneinander bestimmt werden. Sollte $\det(U) = -1$ oder $\det(V) = -1$ oder beides zusammen der Fall sein, so kann die jeweilige Matrix einfach mit -1 multipliziert werden.(Hab ich aus einer nicht zitierbaren quelle... wurde mal besprochen und eingebaut im code...)

E setzt sich zusammen aus einer Rotationsmatrize D und einer schiefsymmetrischen Matrix S [3].

$$E = [v]_x R \quad (4.22)$$

$$S = [v]_x \quad (4.23)$$

$$E = SR \quad (4.24)$$

Zur Schätzung von S und D werden des Weiteren die schiefsymmetrische Matrix W und die Blockdiagonale Matrix Z eingeführt[3]. Mit diesen Matrizen lassen sich die gesuchten D und S für C' rekonstruieren, jedoch nur bis zu einer gewissen Skaleninvarianz[3, 4].

$$W = \begin{pmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{pmatrix} \quad Z = \begin{pmatrix} 0 & 1 & 0 \\ -1 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad (4.25)$$

Mit dem Ergebnis der SVD von E lassen sich nun jeweils zwei mögliche Lösungen für S und D aufstellen.

$$S_1 = -UZU^T \quad D_1 = UW^TV^T \quad (4.26)$$

$$S_2 = UZU^T \quad D_2 = UWV^T \quad (4.27)$$

Um sicher zu gehen, dass es sich bei D_1 und D_2 auch um gültige Rotationsmatrizen handelt, kann eine Probe durchgeführt werden. Zum einen muss $D \cdot D^T = I_{3x3}$ sein. I_{3x3} steht für die 3x3-Einheitsmatrix. S_1 und S_2 sind jeweils schiefsymmetrische Matrizen, welche die Information für den noch gesuchten Vektor \vec{v} beherbergt[3].

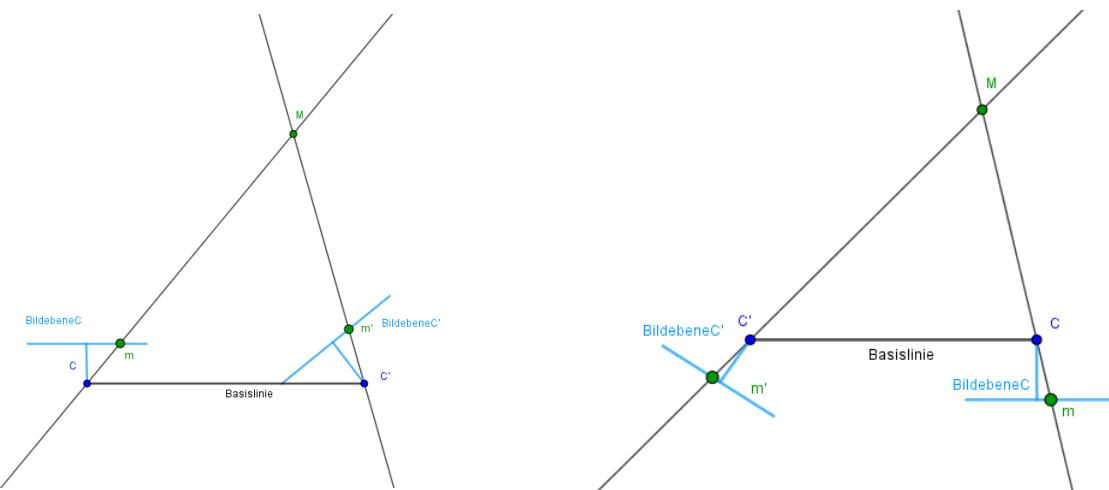
$$St = [v]_\times \cdot v = v \times v \quad (4.28)$$

Um v zu ermitteln muss lediglich der Kern von S_1 und S_2 gefunden werden. Die jeweiligen Ergebnisse für v_1 und v_2 sind bis auf ihre Vorzeichen die selben. Jedoch lässt sich hier \vec{x} nur bis auf eine Skaleninvarianz genau bestimmen.(Grund nochmal finden... vergessen wo der stand... und was er war...). Die externen Kameraparameter lassen sich wie gesagt nur bis zu einem Skalierungsfaktor genau bestimmen[3, 4]. Wie die Skaleninvarianz im viruellen Beispiel gelöst wurde, wird im nachfolgenden Abschnitt der Szenenrekonstruktion durch Triangulierung noch erklärt.

Letztendlich können, für die Rekonstruktion der externen Kameraparameter vier mögliche Lösungen für R gefunden werden.[3, 4]. 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].

$$R = |UWV^T| + \lambda v \quad \text{oder} \quad |UW^TV^T| + \lambda v \quad (4.29)$$

$$\text{oder} \quad |UWV^T| - \lambda v \quad \text{oder} \quad |UW^TV^T| - \lambda v \quad (4.30)$$



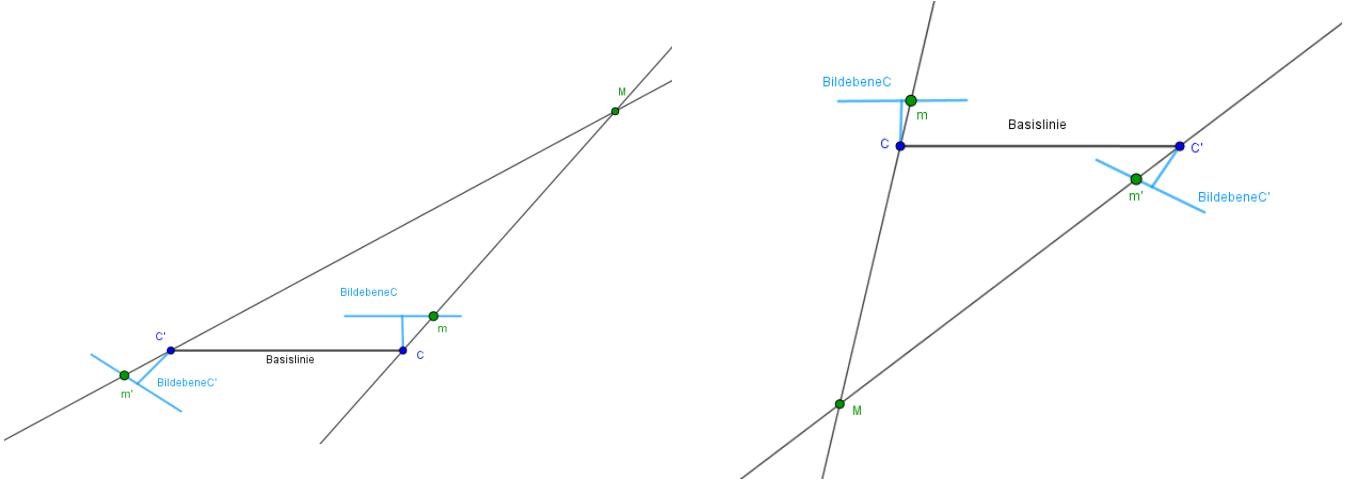


Abbildung 4.5: Die Abbildung zeigt die vier möglichen Lösungen für R .

(Noch einen Ansatz einbauen wie man die richtige Lösung algorithmisch auswertet? Im implementierte Algorithmus der arbeit wurden die Vier Lösungen jeweils rekonstruiert und alle ergebnisse ausgegeben.)

4.2.3 Szenenrekonstruktion durch Triangulation

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

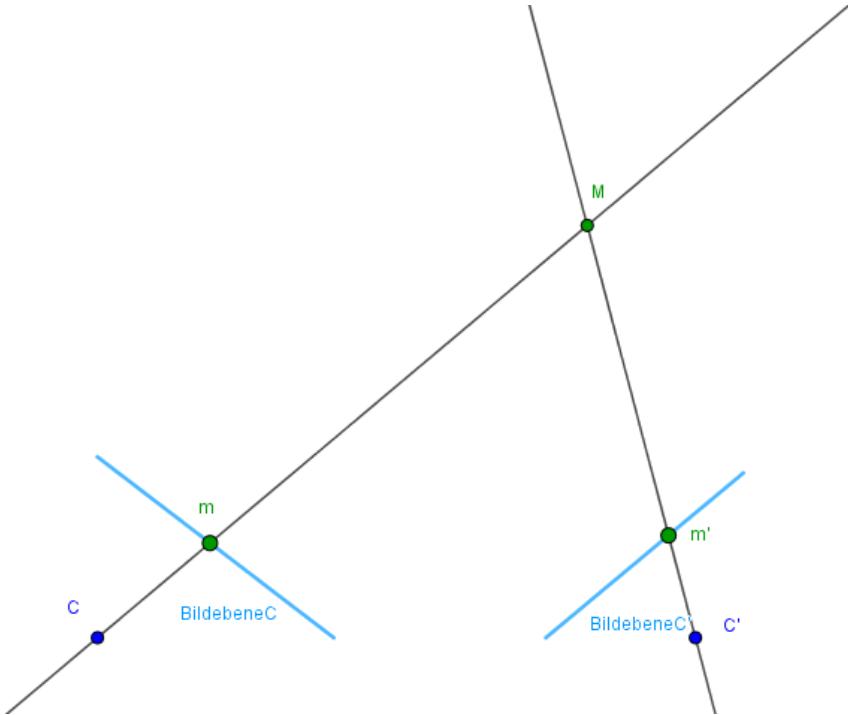


Abbildung 4.6: Optimale Triangulierung: Beide Geraden Treffen sich in einem Punkt im 3D-Raum

Da das Minimalbeispiel mit reinen Daten arbeitet, ist damit garantiert, dass die Linien sich in einem Schnittpunkt treffen. Anders hingegen wäre es in einem realen Beispiel mit korrespondierenden Punkten, welche Beispielsweise über einen SURF- Algorithmus detektiert wurden[18]. In Realbildern, können Bildfehler wie beispielsweise Rauschen nicht vermieden werden, des Weiteren können korrespondierende Punkte nicht immer exakt auf den Pixel genau bestimmt werden. Diese Fehler führen

dazu, dass wenn ein Schnittpunkt der Geraden durch die vermeintlichen korrespondierenden Punkte nicht gefunden werden kann, da die Geraden sich sehr wahrscheinlich nicht in einem Punkt treffen werden[18, 3].

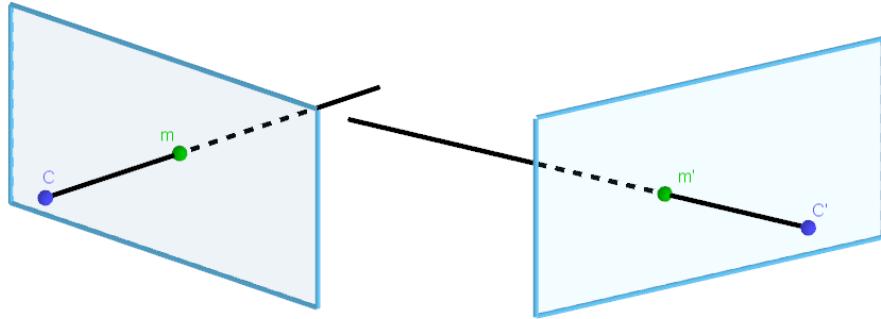


Abbildung 4.7: Durch Ungenauigkeiten in der korrespondierenden Punkte, verfehlten sich die Linien und es kommt zu keinem Schnittpunkt

Für diese Fälle gibt es mehrere Näherungsverfahren, wovon eines, im Kapitel Minimieren der Kostenfunktion durch Sampson-approximation, im Realbeispiel eingeführt wird[3]. In diesem Minimalbeispiel tritt der optimale Fall ein. Das bedeutet, dass ein zu Bildpunkt m korrespondierender Bildpunkt m' auf der zu m korrespondierenden Epipolarlinie l' liegt und somit garantiert ist, dass sich die Geraden \overline{mM} und $\overline{m'M}$ auf jedenfall in einem gemeinsamen Punkt schneiden. Um den Schnittpunkt beider Geraden zu berechnen, werden zum einen die Bildpunkte $A_\tau, B_\tau, C_\tau, D_\tau, A2_\tau, B2_\tau, C2_\tau, D2_\tau$ und $A_{\tau'}, B_{\tau'}, C_{\tau'}, D_{\tau'}, A2_{\tau'}, B2_{\tau'}, C2_{\tau'}, D2_{\tau'}$, sowie die korrekt ermittelte Projektionsmatrix P' von zuvor benötigt. Bei der Berechnung der externen Kameraparameter wurde festgesetzt, dass die Projektionsmatrix von Kamera eins $P = [I|0]$ und dementsprechend $P' = [R^T | -R^T V]$ die Transformation für Kamera zwei, ausgehend von Kamera eins ist. Also ist die Position von Kamera eins in Weltkoordinaten $C_\delta = [0 \ 0 \ 0 \ 1]^T$. Was jetzt noch fehlt ist die Position von Kamera zwei in Weltkoordinaten. Da die Szene dieses Minimalbeispiels durchkonstruiert wurde, ist C' eigentlich bekannt. Es soll nun aber angenommen werden, dass die Position von Kamera zwei nicht bekannt ist und diese wie im realen Fall erst einmal aus P berechnet werden muss. Um C' zu ermitteln, wird der Translationsvektor V aus $P' = [R^T | -R^T V]$ benötigt.

$$C'_\delta = -R * (-R^T V) \quad (4.31)$$

Im nächsten Schritt, müssen die Bildebenepunkte von C' noch in das selbe Koordinatensystem wie die Bildebenepunkte von C transformiert werden. Da Kamera eins Deckungsgleich mit dem Weltkoordinatensystem ist, sind die Bildpunkt der Bildebene I von C bereits in Weltkoordinaten gegeben, diese müssen nur noch mit, ζ als dritte Tiefenkomponenten z , erweitert werden. Die Bildpunkte von C' , werden ebenfalls mit ihrem ζ erweitert und danach mit der Projektionsmatrix P , welche als V die Koordinaten von C' besitzt, in das Weltkoordinatensystem überführt. Sind die Basen der Bildpunkte von C_δ und C'_δ , mit $\delta = \beta$ im selben Koordinatensystem, so können nun die Geradengleichungen durch die Projektionszentren C und C' und den entsprechenden Bildebene koordinaten $A_\delta, B_\delta, C_\delta, D_\delta, A2_\delta, B2_\delta, C2_\delta, D2_\delta$ und den neu umgerechneten Punkten $A'_\delta, B'_\delta, C'_\delta, D'_\delta, A2'_\delta, B2'_\delta, C2'_\delta, D2'_\delta$ gezogen. Beispielhaft wird dies Anhand der korrespondierenden Punkte A_τ und dem umgerechneten A'^τ aufgezeigt.

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

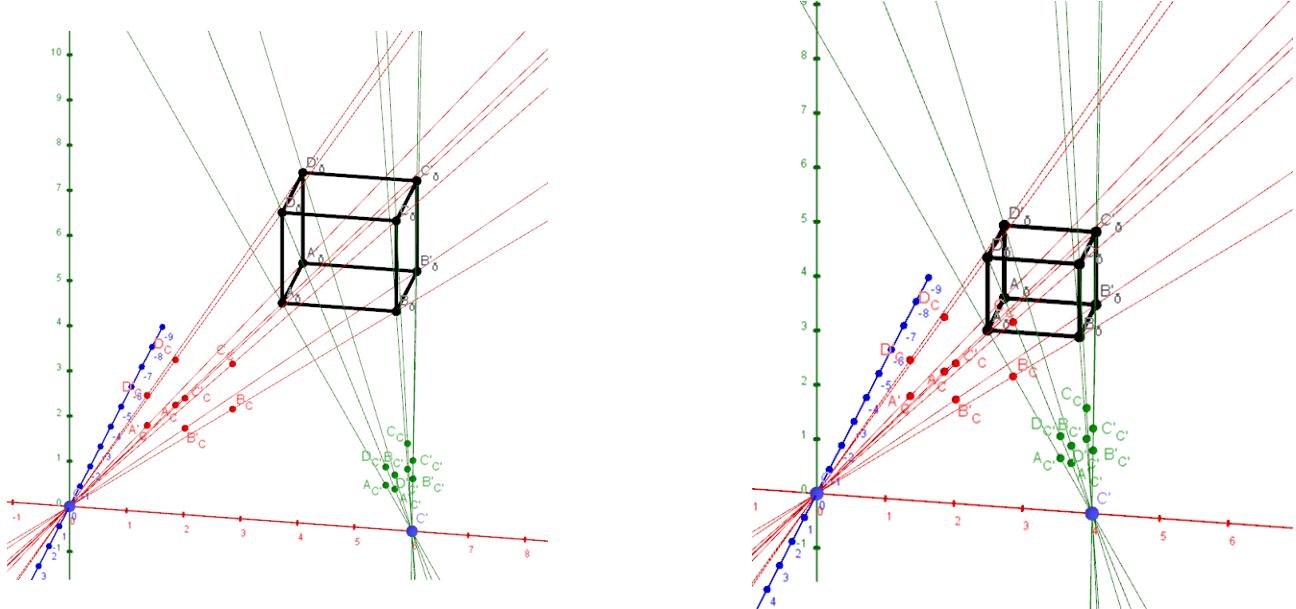
$$A'_\delta + t' * (A'_\delta - C'_\delta) = 0$$

Geraden gleichsetzen:

$$A_{\delta x} - t_x - C_{\delta x} = A'_{\delta x'} - t'_x - C'_{\delta' x}$$

$$A_{\delta y} - t_y - C_{\delta y} = A'_{\delta y'} - t'_y - C'_{\delta' y}$$

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



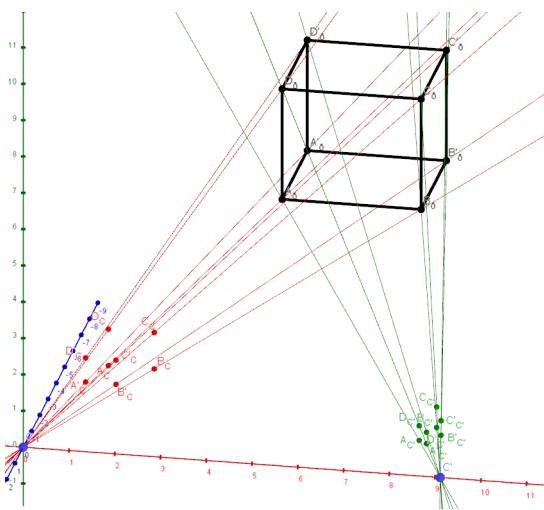


Abbildung 4.8: Veranschaulichung der Skaleninvarianz und dessen Auswirkung auf die geometrische Form der Objekte

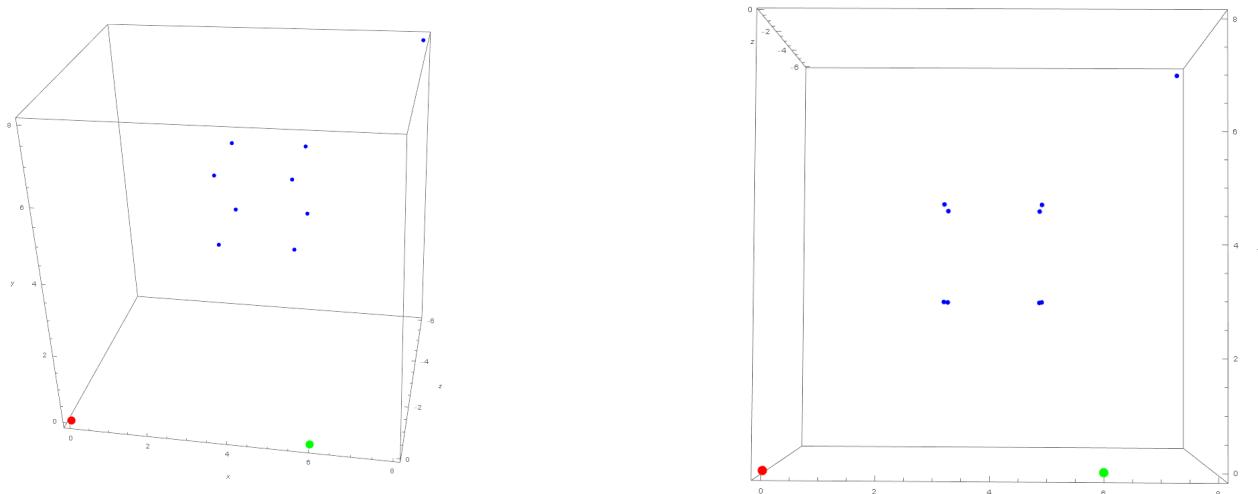


Abbildung 4.9: auf Originalgröße skalierte rekonstruierte Szene

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

5 Auswirkungen von unterschiedlichen Kameraauflösungen

Wenn man sich mit digitalen Bilddaten auseinandersetzt, so kommt man nicht drum herum sich auch mit den verschiedenen Auflösungsarten beschäftigen zu müssen. Im vorherigen Kapitel wurde gezeigt, wie eine 3D-Szene aus zwei heterogenen Bildquellen, welche von zwei Kameras gleicher Auflösung aufgenommen wurden, rekonstruiert werden konnte und gleichzeitig noch die externen Kameraparameter von C' in Relation zu C geschätzt werden konnten. Dies wirft die Frage auf, welche Auswirkungen Bilder zweier Kameras mit unterschiedlichen Auflösungen für die Rekonstruktionen haben können. Aus der Stereokalibrierungsapp, welche *MatLab* anbietet, ist bekannt, dass diese nicht mit Bildern unterschiedlicher Auflösung eine Szeneriekonstruktion durchführen kann. Der erste Schritt bestand erstmal darin zu überprüfen, warum zwei unterschiedliche Auflösungen in *MatLab* Probleme machen. *MatLab* verfolgt einen etwas anderen Rekonstruktionsansatz. Zu aller erst werden die Kameras kalibriert. Dies geschieht über die Matlab-Funktion `estimateCameraParameters`[29]. Diese Funktion funktioniert auch bei Bildern unterschiedlicher Auflösung noch ohne Probleme. Das Problem, welches sich als eigentlich minimales Problem herausstellt, ist, dass die darauf folgenden Rektifizierung der Stereobilder nicht mit zwei Bildern unterschiedlicher Auflösung funktioniert. In den *MatLab* references, steht es nicht explizit drin [19]. Die Rektifizierung in Matlab funktioniert nach einem Schema, welches ähnlich dem aufgezeigten Beispiel im Kapitel Vergleich entwickelter Rekonstruktions Algorithmus mit bereits vorhandenen (Matlab) bereits erwähnt wurde und in [22, 24] nochmal genau aufgeführt wird. Das Problem liegt also nicht daran, dass Bilder unterschiedlicher Auflösung nicht rektifiziert werden können, sondern das Problem liegt an dem in *MatLab* verwendeten Algorithmus für die Rektifizierung zweier Stereobilder (Foren Zitieren????). Warum *MatLab* überhaupt rektifiziert, liegt daran, dass ein Ansatz der Szeneriekonstruktion gewählt wurde, welcher die essentielle Matrix nicht benötigt. In diesem Falle, werden zwei Stereobilder aufgenommen, danach rektifiziert und anschließend über eine sogenannte *Disparity-Map*, die Szenen Punkte rekonstruiert[30, 6, 22, 21]. Der in dieser Arbeit gewählte Rektifizierungsalgorithmus, ist nicht auf gleiche Kameraauflösungen beschränkt. Mittlerweile gibt es natürlich deutlich fortgeschrittenere Rektifizierungsansätze, jedoch wurde für diese Arbeit der Ansatz von [20] gewählt, um ein Gefühl zu vermitteln, dass wenn man sich auf die Epipolargeometrie bezieht, die Auflösungen der Kameras keine Rolle spielen[5]. Was genau unterschiedliche Auflösungen der Kameras für die einzelnen Bilder bedeutet und was genau sich bei der Aufnahme mit dem Sensor dabei ändert, soll im folgenden Unterkapitel kurz erläutert werden. Danach wird aufgezeigt, was genau diese Veränderungen für die Epipolargeometrie bedeuten und letztendlich wird das Minimalbeispiel so angepasst, als hätten zwei Kameras unterschiedlicher Auflösung die selbe Szene wie davor aufgenommen und die Resultate miteinander verglichen.

5.1 Abbildungsunterschiede

Das Herz einer modernen Kamera ist der Halbleiter-Bildsensor. Die Bildsensoren spielen in der optischen Messtechnik und industriellen Bildverarbeitung eine große Rolle[10]. Die in dieser Arbeit verwendeten Kameras für die Aufnahme der Stereobildpaare, waren zum einen die Vollformatkamera Canon EOS 6D und die Halbformatkamera Canon EOS 60D. Beide besitzen einen CMOS-Sensorchip, welcher zu den Halbleiterbildsensoren gehört[31]. Die Geometrie dieser Sensoren, ist grob gesprochen als eine $M \times N$ - Matrix, bestehend aus $M \times N$ Pixeln. Die Pixel bedecken nur einen Teil der Sensorfläche und können in ihrer geometrischen Beschaffenheit von der Form des Sensors unterschieden werden[10]. Abbildung 5.1 zeigt einen schematischen Aufbau eines Halbleiterbildsensors

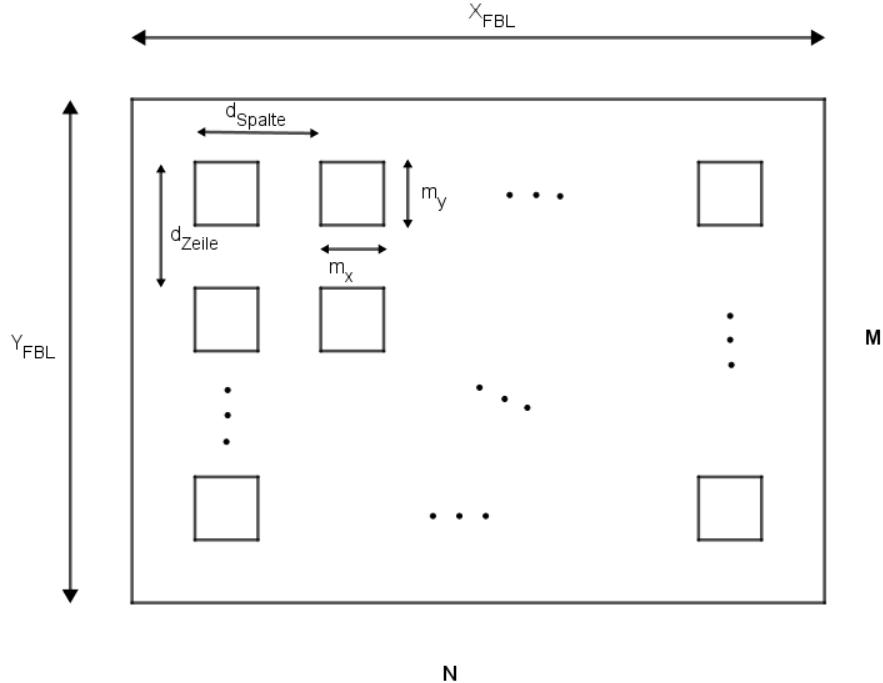


Abbildung 5.1: Rechteckiger Bildsensor mit darauf sich befindenden quadratischen Pixeln. Vlg [10]

Die Auflösung des Sensors hängt von den horizontalen und vertikalen Pixelabständen ab und gibt wieder, welche Objektdetails mit dem Sensor gerade noch erkannt werden können.[10]. Ein Sensor hat eine maximale Auflösung, welche durch die Anzahl seiner fest installierten Pixel bestimmt wird. Die Bildqualität ist abhängig von der Größe des Sensorchips und der Menge der sich darauf befindenden Pixel. Der CMOS-Sensor einer Canon 6D hat beispielsweise eine Größe von $36 \times 24\text{mm}$ und eine maximale Auflösung von 5.472×3.648 Pixel. Jedoch ist das nicht die einzige Auflösung, welche beim photographieren oder filmen mit der Kamera genutzt werden kann. Es können sowohl die Pixelanzahl als auch das Seitenverhältnis der entstehenden Bilder eingestellt werden. Bei der Canon EOS 6D können insgesamt vier verschiedene Seinsverhältnisse eingestellt werden [3 : 2], [4 : 3], [16 : 9] und [1 : 1][31]. Die Bildauflösungen unterscheiden sich pro Seitenverhältnis in sechs Auflösungseinstellungen L , M , $S1$, $S2$, $S3$.

Tabelle 5.1: Auflösungen Canon EOS 6D

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

Tabelle 5.2: Vgl [31]

Je geringer die Auflösung, desto geringer ist die Anzahl der Pixel. Die Anzahl der Pixel auf einem Sensorchip kann natürlich nicht variieren. Eine geringere Pixelanzahl bei gleichbleibender Bildgröße, bedeutet, dass sich ein Pixel mit den um sich befindenden Pixeln interpoliert, so dass ein neuer Pixel bestehend aus mehreren kleinen Pixeln entsteht. Dieser Prozess wird Nachbarschaftsoperation genannt. Für die Berechnung des neuen Bildpixels px' an der Stelle (m, n) wird nicht nur das Bildpixel p des Originalbildes an der Stelle (m, n) verwendet, sondern auch einige seiner Nachbarpunkte[10]. Bei

der Canon 6D bietet das Seitenverhältnis [3 : 2] die Möglichkeit die maximale Pixelanzahl des Sensors zu verwenden, vergleiche hierzu Tabelle 5.1. Bei einem Seitenverhältnis von [4 : 3] ist die Anzahl der maximal möglichen Pixel nur noch 4864×3648 . Anders sich das Seitenverhältnis des Bildausschnitts, so wird auch nicht mehr die gesamte Fläche des Sensors benutzt.

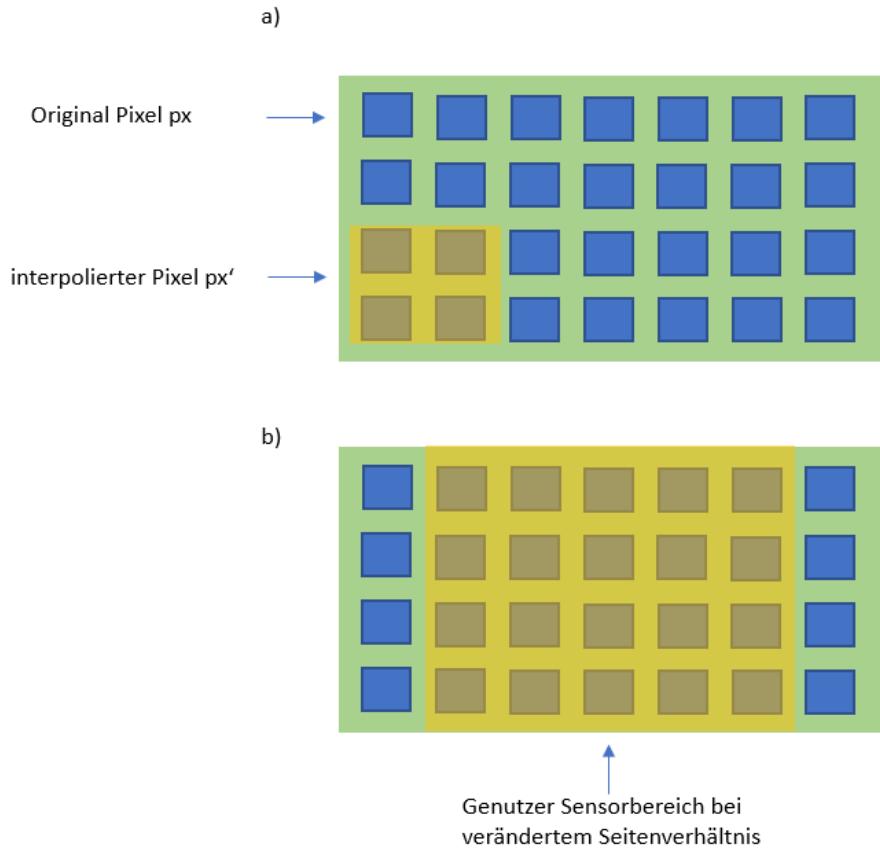


Abbildung 5.2: Bild a) zeigt die Interpolation von Pixeln, wenn bei gleichbleibenden Seitenverhältnissen weniger Pixel für das Bild verwendet werden sollen. Die interpolierten Pixel leiten dann alle das selbe Signal weiter. Bild b) zeigt in gelb markiert, den verwendeten Bereich des Sensors, wenn sich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

5.2 Auswirkungen auf die Epipolargeometrie

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

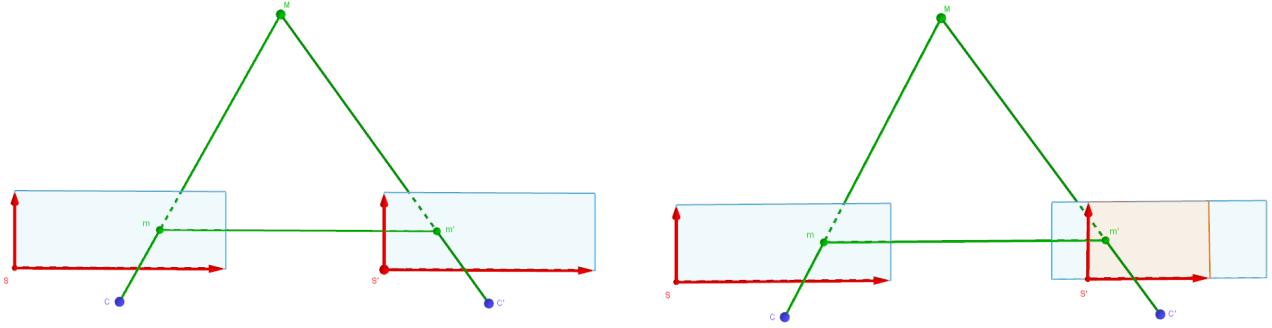


Abbildung 5.3: C und C' haben die selbe Auflösung
Abbildung 5.4: C und C' haben unterschiedliche
eingestellt Auflösungen eingestellt

Es bildet sich wieder das bekannte Dreieck zwischen den Bildpunkten m_σ und m'_σ und dem Objektpunkt M_δ . Das in diesem Falle einen korrekten Szenenrekonstruktion funktioniert, ist im Kapitel Virtuelle Rekonstruktion anhand des Minimalbeispiels aufgezeigt worden. Wird die Auflösung auf einer der beiden Kameras verringert und damit einhergehend auch noch das Seitenverhältnis geändert, so ändert sich zum einen der aktive Teilbereich des Sensorschips, sowie die Skalierung der Werte auf den Koordinatenachsen des Sensorskoordinatensystems. Die Skalierung der Koordinatenachsen hängt mit der Interpolation der mehrerer Pixel zu einem neuen Pixel zusammen. Abbildung 5.3 zeigt schematisch, was sich nach veränderten Auflösungseinstellungen am Sensor verändert. Epipolare geometrisch ändert sich wie man in Abbildung ??? sehen kann nichts. Das zuvor erwähnte Dreieck zwischen den Bildpunkten und dem Objektpunkt bleibt unverändert. Wie in Kapitel geometrische Erläuterung der Epipolare Geometrie bereits erwähnt, dürfen die Bildkoordinatensysteme und somit auch die Sensorskoordinatensysteme unterschiedlich voneinander sein [5]. Für die relative Position des Bildpunktes auf dem Sensor ändert sich nichts, dieser bleibt statisch, einzige seine Koordinatenwerte ändern sich im Bezug auf das Sensorskoordinatensystem. Für die Fundamentalmatrix und die Essentielle Matrix ergeben sich lediglich andere Vielfache voneinander, welche wie erwähnt ebenfalls gültige Lösungen sind [3, 4].

5.3 Minimalbeispiel mit unterschiedlichen Kameraauflösungen

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

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

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

$$(5.3)$$

$$K'_2 = \begin{bmatrix} -3.2 & 0 & 0 & 0 \\ 0 & -1.2 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (5.4)$$

$$K'_3 = \begin{bmatrix} -0.5 & 0 & 0 & 0 \\ 0 & -4.3 & 0 & 0 \\ 0 & 0 & -1 & 0 \end{bmatrix} \quad (5.5)$$

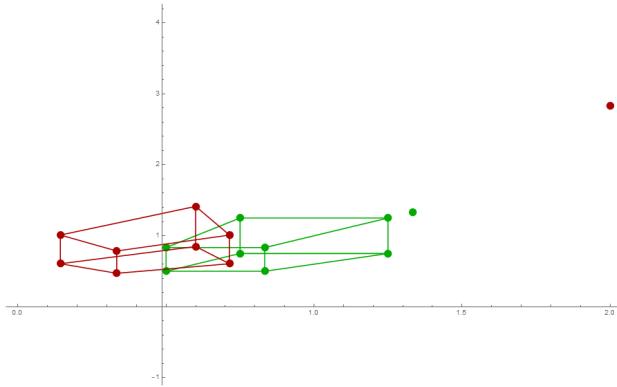


Abbildung 5.5: C und C' haben die selbe Auflösung eingestellt

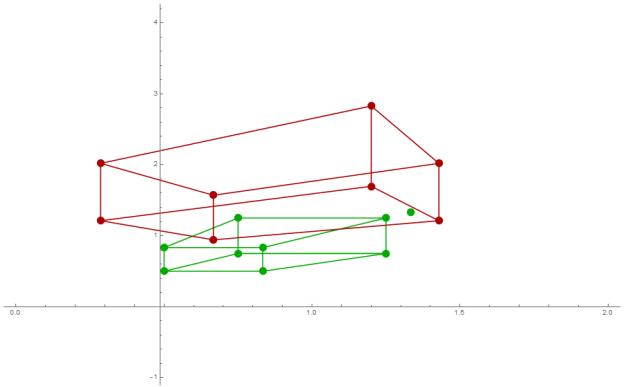


Abbildung 5.6: C und C' haben unterschiedliche Auflösungen eingestellt

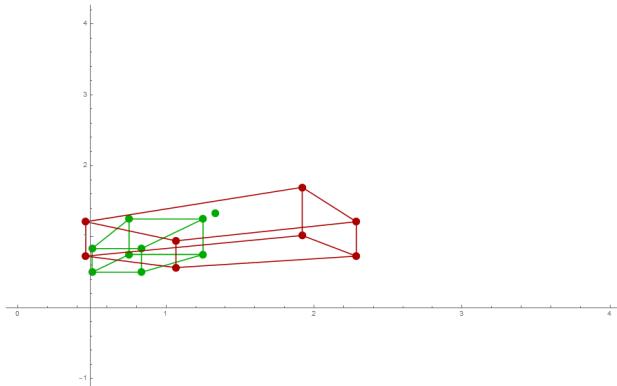


Abbildung 5.7: C und C' haben die selbe Auflösung eingestellt

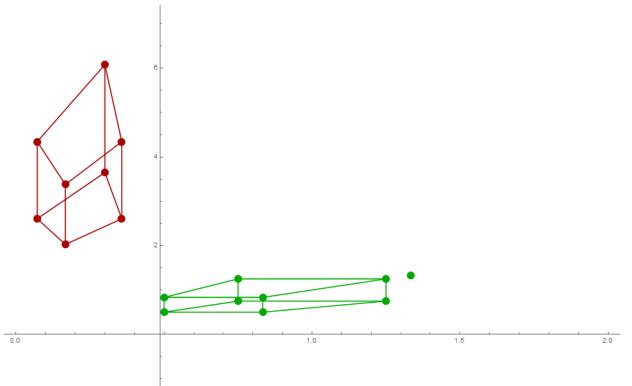


Abbildung 5.8: C und C' haben unterschiedliche Auflösungen eingestellt

Während die Abbildung von C Unverändert bleibt, wird in Abbildung ??? Die Abbildung des Quaders in C' "vergrößert", was für eine höhere Anzahl an verwendeten Pixeln steht. In Abbildung ??? werden die Pixel in horizontaler Richtung um das 3.2-fache und in vertikaler Richtung um das 1.2-fache erweitert und in Abbildung ??? wird in horizontaler Richtung die Anzahl der Pixel um das 0.5-fache und in vertikaler Richtung um das 4.3-fache skaliert. Für die Fundamentalmatrix und die essentielle Matrix ergeben sich verglichen mit denen aus dem Beispiel mit gleicher Abbildung folgende Matrizen.

$$\zeta_x = 1, \zeta_y = 1 : F = \begin{pmatrix} 0 & -0.5 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & -0.5 & 0 \end{pmatrix} | : -0.5 \rightsquigarrow F = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}$$

$$\zeta_x = 2, \zeta_y = 2 : F = \begin{pmatrix} 0 & 0.378 & 0 \\ 0 & 0 & -0.534 \\ 0 & 0.756 & 0 \end{pmatrix} | : 0.756 \rightsquigarrow F = \begin{pmatrix} 0 & 0.5 & 0 \\ 0 & 0 & -\frac{1}{\sqrt{2}} \\ 0 & 1 & 0 \end{pmatrix}$$

$$\frac{1}{\zeta_x} = 0.5, \quad \frac{\sqrt{2}}{\zeta_y} = \frac{1}{\sqrt{2}}$$

$$\zeta_x = 3.2, \zeta_y = 1.2 : F = \begin{pmatrix} 0 & 0.198 & 0 \\ 0 & 0 & -0.747 \\ 0 & 0.634 & 0 \end{pmatrix} | : 0.634 \rightsquigarrow F = \begin{pmatrix} 0 & 0.312 & 0 \\ 0 & 0 & -1.178 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\frac{1}{\zeta_x} = 0.312, \quad \frac{\sqrt{2}}{\zeta_y} = -1.178$$

$$\zeta_x = 0.5, \zeta_y = 4.3 : F = \begin{pmatrix} 0 & 0.885 & 0 \\ 0 & 0 & -0.145 \\ 0 & 0.442 & 0 \end{pmatrix} | : 0.442 \rightsquigarrow F = \begin{pmatrix} 0 & 2 & 0 \\ 0 & 0 & -0.328 \\ 0 & 1 & 0 \end{pmatrix}$$

$$\frac{1}{\zeta_x} = 2, \quad \frac{\sqrt{2}}{\zeta_y} = -0.328$$

(hier können vllt noch die verschiedenen rotationen gezeigt werden ,welche immer die selbe rotation bewirken) Die Werte für ζ_x wirken sich auf die erste Zeile der Fundamentalmatrix aus, während die Werte von ζ_y sich auf die zweite Zeile auswirken. Bei der nachfolgenden Umrechnung der Fundamentalmatrix in die essentielle Matrix mit Hilfe der Kameramatrizen K und K' , kann gestellt werden, dass die Ergebnisse jeweils Vielfache voneinander sind.

$$\zeta_x = 1, \zeta_y = 1 : E = \begin{pmatrix} 0 & -0.5 & 0 \\ 0 & 0 & \frac{1}{\sqrt{2}} \\ 0 & 0.5 & 0 \end{pmatrix} | : 0.5 \rightsquigarrow E = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}$$

$$\zeta_x = 2, \zeta_y = 2 : E = \begin{pmatrix} 0 & 0.756 & 0 \\ 0 & 0 & 1.069 \\ 0 & -0.756 & 0 \end{pmatrix} | : -0.756 \rightsquigarrow E = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}$$

$$\zeta_x = 3.2, \zeta_y = 1.2 : E = \begin{pmatrix} 0 & 0.634 & 0 \\ 0 & 0 & 1.069 \\ 0 & -0.634 & 0 \end{pmatrix} | : -0.634 \rightsquigarrow E = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}$$

$$\zeta_x = 0.5, \zeta_y = 4.3 : E = \begin{pmatrix} 0 & 0.442 & 0 \\ 0 & 0 & 1.069 \\ 0 & -0.442 & 0 \end{pmatrix} | : -0.442 \rightsquigarrow E = \begin{pmatrix} 0 & -1 & 0 \\ 0 & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{pmatrix}$$

Bei der Rekonstruktion der externen Kameraparameter ergibt sich daraus stehts die selbe Matrix für P' . Was wie gezeigt daran liegt, dass sich geometrisch nichts ändert, sondern lediglich die Skalierung der Koordinatenwerte der Bildpunkte und somit auch eine Skalierung der Einträge in F und E , welche aber ebenfalls als Skaleninvariant definiert sind[3]. Die Ergebnisse der darauffolgenden Szenenrekonstruktionen, der einzelnen Szenen zeigt, dass sich immer die selbe Szene ergibt, welche mit der eigens aufgebauten Szene übereinstimmen.

Reconstructed scaled Points 3D =

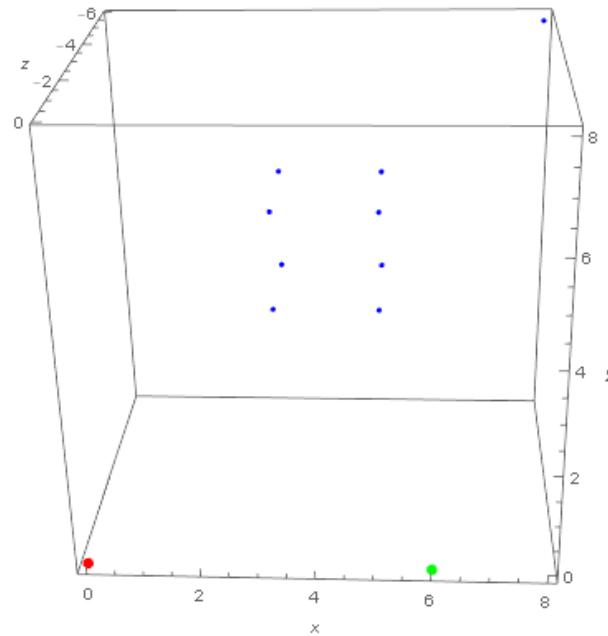
$$\begin{pmatrix} 3. & 3. & -4. \\ 5. & 3. & -4. \\ 5. & 5. & -4. \\ 3. & 5. & -4. \\ 3. & 3. & -6. \\ 5. & 3. & -6. \\ 5. & 5. & -6. \\ 3. & 5. & -6. \\ 8. & 8. & -6. \end{pmatrix}$$


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

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

(HIER VLLT NOCH ZEIGEN, DAS DER REKTIFIZIERUNGSANSATZ AUCH MIT KAMERA UNTERSCHIEDLICHER AUFLÖSUNG FUNKTIONIERT)

6 Relle Rekonstruktion

Durch im Minimabeispiel durchgeführten Stereoanalyse steht das Grundgerüst für die Stereoanalyse mit realen Bilddaten. Der selbe Arbeitsprozess soll nun auch auf ein Realbeispiel mit Bildern zweier Kameras durchgeführt werden. Die externen Kameraparameter, werden in extern beschaffen. Über *Matlab* wird für jede Kamera einzeln mit Hilfe der *Single-camera-aclibration* App eine Kalibrierung durchgeführt. Da diese auch die externen Parameter liefert, können diese später als Vergleich mit den eigens berechneten Parameter dienen. Im fortschreitenden Arbeitsprozess der Stereoanalyse, werden des öfteren Ungereimtheiten im Vergleich mit dem Minimalbeispiel auftreten, welche behoben werden müssen um weiter zu verfahren. Diese Fehler entstehen durch die Ungenauigkeit, der zuvor detektierten korrespondierenden Punkte oder durch Bildfehler wie Rauschen oder Unschärfe, welche in der Photographie nicht immer vermieden werden können. Um diese Fehler zu beheben, werden einige Zwischenschritte benötigt, welche im Minimalbeispiel durch die Reinheit der Daten und der selbst aufgebauten Szenen nicht nötig waren. Im Minimalbeispiel wurden einige solcher möglichen Fehler bereits erwähnt. Für die Stereoaufnahmen wurden die halbformat Kamera Canon 60 D und die Vollformatkamera 6D verwendet. Die Bilder wurden mit beiden Kameras mit selber Auflösung und Seitenverhältnis aufgenommen. Später wurden auch noch Aufnahmen mit unterschiedlichen Auflösungen gemacht und ebenfalls die Stereoanalyse auf die Bildpaare angewandt.

6.1 Arbeitsprozess

Die beiden Kameras wurden so im Raum platziert, dass sie leicht zueinander hin gedreht waren. Da die Canon 60D eine halbformatkamera ist, wurde sie weiter hinten als die Canon 6D positioniert. Somit konnte ungefähr der selbe Bildausschnitt der Szene mit beiden Kameras aufgenommen werden.

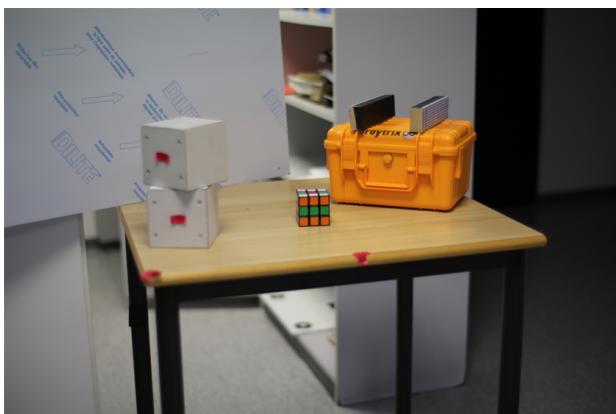


Abbildung 6.1: Aufnahme der Canon 6D von links

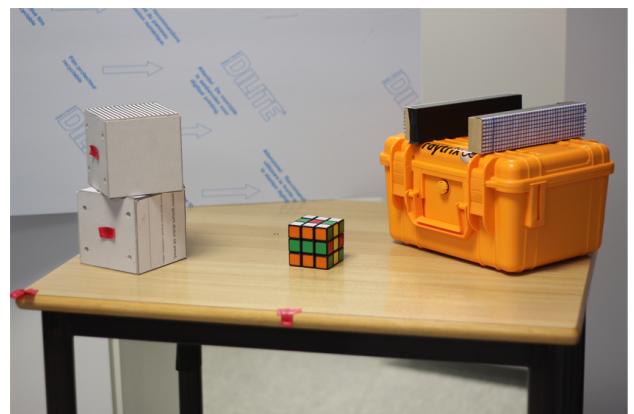


Abbildung 6.2: Aufnahme der Canon 60D von rechts



Abbildung 6.3: 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 Kamerassen ist die in den Abbildungen 6.1 und 6.1 abgebildete Szene zu sehen. Die Canon 60D ist etwas hinter der Canon 6D positioniert. Beide Kamerassen sind zu Szene hin gedreht und auch leicht nach unten geneigt.

Die Canon 6D wird als primär Kamera definiert und bekommt somit die bezeichnung C , während die Canon 60D ab jetzt mit C' gekennzeichnet wird. Die räumliche Orientierung und Position von C' wird also relativ zu C berechnet und auch die Szene wird davon ausgehend, dass C als Projektionsmatrix $P = [I|0]$ besitzt.

$$P = (I|0) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \quad (6.1)$$

Die Koordinatensysteme von C mit (C, β) und das Weltkoordinatensystem (O, δ) werden deckungsgleich definiert. Das Koordinatensystem von C' wird mit (C', β') definiert. Für die spätere Skalierung der rekonstruierten Szene ist es ratsam innerhalb der Szene einen Abstand zweier Punkte zueinander abzumessen, um einen Referenzwert für die Skalierung zu bekommen. Der Stereoaufbau ist somit fertig installiert. Der nächste Schritt ist mit Hilfe von *Matlab* und dem mitgelieferten Kalibrierungs-Tool namens *Single-camera-calibration* für jede Kamera eine Kalibrierung durchzuführen, um so die intrinsischen Parameter K und K' zu bekommen. Diese werden für den verwendeten Ansatz für die Schätzung der extrinsischen Parameter benötigt. Nach der Kalibrierung werden Stereoaufnahmen der Szene gemacht und anschließend korrespondierende Bildpunkte aus den beiden Bildern gesucht. Für den ersten Versuch wurden diese von Hand ausgelesen, da dies jedoch sehr Zeitaufwändig ist, wurden andere Möglichkeiten zur Detektion von korrespondierenden Bildpunkten überlegt. Wie in Abbildung (Einleitung Workflow) ersichtlich ist, wurde zwei Ansätze verfolgt. Fürs erste, wurde überlegt, ähnlich wie in *Matlab* anhand von 2D Schachbrettern eine Stereoanalyse durchzuführen.[6]. Ein 2D Schachbrett wird von beiden Kamerassen positioniert und es werden Stereoaufnahmen des selbigen gemacht. Mit Hilfe eines Algorithmus, welcher in einer vorherigen Arbeit angefertigt wurde, können die Eckpunkte des Schachbretts der jeweiligen Bilder bestimmt werden. Wichtig für die Funktion des Algorithmus ist, dass die Schachbretter vor einem einfarbigen Hintergrund aufgenommen wurden, so dass sich die Eckpunkt Bestimmung auch nur auf das Schachbrett bezieht und keine weiteren Punkte außerhalb mit in die entstehende Punkteliste aufgenommen werden. Wenn die Koordinaten der Eckpunkte des

Schachbrettmusters bekannt sind, folgt ein weiterer Algorithmus, welcher im Zuge dieser Arbeit implementiert wurde. Dieser Algorithmus nimmt die Liste mit den Koordinaten der Eckpunkte entgegen und sortiert und nummeriert diese Zeilen- und Spaltenweise durch. Jeder Punkt ist somit über zwei Indizes codiert und enthält die Information, in welcher Zeile und in welcher Spalte des Schachbrettmusters er sich befindet. Dieser Algorithmus wird auf beiden Schachbrettern angewandt. Nach der Sortierung der Punkte auf beiden Bildern, ist es möglich, korrespondierende Punkte der Bilder anhand gleicher Indizes der Eckpunkte zu bestimmen. Da nicht immer garantiert ist, dass alle Punkte innerhalb des Schachbretts zuvor gefunden worden, enthält der Sortierungsalgorithmus eine Funktion, in welchem er Lücken des innerhalb ausfindig macht und synthetische Eckpunkte setzt. Diese synthetisch gesetzten Punkte, werden markiert, so dass sie nicht in die Liste der möglichen korrespondierenden Punkte fallen. Wie genau der Sortierungsalgorithmus implementiert wurde wird im Kapitel Punktesortierung in Schachbrettmustern genauer beschrieben. Danach wird wie in Abbildung ???(Einleitung) gezeigt weiter verfahren. Um eine 3D Szene rekonstruieren zu können, wurde mit Hilfe eines in *Mathematica* bereits implementierten Verfahren zu Findung korrespondierender Punkte in zwei Bildern genutzt. Die benutzte Funktion basiert auf dem sogenannten *SURF*-Algorithmus. *SURF* steht für *Speeded Up Robust Features* und ist ein Rotations- und Skaleninvarianter Punkte Detektor und Deskriptor[32]. Die Suche nach diskreten Bildkorrespondenzen kann in drei Schritte eingeteilt werden. Im ersten Schritt werde sogennaten *Point of interest* an markanten Stellen im Bild detektiert. Darunter fallen zum Beispiel Eckpunkte-Erkennung, "Blob"- Erkennung oder "T-Junctions"- Erkennung[32]. Diese Aufgabe wird dem Detektor zugeordnet. Der wohl am meisten genutzte Detektor in heutigen Computer Vision Applikationen ist der sogenannte *Harris corner detector*[32]. Die Umgebung eines jeden gefundenen Punktes wird durch einen Merkmalsvektor beschrieben, den Deskriptor[32]. Dieser Deskriptor muss unverwechselbar und zu gleichen Zeit robust gegenüber Bildrauschen, Detektionsfehlern und geometrischen Deformationen sein. Im letzten Schritt müssen die Deskriptoren zwischen den Bildern abgestimmt werden. Meistens wird dieses *matching* über die Distanzen der Vektoren betrieben. In Abbildung 6.4 ist eines der Ergebnisse des verwendeten *SURF*-Algorithmus zu sehen. Das linke Bild wurde mit der Canon 6D aufgenommen, das rechte mit der Canon 60D. Die gelben Ziffern in den Bildern markieren die jeweiligen korrespondierenden Punkte in den Bildern.

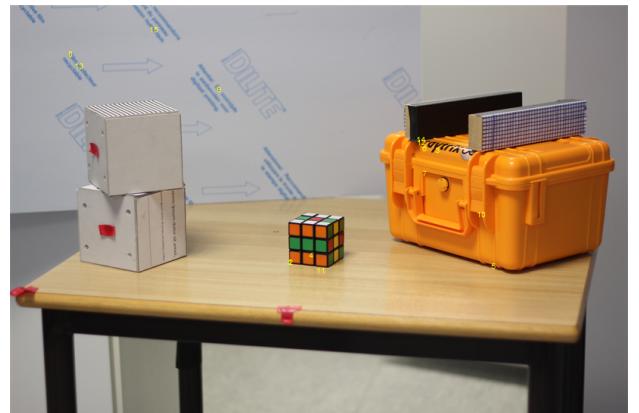
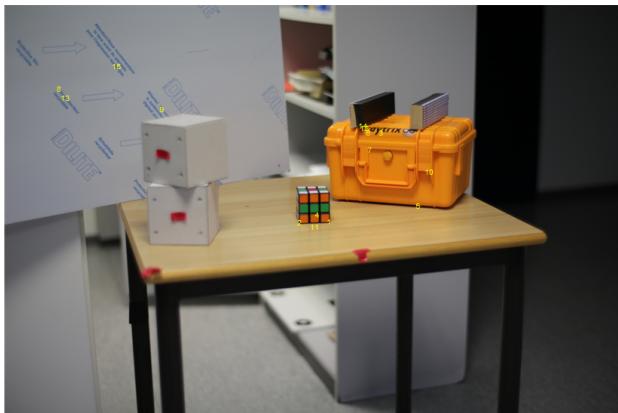


Abbildung 6.4: Die mit dem *SURF*-Algorithmus gefundenen Punkte sind mit den gelben Ziffern im Bild gekennzeichnet

6.2 Normalized-eight-Point-Algorithm

Nachdem die korrespondierenden Punkte in den Bilder gefunden wurden, wird nun der Arbeitsprozess wie in den Abbildungen ??? und ??? in der Einleitung und wie bereits aus dem Minimalbespiel bekannt, weiterverfolgt. In den einzelnen Schritten müssen jedoch ein paar Änderungen vorgenommen werden, um die durch die ungenauen Bilddaten entstehenden Fehler im Verlauf des Arbeitsprozesses zu minimieren.

Als erstes erfolgt die Schätzung der Fundamentalmatrix. Für die Schätzung wurde eine leicht abgeänderte Form des *eight-point-algorithms* namens *normalized-eight-point-algorithm* angewandt[3, 17, 4]. Zur Durchführung des *normalized-eight-point-algorithm* wird eine vorherige Normierung der eingehenden Bildpunkte pro Bild verlangt. Diese sollen so normiert werden, dass ihr durchschnittlicher Abstand zu ihrem den Koordinatenursprung verschobenenen Schwerpunkt $\sqrt{2}$ beträgt[3, 17, 4]. Zu aller erst werden die jeweiligen Schwerpunkte der Punkte in den einzelnen Bildern gesucht und dieser dann in den jeweiligen Sensorskoordinatenursprung verschoben. Die Bildpunkte werden unter Beibehaltung ihres momentanen Abstandes zum Schwerpunkt mit verschoben. Danach werden die Anständer der Bildpunkten zum Schwerpunkt so skaliert, dass der Durchschnittsabstand der Punkte zu Schwerpunkt $\sqrt{2}$ beträgt. Die so skalierten Bildpunkte befinden sich nun in einem deutlich kleineren Zahlenbereich von circa -1 bis 1[3, 17, 4]. Die Transformation der Bilddpunkte für beide Bilder wird jeweils einer Matrix T und T' vollzogen. Diese Matrix ist wichtig, um nach dem schätzen einer auf den normalisierten Koordinaten basierten Fundamentalmatrix \hat{F} wieder eine denormalisierte F zu generieren. Die Normierung der Bildkoordinaten ist wichtig, um die Auswirkung der Bildfehler auf das Endergebnis zu minimieren. Die Entscheidung den normalized-8-Point-Algorithm zu benutzen fiel als festgestellt wurde, dass ohne vorherige Normalisierung der ausgelesenen Punkte es zu größeren Fehlern in den weiteren Berechnungen kam. Zur Erklärung dieser Fehler kann zum einen die *Condition-Number* betrachten. Als *Condition Number*, Kondition im deutschen, wird die Abhängigkeit der Lösung eines Problems von der Störung der Eingangsdaten beschrieben. 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 einer 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 [17]. Wird angenommen, dass AA^T eine 9×9 -Matrix ist, so ergeben die Einträge d_1/d_9 die gesuchte *Condition Number*. Je größer die *Condition-Number* ist, desto größer wirken sich auch kleinste Abweichungen, wie Bildrauschen, auf die Resultate aus. Da sich die original Bildkoordinaten in diesem Beispiel in einem Zahlenbereich von 0 bis 5478 befinden, sind auch die Werte innerhalb der Koeffizientenmatrix in einem sehr großen Zahlenbereich, was zu Folge hat, dass schon kleinste Abweichungen in den Bilddaten, große Auswirkungen auf die daraus resultierende Fundamentalmatrix haben kann, in Bezug darauf, dass die Werte der Einträge innerhalb von F , sehr große Ungleichgewichte aufweisen. Anders im Fall von Normierten Koordinaten, deren Zahlen sich in einem Bereich zwischen circa -1 und 1 befinden.

$$F = \begin{pmatrix} 10^{-4} & 10^{-4} & 10^{-2} \\ 10^{-4} & 10^{-4} & 10^{-2} \\ 10^{-2} & 10^{-2} & 1 \end{pmatrix} \quad (6.2)$$

$$F = \begin{pmatrix} -10^{-9} & 10^{-6} & -10^{-4} \\ -10^{-7} & 10^{-4} & 10^{-3} \\ 10^{-4} & -10^{-3} & -10^{-2} \end{pmatrix} \quad (6.3)$$

Gleichung 6.2 zeigt schematisch was unter einer Gleichgewichtigen Fundamentalmatrix zu verstehen ist, welche bei einer sehr geringen *Condition-number* resultieren kann. Gleichung 6.3 wiederum zeigt schematisch das Resultat einer Ungleichgewichteten Fundamentalmatrix, dessen *Condition-Number* sehr groß ausfällt[17]. Durch normieren der Bildkoordinaten, kann die *Condition-Number* kleiner und damit einhergehend die entstehenden Fehler minimiert werden. Nachdem die Fundamentalmatrix aus den normierten Koordinaten geschätzt wurde, wird sie anschließend mit den beiden aufgestellten Matrizen T und T' wieder denormalisiert, so dass sie wieder als *epipolar – constraint* zwischen die original Koordinaten geschalten werden kann. Für normierte Koordinaten \hat{m} und \hat{m}' gilt $\hat{m}'^T \cdot \hat{F} \cdot \hat{m} = 0$ und für die ursprünglichen Bildkoordinaten gilt, dass $m'^T \cdot T'^T \cdot \hat{F} \cdot T \cdot m = 0$ und somit wieder $m'^T \cdot F \cdot m = 0$ [3, 17, 4]. Die Normierung der Koordinaten für die Verwendung des *eight-point-algorithms* darf auf keinen Fall mit der Normierung der Koordinaten für die essentielle Matrix E verglichen werden. Die Normierung der Koordinaten für die Schätzung von F , soll die Auswirkungen von Fehler auf die Resultate minimieren, während die Normierung der Koordinaten durch deren Verrechnung mit den

Kameramatrizen K und K' dafür sorgt, dass daraus normierte Bildkoordinaten entstehen, dessen Koordinatenursprung nicht mehr in einer Bildecke sondern in der Bildmitte sich befindet[3]. Um zurück zum Arbeitsprozess zu kommen, sind die Koordinaten normiert, so wird der im Kapitel Einleitung aufgezeigte Verfahren zur Schätzung der Fundamentalmatrix gleichermaßen wie in den Gleichungen 4.29 bis 4.34 aufgebaut. Durch die möglichen Ungenauigkeiten wie Bildrauschen oder dem detektieren der korrespondierenden Punkte, ist der Rang der aufgestellten Koeffizientenmatrix A in den meisten Fällen größer als acht, was bedeutet, dass hier nicht einfach der Kern mit $A \cdot f = 0$ gesucht werden kann, um eine Lösung zu finden. Im Falle eines höhren Ranges als 8 muss ein Verfahren, ähnlich wie dem, welches angewandt wurde um überbestimmte Systeme zu Lösen um eine Homographiematrix zu erhalten. Es wird also derjenige Vektor für f gesucht, welcher $\|A \cdot f\|$ minimiert. Hierzu wird eine Singulärwertszerlegung von A in $A = UDV^T$ durchgeführt. die Lösung für f , welche $\|A \cdot f\|$ minimiert ist dann diejenige Spalte von V^T , welche mit dem kleinsten Singulärwert von D korrespondiert. Da die Singulärwerte eine absteigende Reihenfolge besitzen, bildet die letzte Spalte von V^T den Vektor f [3].

6.2.1 Singularity-Constraint der Fundamentalmatrix

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

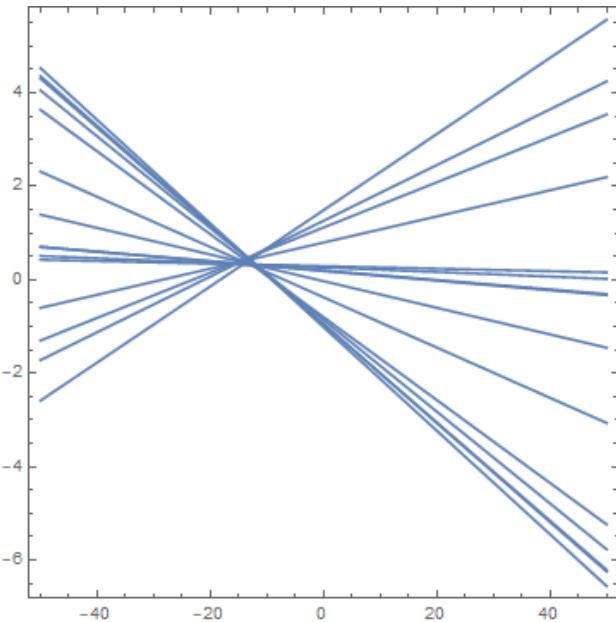


Abbildung 6.5: Epipolarlinien ohne *Epipolar-constraint* im Bild der Canon 6D

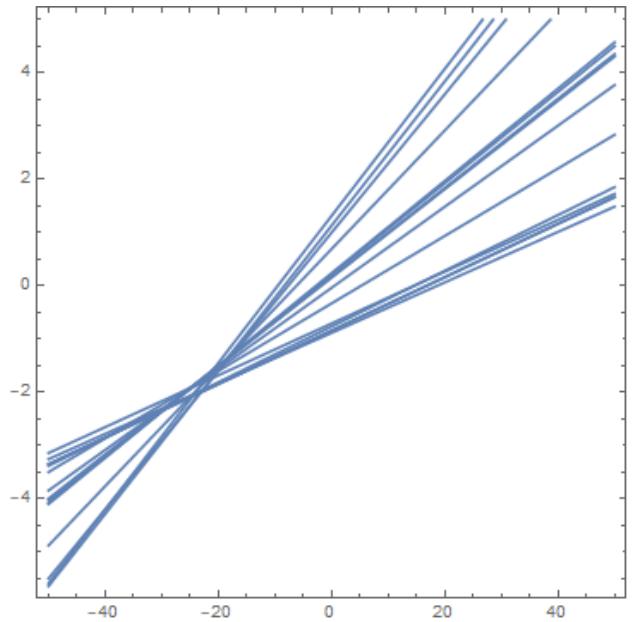


Abbildung 6.6: Epipolarlinien ohne *Epipolar-constraint* im Bild der Canon 60D

Um eine gültige Fundamentalmatrix für den weiteren Arbeitsprozess zu generieren, kommt hier ein sogenannter *singularity constraint* zum Einsatz. Dieser erzwingt in \hat{F} eine Singularität. Zu aller erst wird eine Singulärwertszerlegung an F durchgeführt, so dass \hat{F} in $\hat{F} = UDV^T$ zerlegt wird. D beinhaltet in einer diagonalen Matrix die Singulärwerte $D = \text{diag}(r, s, t)$, welche die Bedingung $r \leq s \leq t$ erfüllen. Um nun den *singularity-constraint* in \hat{F} zu erzwingen, wird der letzte Singulärwert $t = 0$

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

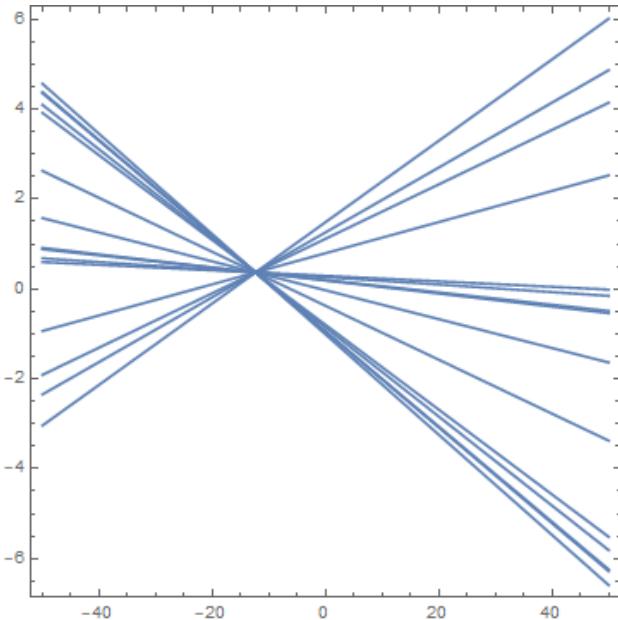


Abbildung 6.7: Epipolarlinien mit *Epipolar-constraint* im Bild der Canon 6D

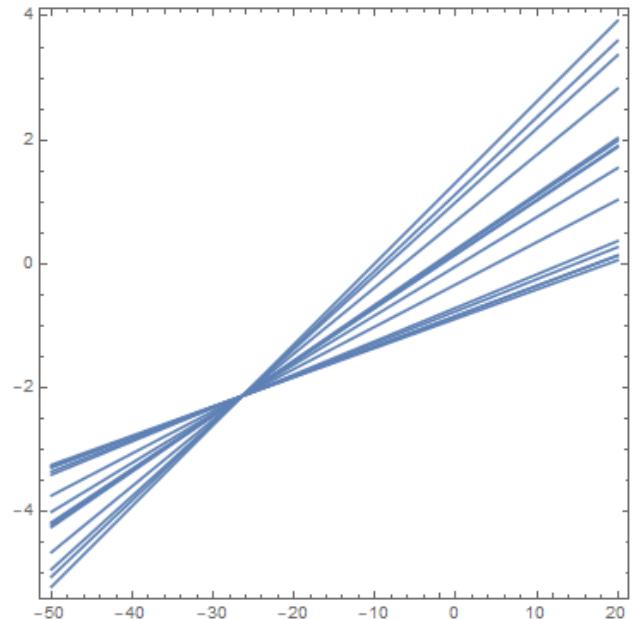


Abbildung 6.8: Epipolarlinien ohne *Epipolar-constraint* im Bild der Canon 60D

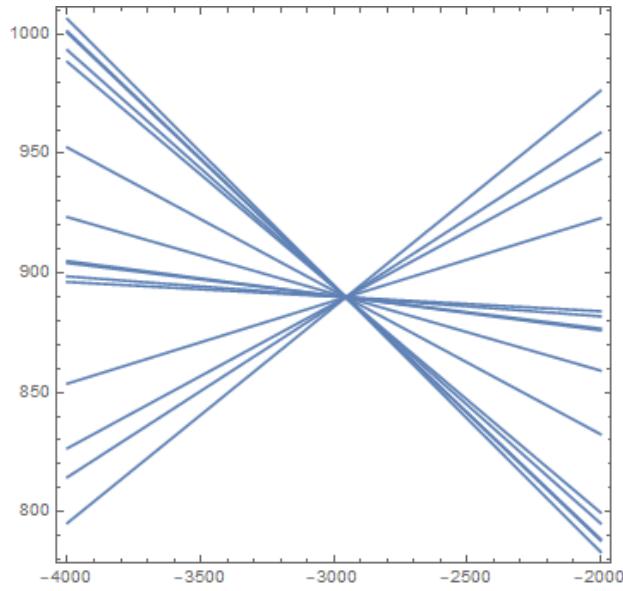


Abbildung 6.9: Epipolarlinien mit *Epipolar-constraint* im Bild der Canon 6D, nach der denormalisierung von F

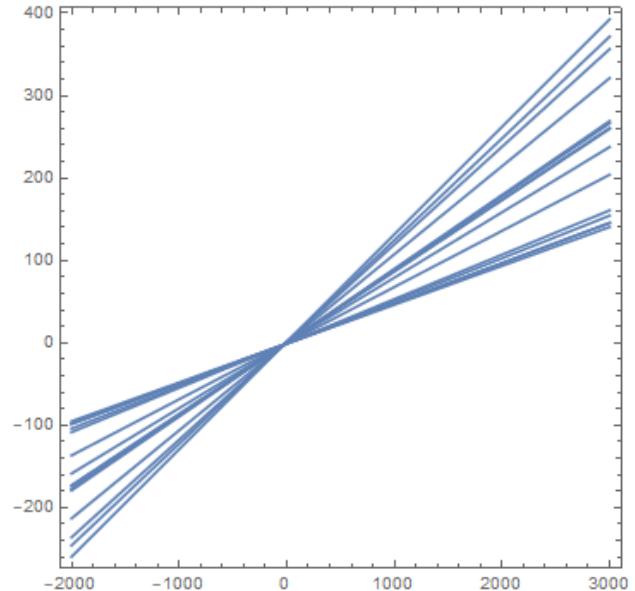


Abbildung 6.10: Epipolarlinien ohne *Epipolar-constraint* im Bild der Canon 60D, nach der denormalisierung von F

6.2.2 Singularity- Constraint der essentiellen Matrix

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

6.3 Szenenrekonstruktion

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

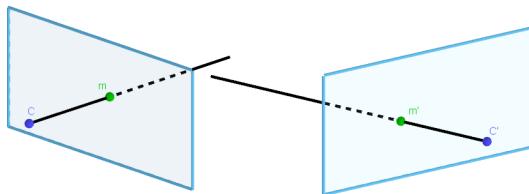


Abbildung 6.11: a)

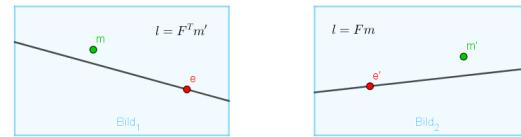


Abbildung 6.12: b)

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

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

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

6.3.1 Minimieren der Kostenfunktion durch Sampson-approximation

Es sollen zwei Punkte \hat{m} und \hat{m}' gefunden werden, welche nahe an den Ursprünglichen m und m' liegen und gleichzeitig den *Epipolar-Constraint* erfüllen. \hat{m} und \hat{m}' sollen durch Minimierung einer Kostenfunktion C ermittelt werden, welche die Distanz d zwischen m und \hat{m} und m' und \hat{m}' minimiert.

$$C(m, m') = d(m, \hat{m})^2 + d(m', \hat{m}')^2 \quad (6.4)$$

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

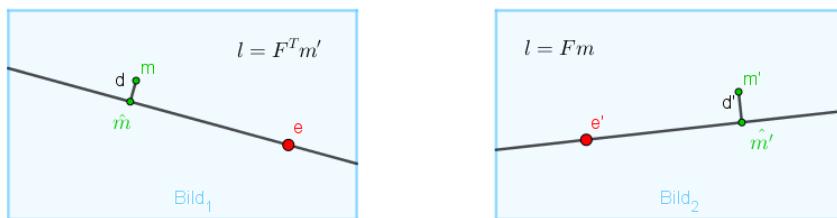


Abbildung 6.14: Frafische Darstellung der optimalen Punkte \hat{m} und \hat{m}'

Jedes Punktpaar auf l und l' würde den *Epipolar-Constraint* erfüllen, jedoch minimieren nur x_{\perp} und x'_{\perp} die quadratischen Distanzen in der Kostenfunktion C . Ausgehend von dieser Aussagen wird C so umformuliert, dass gilt $d(m, \hat{m}) = d(\hat{m}, l)$ und $d(m', \hat{m}') = d(\hat{m}', l')$, was jeweils den senkrechten Abstand m zu l und m' zu l' beschreibt. Werden l und l' frei aus allen möglichen Epipolarlinien gewählt, so wird immer der senkrechte Abstand von x zu dieser gewählten l berechnet, entsprechend gilt das auch für m' und irgendeine l' . Nun muss der Abstand $d(\hat{m}, l)^2 + d(\hat{m}', l')$ minimiert werden, da natürlich nicht einfach jede beliebigen Epipolarlinien genutzt werden können. Es wird eine Strategie mit insgesamt vier Schritten für die Minimierung verfolgt[3]. Zuerst werden die Epipolarlinienbündel

pro Bild so Parametrisiert, dass beispielsweise eine Epipolarlinie im ersten Bild als $l(t)$ geschrieben werden kann. Danach wird die Fundamentalmatrix F dazu benutzt, die entsprechend korrespondierende Epipolarlinie l' zu berechnen. Die Kostenfunktion C kann somit als eine Funktion von t definiert werden. Schlussendlich muss ein Wert für t gefunden werden, welcher C minimal werden lässt.

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

$$\rightsquigarrow C(m, m') = d(m, l(t))^2 + d(m', l'(t))^2 \quad (6.6)$$

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

$$T = \begin{bmatrix} 1 & 0 & -x \\ 0 & 1 & -y \\ 0 & 0 & 1 \end{bmatrix} \rightsquigarrow \bar{m} = T \cdot m \quad (6.7)$$

$$T' = \begin{bmatrix} 1 & 0 & -x' \\ 0 & 1 & -y' \\ 0 & 0 & 1 \end{bmatrix} \rightsquigarrow \bar{m}' = T' \cdot m' \quad (6.8)$$

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

$$\bar{F} = T'^{-T} F T^{-1} \quad (6.9)$$

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

$$\bar{F} = \begin{pmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{pmatrix} \quad (6.10)$$

$$\begin{pmatrix} ff'd & -f'c & -f'd \\ -fb & a & b \\ -fd & c & d \end{pmatrix} \cdot \begin{pmatrix} 1 \\ 0 \\ f \end{pmatrix} = \begin{pmatrix} ff'd + (-ff'd) \\ -fb + fb \\ -fd + fd \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \end{pmatrix} \quad (6.11)$$

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

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

$$R = \begin{bmatrix} e_1 & e_2 & 0 \\ -e_2 & e_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.13)$$

$$R' = \begin{bmatrix} e'_1 & e'_2 & 0 \\ -e'_2 & e'_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.14)$$

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

$$\begin{pmatrix} 0 \\ t \\ 1 \end{pmatrix} \times \begin{pmatrix} 1 \\ 0 \\ f \end{pmatrix} = \begin{pmatrix} tf \\ 1 \\ -t \end{pmatrix} \quad (6.15)$$

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

$$d(m', l'(t))^2 = \frac{t^2}{1 + (tf)^2} \quad (6.16)$$

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

$$Ax + By - C = 0 \quad (6.17)$$

Die Selbe Gerade kann auch in Normalform ausgedrückt werden

$$\vec{n} \cdot (\vec{x} - \vec{p}) = 0 \quad (6.18)$$

$$\begin{pmatrix} A \\ B \end{pmatrix} \cdot (\vec{x} - \vec{p}) = 0 \quad (6.19)$$

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

$$\vec{v} = \frac{\vec{p} \cdot \vec{n}}{\vec{n} \cdot \vec{n}} \cdot \vec{n} \rightsquigarrow \frac{-C}{A^2 + B^2} \cdot \begin{pmatrix} A \\ B \end{pmatrix} \quad (6.20)$$

$$\|\vec{v}\| = \frac{|\vec{p} \cdot \vec{n}|}{\|\vec{n}\|^2} \cdot \|\vec{n}\| \rightsquigarrow \|\vec{v}\| = \frac{|\vec{p} \cdot \vec{n}|}{\|\vec{n}\|} \quad (6.21)$$

$$\Rightarrow |C| = |\vec{p} - \vec{n}| \quad (6.22)$$

$$\Rightarrow |\sqrt{A^2 + B^2}| = \|\vec{n}\| \quad (6.23)$$

$$\|\vec{v}\| = \frac{|C|}{\sqrt{A^2 + B^2}} \quad (6.24)$$

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

$$A = tf, B = 1, C = -t, \vec{v} = d \quad (6.25)$$

$$d^2 = \frac{t^2}{\sqrt{((tf)^2 + 1^2)^2}} = \frac{t^2}{(tf)^2 + 1^2} = \frac{t^2}{1 + (tf)^2} \quad (6.26)$$

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

$$l'(t) = F(0, t, 1)^T = (-f'(ct + d), at + b, ct + d)^T. \quad (6.27)$$

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

$$d(m', l'(t))^2 = \frac{(ct + d)^2}{(at + b)^2 + f'^2(ct + d)^2} \quad (6.28)$$

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

$$C(m, m') = d(m, \hat{m})^2 + d(m', \hat{m}')^2 \quad (6.29)$$

$$\rightsquigarrow C(m, m') = d(m, l(t))^2 + d(m', l'(t))^2 \quad (6.30)$$

$$\rightsquigarrow s(t) = \frac{t^2}{1 + (tf)^2} + \frac{(ct + d)^2}{(at + b)^2 + f'^2(ct + d)^2} \quad (6.31)$$

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

$$g(t) = t((at + b)^2 + f'^2(ct + d)^2)^2 - (ad - bc)(1 + f^2t^2)^2(at + b)(ct + d) \quad (6.32)$$

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

$$l = (tf, 1, -t) \quad (6.33)$$

$$\hat{m}_{Rot} = (-(tf) \cdot v, -1 \cdot v, (tf)^2 \cdot 1^2) \quad (6.34)$$

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

$$\hat{m} = T^{-1}R^T \hat{m}_{Rot} \quad (6.35)$$

$$\hat{m}' = T'^{-1}R'^T \hat{m}'_{Rot} \quad (6.36)$$

Vergleicht man die Punkte m und \hat{m} und die Punkte m' und \hat{m}' , so kann die minimalen Abweichungen der Punkte voneinander sehen. Um nun noch den Punkt \hat{M} im 3D-Raum zu rekonstruieren, kann nun

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

$$\hat{m} \times (P\hat{M}) = 0 \quad (6.37)$$

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

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

$$x(p^{3T}X) - (p^{1T}X) = 0 \quad (6.39)$$

$$y(p^{3T}X) - (p^{2T}X) = 0 \quad (6.40)$$

$$x(p^{3T}X) - y(p^{1T}X) = 0 \quad (6.41)$$

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

$$A = \begin{bmatrix} xp^{3T} - p^{1T} \\ yp^{3T} - p^{2T} \\ x'p'^{3T} - p'^{1T} \\ y'p'^{3T} - p'^{2T} \end{bmatrix} \quad (6.42)$$

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

6.4 Ergebnisse einer Stereoanalyse mit Kameras unterschiedlicher Auflösung

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

$$K' = \begin{bmatrix} \alpha_x & s & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \quad (6.43)$$

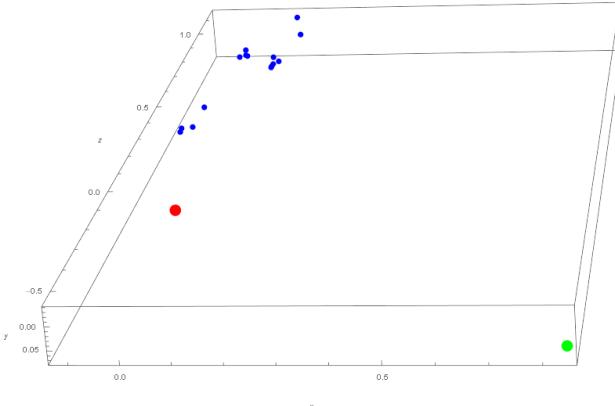


Abbildung 6.15: Rekonstruierte Szene, unskaliert in Pixeleinheiten

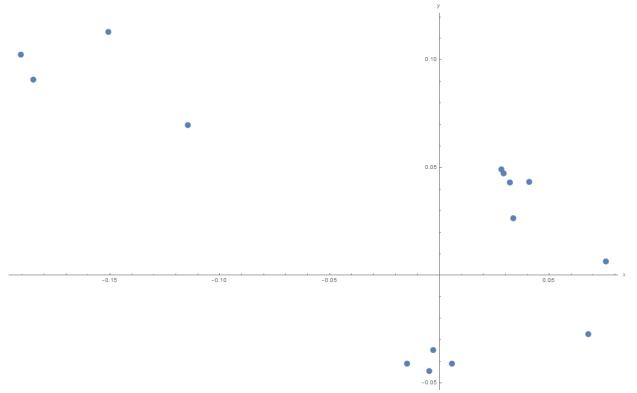


Abbildung 6.16: Rekonstruierte Szene, unskaliert, in Pixeleinheiten und in einem 2D-Plot geschrieben

α_x und α_y setzen sich auch dem Abstand des Kamerazentrums zum Hauptpunkt zusammen, welcher in dieser Arbeit als mit ζ bezeichnet wurde, und den Kantenlängen der Pixel auf dem Sensor m_x und m_y . Um die Auflösung der Kamera zu verändern, wird auf α_x und α_y jeweils ein beliebiger Faktor dazu multipliziert. Zum Beweis, dass die Rekonstruktion der externen Kameraparameter und die Szenerekonstruktion, bei egal welcher Skalierung, die ähnlichen Ergebnisse liefern, wurde die Kameramatrix K' mit den Verhältnissen [2 : 2], [5 : 2], [2 : 1], [1 : 2] und [1.2 : 2.3] skaliert.

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

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

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

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

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

Die Formulierung, dass die jeweils neu rekonstruierten Szenen ähnlich sind, wurde deshalb verwendet, da durch die zuvorigen Fehler der korrespondierenden Punkte und später, bei der Triangulierung, durch die *Sampson-Approximation* Abweichungen auftreten können. Als Beweise werden im folgenden vier Beispiele für die vier Lösungen der rekonstruierten Translationsmatrizen R' aufgezeigt. Des Weiteren werden die 3D-Plots und 2D-Plots der rekonstruierten Szenen bei unterschiedlich Auflösungen im

Vergleich mit der Szene bei gleichen Auflösungen gezeigt. Die Koordinaten sind in unskalierten Pixel-einheiten gegeben. Die Originalszene ist in Abbildung 7.15 und 7.16 zu sehen. Zu beachten ist, das die Ausgabe des 3D Plots in *Mathematica* manchmal rechtsdrehend, manchmal linksdrehend dargestellt sind, weshalb der Eindruck aufkommt, die Szene und die Kameraposition seien gespiegelt dargestellt. Dies leidet auf ein generellen Darstellungsproblem von 3D-Plots in *Mathematica* zurückzuführen. Dieses kann mit zusätzlichem Code bereinigt werden, wurde aber zu diesem Zeitpunkt noch nicht implementiert.

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

Abbildung 6.17: Zeigt die rekonstruierte Matrix R' bei unveränderter Auflösung. Die Auflösungen von C_δ und C'_δ sind die selben.

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

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

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

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

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

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

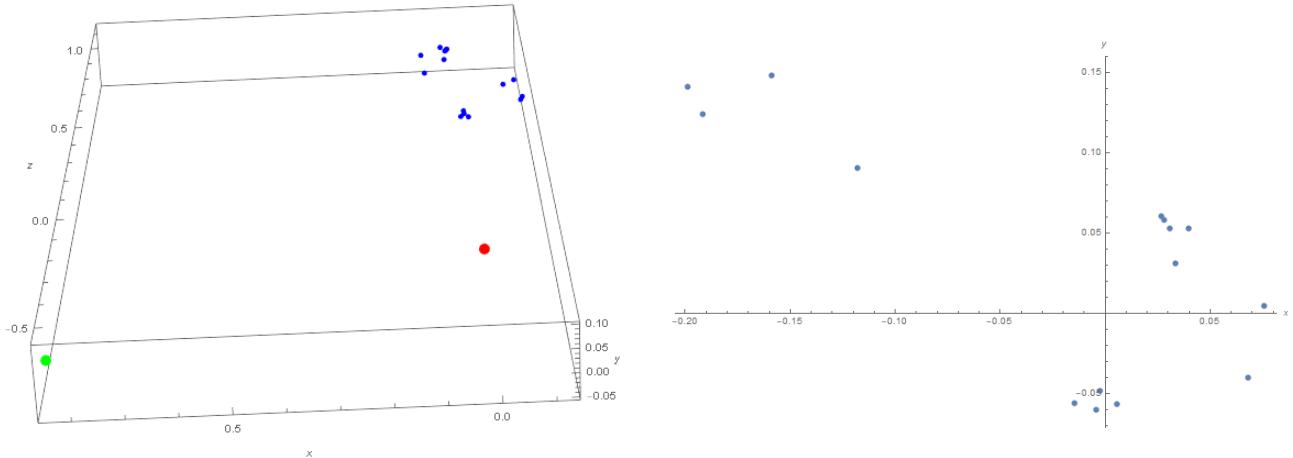


Abbildung 6.21: Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[5 : 2]$ skaliert wurde

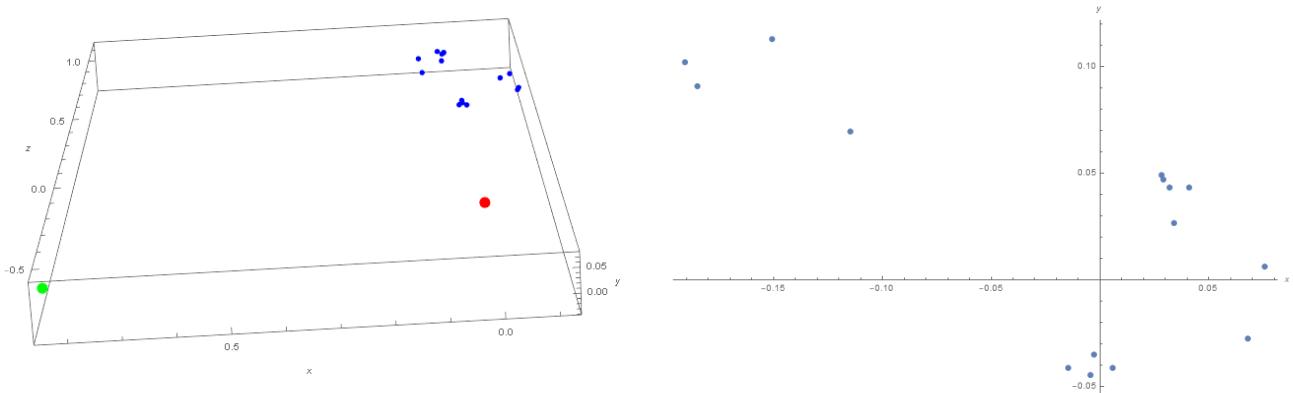


Abbildung 6.22: Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[2 : 1]$ skaliert wurde

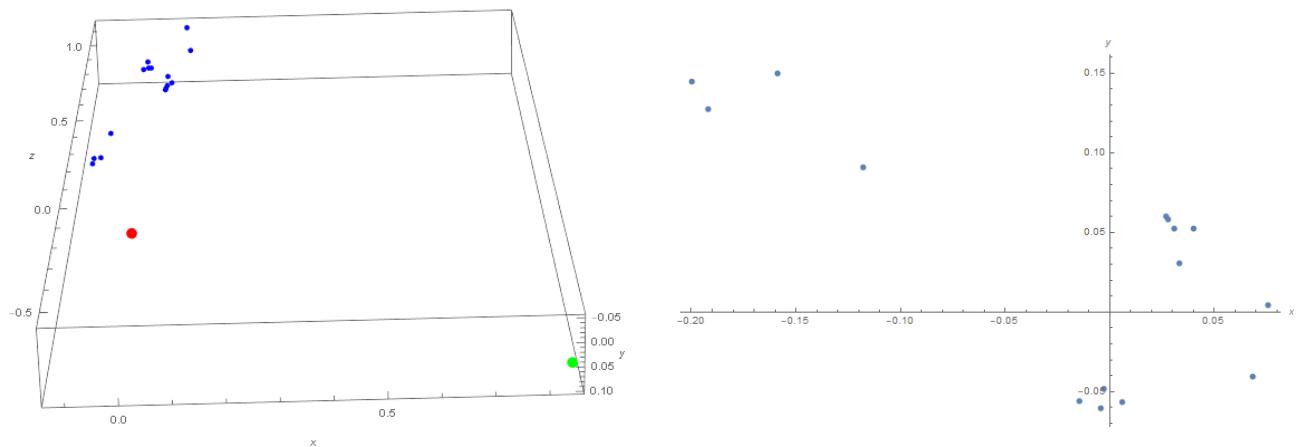


Abbildung 6.23: Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[1 : 2]$ skaliert wurde

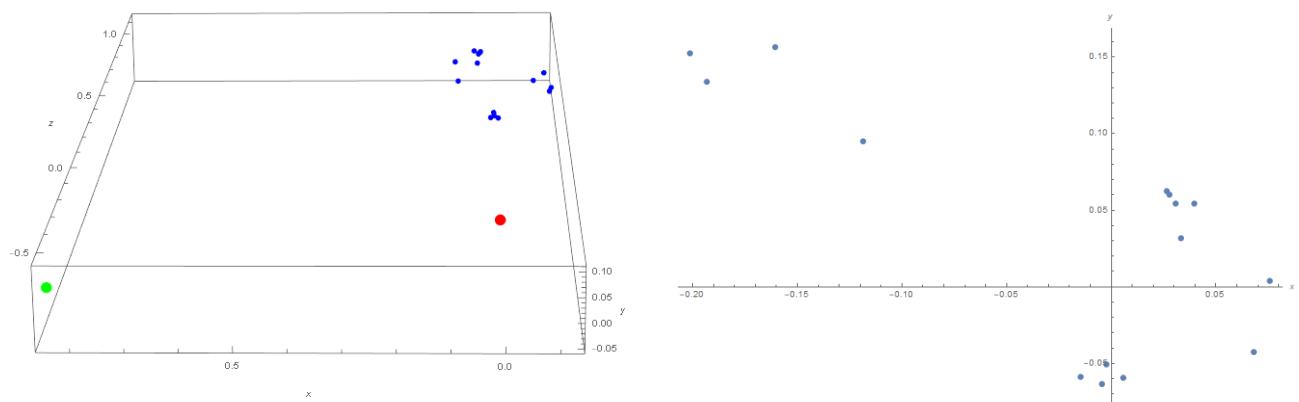


Abbildung 6.24: Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[1.2 : 2.3]$ skaliert wurde

7 Vergleich entwickelter RekonstruktionsAlgorithmus mit bereits vorhandenen (Matlab)

(ÜBERSCHRIFT ÄNDERN!)

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

Arbeitsprozess Matlab eifügen

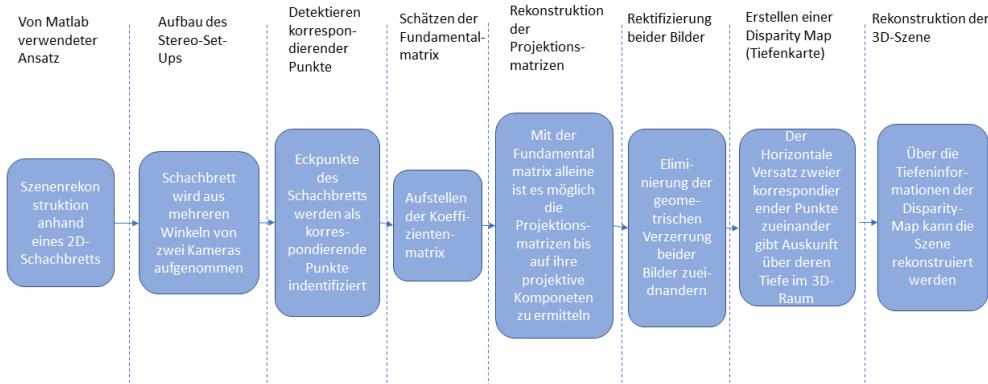


Abbildung 7.1: Durch Ungenauigkeiten in der korrespondierenden Punkte, verfehlen sich die Linien und es kommt zu keinem Schnittpunkt

Ein weiteres weit verbreitetes Verfahren, ist cor der Szenenrekonstruktion durch Triangulierung eine Rektifizierung beider Bilder vorzunehmen[19, 20, 21, 22]. Da bestimmte Formen der Rektifizierung keine vorherige Kalibrierung der Kameras benötigen, wird diese Methode in den meisten gängigen Echtzeit-Szenenrekonstruktionen eingesetzt. [22, 21, 23]. Rektifizierte Bilder müssen zwei Eigenschaften erfüllen. Zum einen müssen alle Epipolargeraden parallel zur x-Koordinatenachse verlaufen und zweitens müssen alle korrespondierenden Punkte die selben y-Koordinaten besitzen[20]. Mit Hilfe dieser Eigenschaften ist es somit möglich die entstandenen korrespondierenden Epipolarlinien als horizontale Scanlinien zu benutzen[21, 20]. Mit hilfe dieser Scanlinien und den darauf sich befindenden korrespondierenden Punkten ist es zum Beispiel Möglich eine Tiefenkarte des Bildes zu berechnen allein

durch die Differenz der horizontalen Lage der korrespondierenden Punkte[21, 20].



Abbildung 7.2: Beispiel eines rektifizierten Bildes. Quelle: [20]



Abbildung 7.3: Beispiel einer einfachen Tiefenkarte eines Stereobildpaars nach der Rektifizierung. Quelle: [21]

Die Rektifizierung, allem voraus vor allem die Optimierung des Rektifizierungsvorgangs, von Stereo- oder auch multplen- Kamerasystemen, wird heutzutage von vielen Entwicklergruppen der Computer Vision untersucht(Der satz ist mist!). Es gibt mittlerweile viele Ansätze, jedoch funktionieren nicht alle bei den selben Fällen. So setzen zum Beispiel manche Rektifizierungsalgorithmen voraus, dass die Bilder von Kameras mit selber Auflösung aufgenommen wurden. Ein Beispiel ist die Rektifizierung welche in *Matlab* verwendet wird [19]. Die Rektifizierung wurde anhand einer Methode implementiert, welcher sich ähnlich verhält wie in [24] beschrieben. Die Grundidee hier hinter ist, dass die Kamera- matrizen von zwei Kameras so aufgebaut sind dass die intrinsischen Parameter die selben sind, sie sich

aber in ihren Rotationen und Translationen voneinander unterscheiden. Die extrinsischen Kameraparameter werden dann dementsprechend so manipuliert, dass die Bildebenen Achsenparallel zueinander stehen[24, 22]. Um horizontale Epipolarlinien zu erhalten muss gleichzeitig die Basislinie zwischen den zwei Kamerazentren parallel zur neuen x-Achse beider Kameras sein. Zudem soll, um eine angemessene Rektifizierung zu gewährleisten, müssen konjugierende Punkte die selbe vertikale Koordinate haben. Dies wird hier durch die Bedingung gewährleistet, dass beide Kameras die selben intrinsischen Parameter haben[24]. Eine Frage welche mit unter in dieser Arbeit beantwortet werden sollte, war, ob es möglich ist, ohne deutlich größeren Aufwand eine Kamerakalibrierung und Szeneriekonstruktion mit Kameras unterschiedlicher Auflösung zu gewährleisten. Im Kapitel geometrische Erläuterung der Epipolargeometrie, in welchem ausführlich die Epipolargeometrie vorgestellt wurde, wurde bereits Bezug auf die unterschiedlichen Auflösungen genommen. Prinzipiell spielen unterschiedliche intrinsische Kameraparameter keine Rolle, wenn es um die Rekonstruktion der Kameraposen geht, da die Fundamental Matrix und die essentielle Matrix die Information über die intrinsischen und extrinsischen Kameraparameter besitzen und es klar gestellt wurde, dass die Bildkoordinatensysteme der Kameras nicht identisch sein müssen. [5]. In dieser Arbeit wurde ein Rektifizierungsalgorithmus nach *Zhang*[20] implementiert, welcher sich die Fundamentalmatrix zu nutzen macht. *Loop* und *Zhang* zerlegen jede Kollinearität in eine Ähnlichkeitstransformation, eine Schertransformation und eine projektive Transformation. Die projektive Komponenten wird dabei in einem nichtlinearen Optimierungsprozess so affin wie möglich gemacht.[22, 20]. Im folgenden wird zunächst der genaue Vorgang des implementierten Algorithmus genauer erklärt und **des Weiteren werden zwei Beispiele vorgestellt, welche die Bilder des Minimalbeispiels einmal mit gleichen intrinsischen Parametern und einmal mit unterschiedlichen intrinsischen Parametern der Kamera aufzeigt. Es wird sich Herausstellen, dass beide Beispiele eine gelungene Rektifizierung der Bilder aufweisen.(Nochmal genau nachprüfen ob das geht!!!)**. Während sich einige Rektifizierungsverfahren im 3D-Raum abspielen, wird beim Verfahren nach *Zhang*, hauptsächlich im 2D-Raum gearbeitet. Des Weiteren wird vorausgesetzt, dass die Fundamental Matrix F und somit auch korrespondierende Punkte bereits bekannt sind. Sind die intrinsischen Kameraparameter bekannt, so wird aus der Fundamentalmatrix die Essentielle Matrix. Das Verfahren kann sowohl in einem kalibrierten als auch in einen unkalibrierten Fall angewendet werden[20]. Im Algorithmus wurde der unkalibrierte Fall implementiert und somit wird in der Erläuterung und in den danach folgenden Beispielen die Fundamentalmatrix F verwendet. Die korrespondierenden Punkte werden mit x für das erste beziehungsweise x' für das zweite Bild definiert, die Kamerazentren dementsprechend mit C und C' . Bildebene der ersten Kamera wird mit I definiert und die Bildebene von Kamera zwei mit I' , die entsprechenden Epipole mit e und e' . Der Prozess der im Algorithmus erfolgt kann quasi als eine Transformation der Epipolar Geometrie eines Bildpaars in eine kanonische Form angesehen werden. Diese Transformation wird durch eine Homographiematrix durchgeführt, welche sich aus den bereits erwähnten drei Komponenten zusammenstellt. Zu Beginn sei noch erwähnt dass wir pro Bild zwei unterschiedliche Homographien H und H' brauchen. Die Fundamentalmatrix liefert, die Epipolarbedingung, dass $x'^T F x = 0$ ergibt, wenn x' auf der zu x korrespondierenden Epipolarlinie liegt. Die korrespondierenden Punkte x und x' werden, für die Rektifizierung, jeweils mit den Homographien H und H' verrechnet.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

7.0.1 Projektive Transformation

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

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

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

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

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

Jedoch sollen sie nicht ganz null werden, da die projektive Matrix dann keine projektive mehr wäre, sondern eine affine. Deswegen heißt es auch sie soll so affin wie möglich gemacht werden. Das selbe gilt natürlich auch für w'_a und w'_b aus H'_p . Wäre das der Fall, so wären die beiden Epipole e und e' bereits im unendlichen und die Matrizen H_p und H'_p hätten keine Auswirkungen auf die Punkte. Für die Minimierung wird die Methode des *least-square-fitting*, also die Anpassung des kleinsten Quadrats,

genutzt[25]. Es werden also die Gewichtungen der Punkte in beiden Bildern in der Methode der Anpassung der kleinsten Quadrate verbaut, welche versucht eine Funktion zu finden, die einen Wert für z berechnen soll welcher die Bildverzerrung minimal hält. Anders ausgedrückt man sucht einen Wert für z , welcher am nächsten an den gegebenen Punktesammlungen der jeweiligen Bildern dran liegt, wobei für z bereits gilt, dass es sich um einen Punkt im Unendlichen handeln soll[20, 25]. Angenommenen, dass die Annäherungsfunktion $g(x)$ eine Funktion $f(x)$, mit $x \in [a, b]$, annähern soll, dann versucht die Methode, die Summe der Quadrate der ordinatischen Differenzen, welche zwischen den von der Funktion generierten Punkten und den Punkten aus den Daten gewonnen wird, zu minimieren[25, 26]. Zum Beispiel werden n Datenpunkte angenommen, dann gilt:

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

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

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

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

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

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

Vereinfacht lässt sich das auch in einer Matrixgleichung angeben

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

in welcher für P gilt:

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

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

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

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

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

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

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

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

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

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

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

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

Hier soll das Polynom aufgestellt werden, ist aber nicht mehr klar wie das ging!!! (7.33)

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

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

Da es sich um eine nicht lineare Optimierung handelt ist die Minimierung von $\frac{z^T A z}{z^T B z}$ gleichzusetzen mit der Maximierung von $\frac{z^T B z}{z^T A z}$. Beide als eine Funktion von $f(z)$. Matrix A wird mit der Cholesky-Zerlegung in zwei höhere Dreiecksmatrizen zerlegt $A = D^T D$ [27]. Dies geht nur da A nachweislich eine

symmetrische und positiv-definite Matrix ist.[27] positiv-Definite bedeutet, dass die Singularwerte von A immer positiv bleiben, egal mit welchem Vektor z diese multipliziert wird. (**HIER NOCH LITERATUR FINDEN UND NOCHMAL PRÜFEN OB DEFINITION SO STIMMT**). Des Weiteren wir definiert, dass $y = Dz$ ist und $f(z)$ wird dann zu einen $\hat{f}(y)$

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

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

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

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

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

Durch die Definition von $y = Dz$ ist y bis auf einen Skalierungsfaktor definiert. $\hat{f}(y)$ ist maximiert, wenn y gleich dem Eigenvektor von $D^{-T} B D - 1$ ist, welcher mit dem größten Eigenwert assoziiert wird. Zum Schluss erhalten wir dann einen Wert für \hat{z}_1 mit $\hat{z}_1 = D^{-1}y$. Exakt das selbe Verfahren wird für die Findung von \hat{z}_2 mit $\frac{\hat{z}^T B' \hat{z}}{\hat{z}^T A' \hat{z}}$ angewandt. Sind \hat{z}_1, \hat{z}_2 und eine erste Schätzung für z gefunden, so kann ein Wert für z gesucht werden, welcher noch näher an ein optimales Ergebnis heranreicht. Beide Lösungen \hat{z}_1 und \hat{z}_2 , werden in die Funktion $f(z)$ eingesetzt und es jeweils ein Wert ermittelt, welcher am nächsten an einem Nullpunkt sich befindet. So kann iterativ eine optimale Lösung für z gefunden werden. Ist der Wert für z bestimmt, so kann dieser die Gleichungen 4.75 und 4.76 eingesetzt werden und w beziehungsweise w' bestimmt werden, welche die Elemente für die Matrizen H_p und H'_p bereitstellen. Abbildung 4.7 zeigt in rot den Quader des Minimalbeispiels wie er in Kamera zwei abgebildet ist und in grün wie er in Kamera eins abgebildet ist. Kamera zwei ist horizontal zu Kamera eins verschoben und um 45° zu Kamera eins um die eigene vertikale Achse gedreht. Die Auflösungen beider Kameras sind identisch, sprich die intrinsischen Kameraparameter sind die selben. Abbildung 4.8 zeigt die momentanen Epipolarlinien. Die Epipolarlinien von Bild eins, also dem grünen Abbild, sind bereits Parallel, was aber keine Voraussetzung für die Funktion des Rektifizierungsalgorithmus ist. Der Schnittpunkt der Epipolarlinien von Bild zwei, also dem Roten Abbild, treffen sich in einem Punkt und bilden somit den Epipol von Bild zwei.

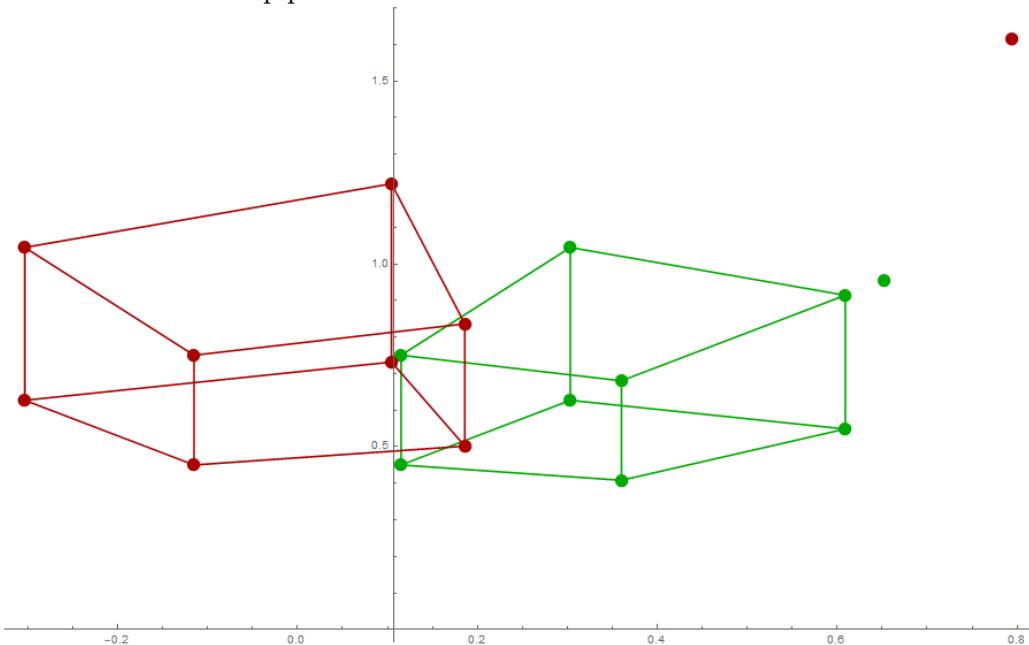


Abbildung 7.4: Aufnahmen zweier Kameras mit den selben Auflösungen, Kamera eins(Grün) und Kamera(rot) zwei gelten jeweils $\zeta = 1$

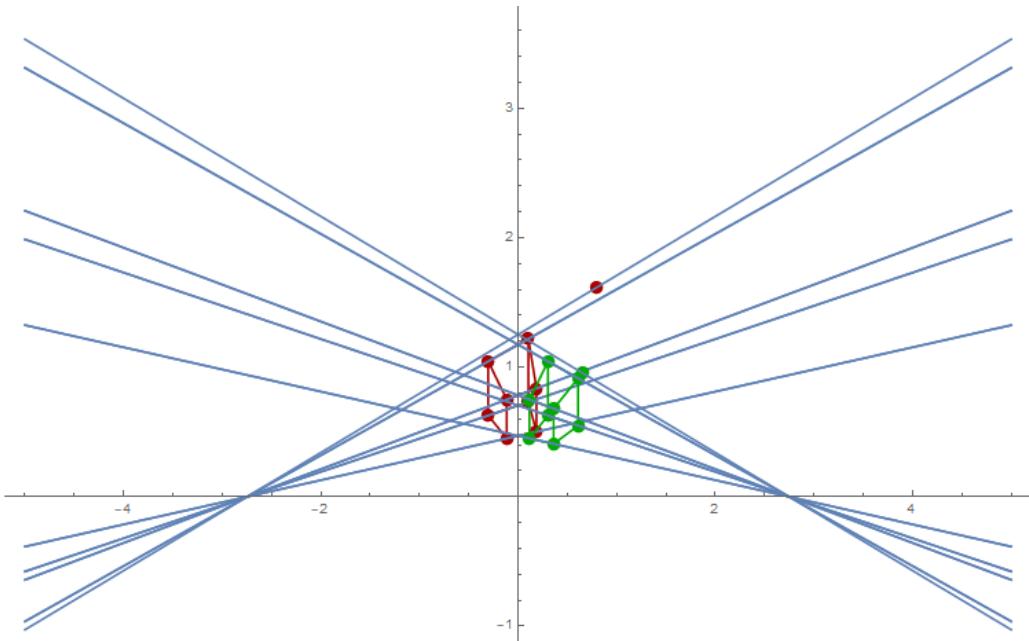


Abbildung 7.5: Epipole für Kamera eins und Kamera zwei vor der Rektifizierung

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

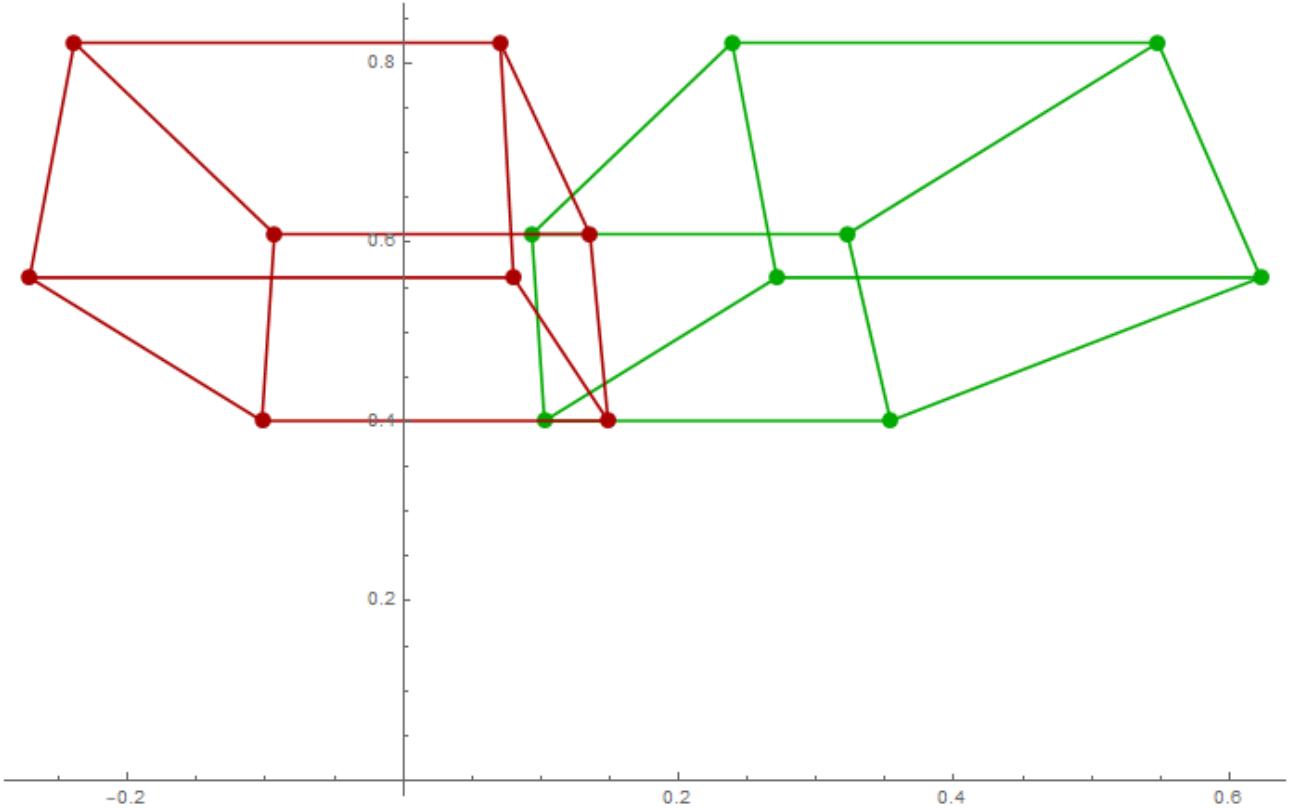


Abbildung 7.6: Abbildung beider Bilder nach anwenden der Matrizen H_p und H'_p . Die Epipole beider Bilder sind nun im unendlichen. Das die entstehenden parallelen Epipolarlinien auch hier schon horizontal ausgerichtet sind ist Zufall. Die Epipolarlinien sind immer parallel nach dieser Transformation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

7.0.2 Ähnlichkeitstransformation

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

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

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

$$(7.42)$$

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

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

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

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

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

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

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

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

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

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

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

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

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

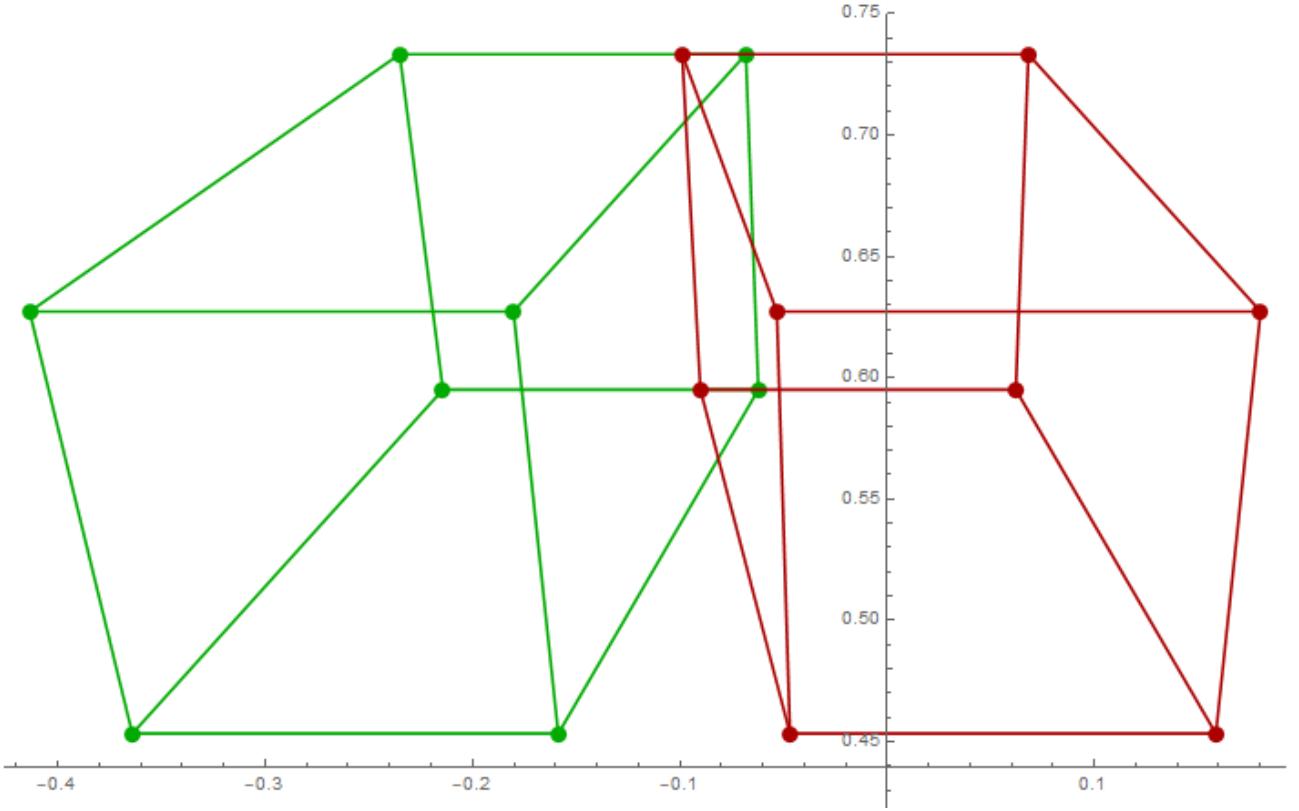


Abbildung 7.7: Abbildung beider Bilder nach anwenden der Matrizen $H_r \cdot H_p$ und $H'_r \cdot H'_p$. Die Epipolarlinien sind nun horizontal zueinander ausgerichtet

7.0.3 Scherungstransformation

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

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

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

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

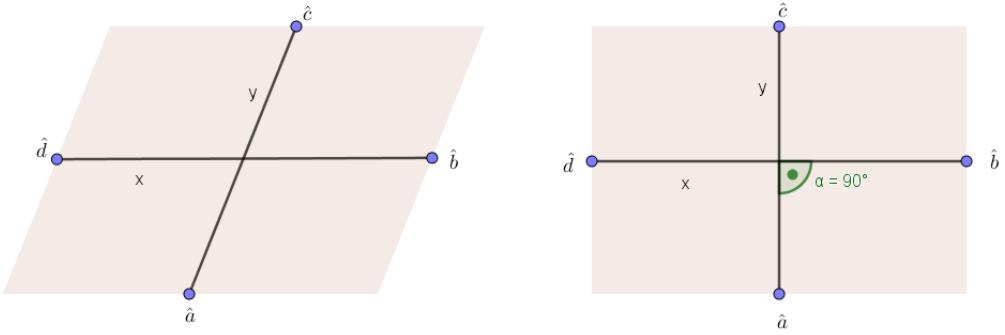


Abbildung 7.8: Die Abbildung verdeutlicht noch mal Schematisch, wie sich die Punkte ausrichten sollen. Bild a) zeigt die durch die Rektifizierung verschobenen Bildkantenmitten. Bild b= zeigt, wie sich die Bildkantenmitten durch die Scherungstransformation wieder ausrichten sollen.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Die Seitenverhältnisse der Bilder werden beibehalten, wenn gilt:

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

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

Für u_a, u'_a, u_b und u'_b jeweils Gleichungen auf Basis der jeweiligen Bild Höhen und Breiten w, w', h, h' und x, x', y und y' und unter einhaltung der Aussagen der Gleichungen 5.118 bis 5.121, aufgestellt werden[20, 28].

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

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

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

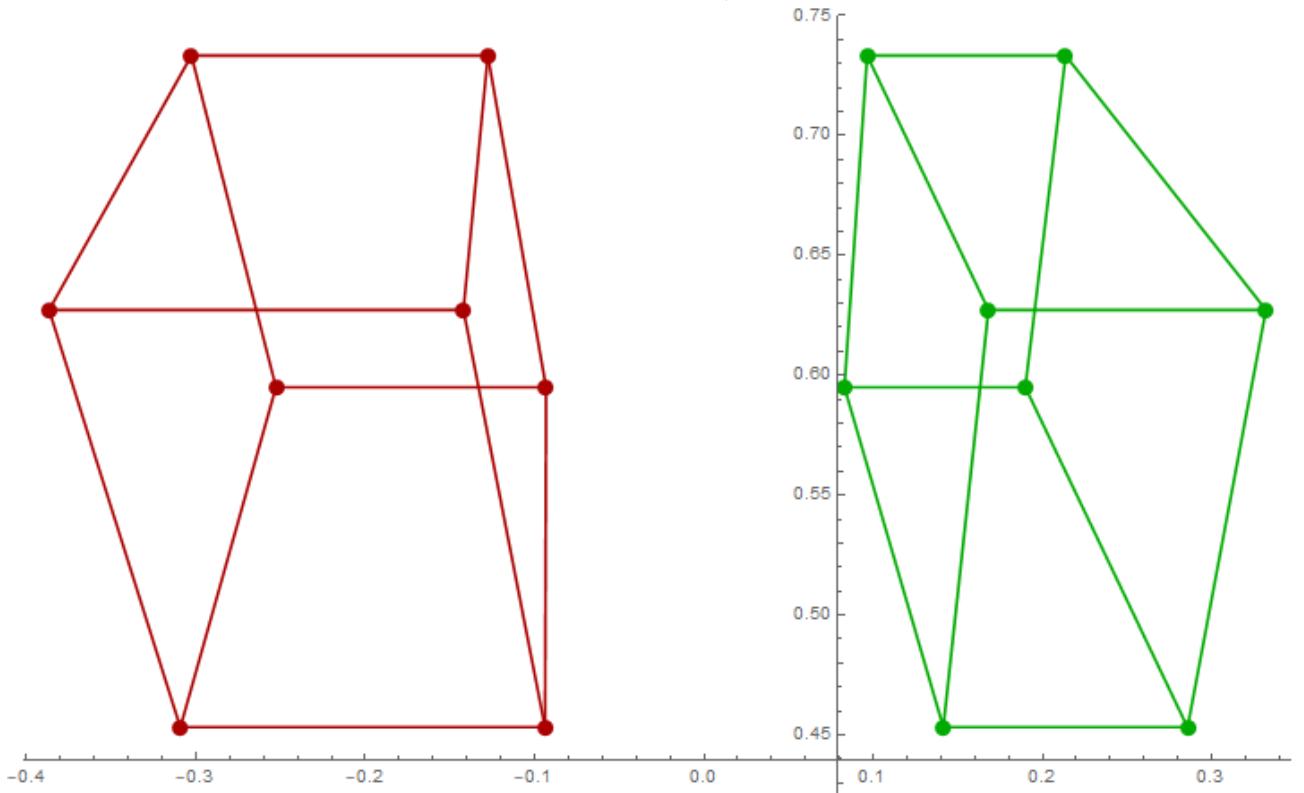


Abbildung 7.9: Abbildung beider Bilder nach anwenden der Matrizen $H_s \cdot H_r \cdot H_p$ und $H'_s \cdot H'_r \cdot H'_p$. Die horizontale Verzerrung wurde reduziert.

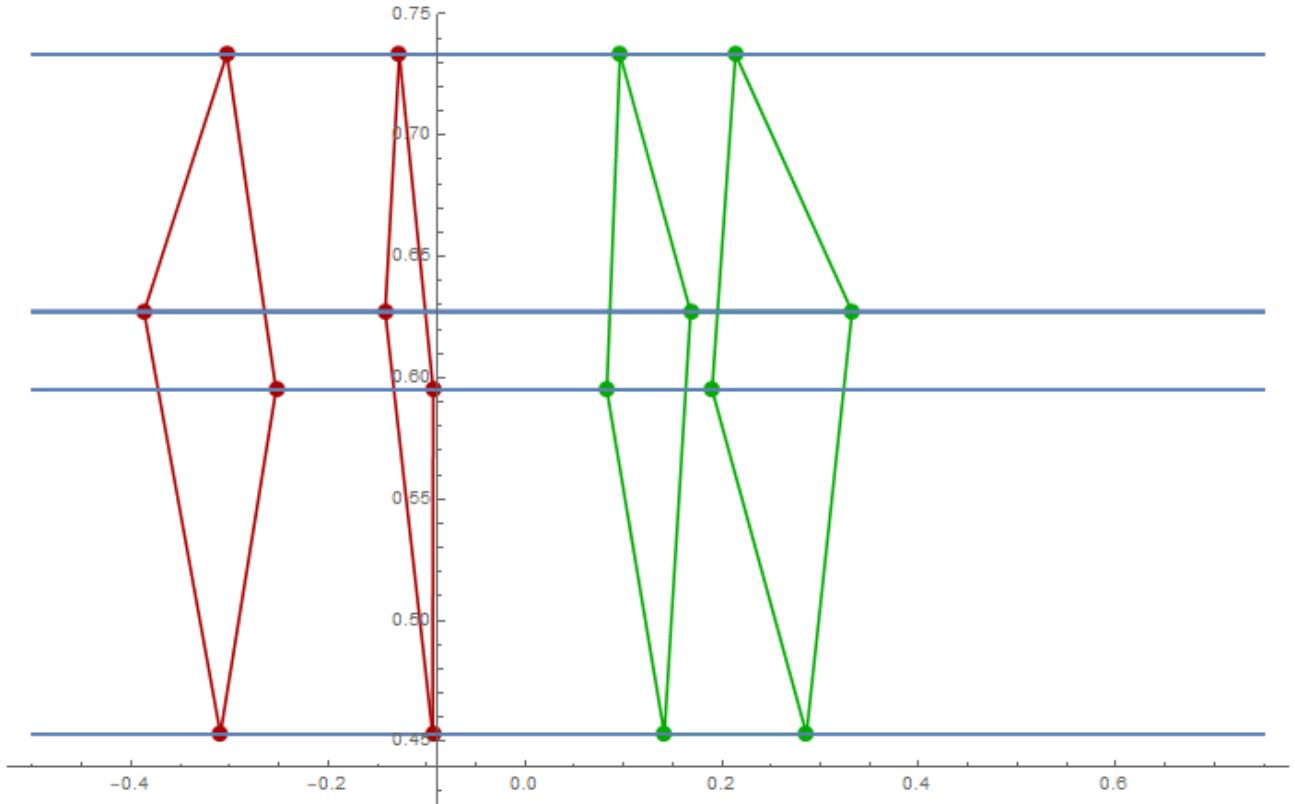


Abbildung 7.10: In dieser Abbildung wurden die Epipolarlinien noch in den Grafikplot mit eingebaut

8 Punktesortierung in Schachbrettmustern

8.1 Algorithmus zur Punktesortierung in verzeichneten Schachbrettbildern

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

8.1.1 Vorläufiges Klassendiagramm

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

Abbildung 8.1: Klassendiagramm

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

Abbildung 8.2: Klassendiagramm

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

Abbildung 8.3: Klassendiagramm

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

Abbildung 8.4: Klassendiagramm

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

Abbildung 8.5: Klassendiagramm

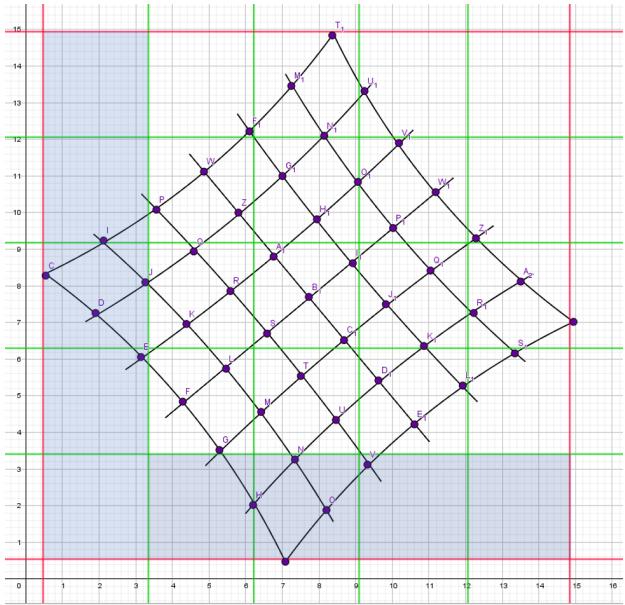


Abbildung 8.6

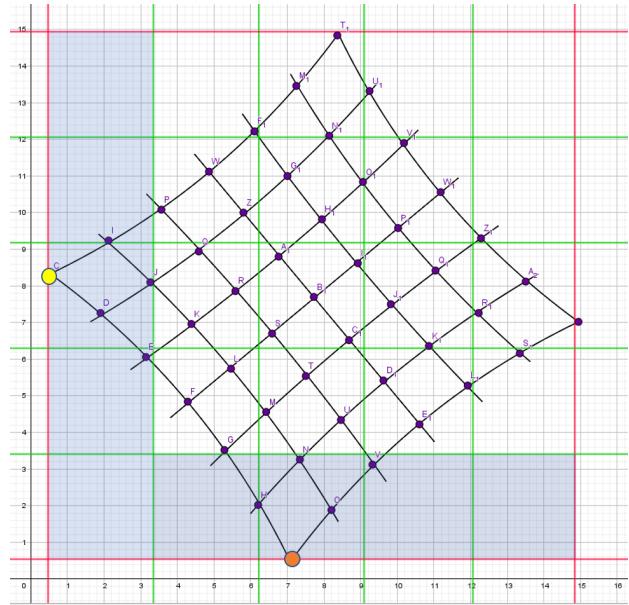


Abbildung 8.7

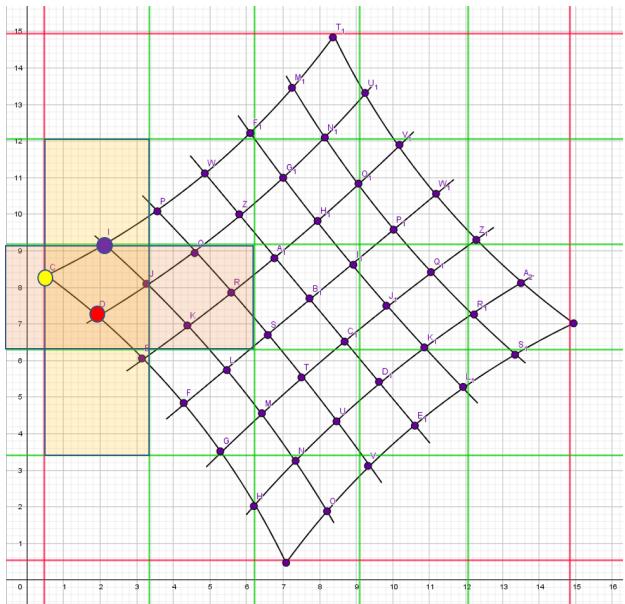


Abbildung 8.8

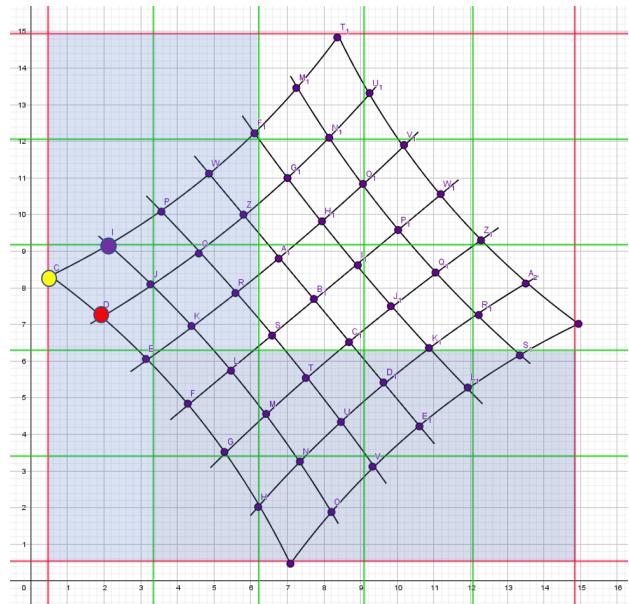


Abbildung 8.9

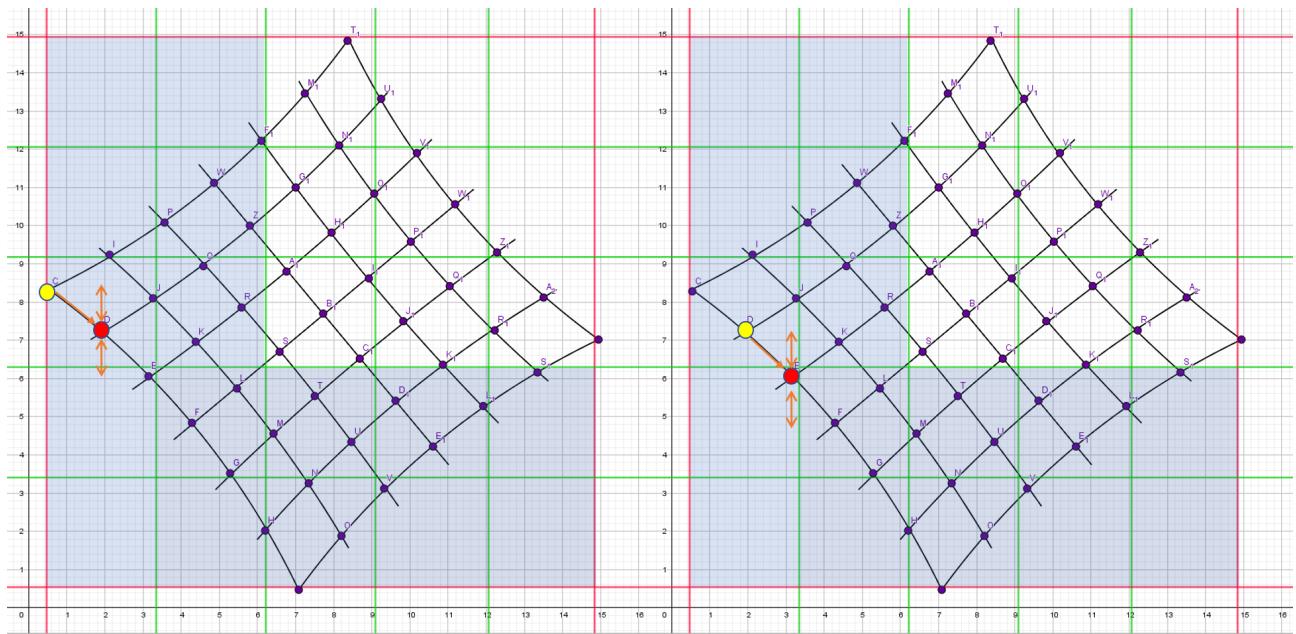


Abbildung 8.10: Klassendiagramm

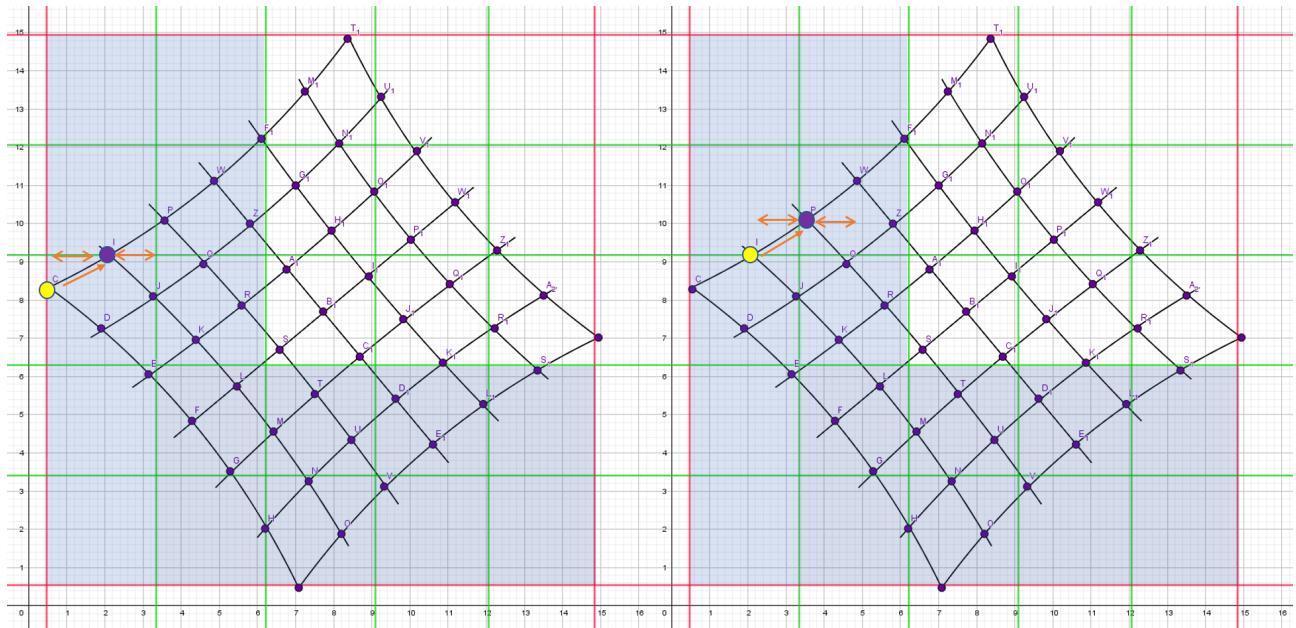


Abbildung 8.11: Klassendiagramm

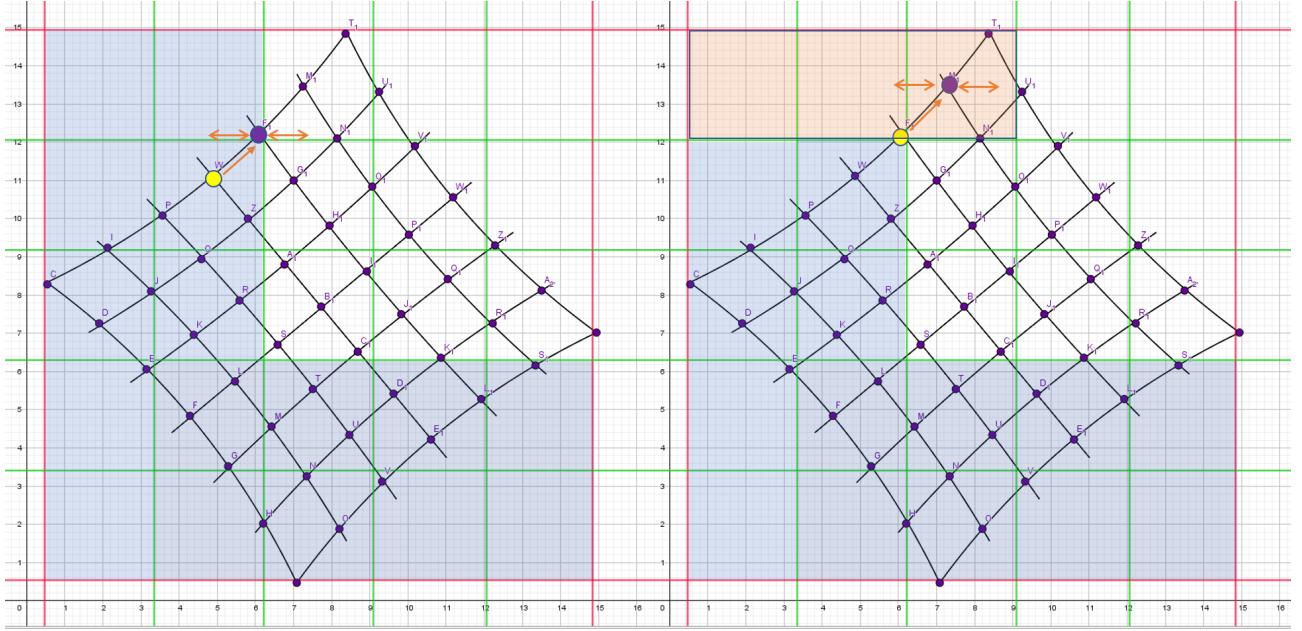


Abbildung 8.12: Klassendiagramm

8.1.2 Beispiele

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

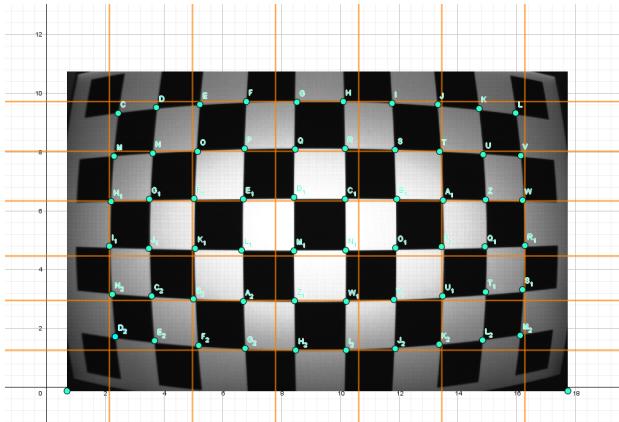


Abbildung 8.13: Bild eines Tonnenförmig verzeichneten Schachbretts

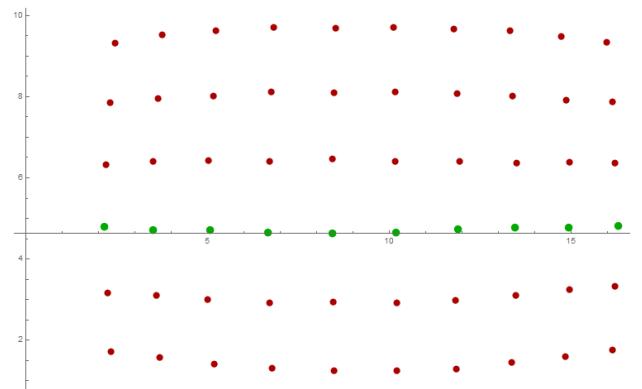


Abbildung 8.14: Algorithmisch detektierte Linie der dritten i-Reihe

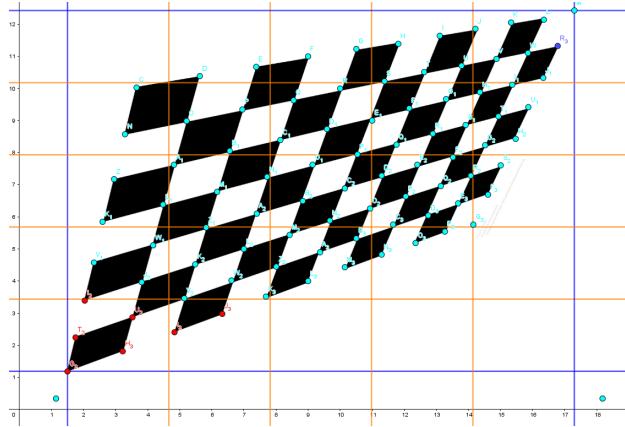


Abbildung 8.15: Bild eines perspektivisch verzerrtem Schachbretts

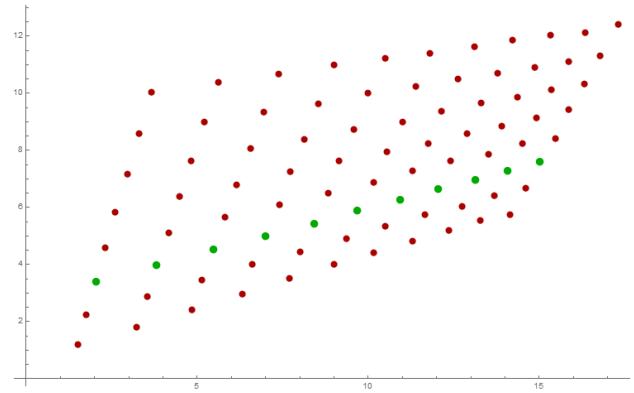


Abbildung 8.16: Algorithmisch detektierte Linie der dritten i-Reihe

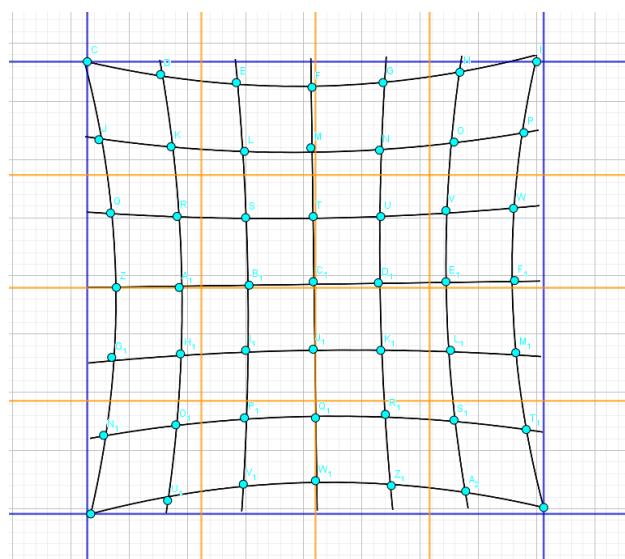


Abbildung 8.17: Bild eines Kissenförmig verzeichnetem Schachbretts

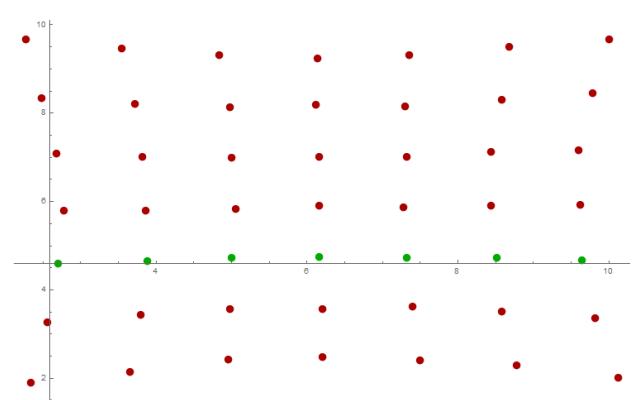


Abbildung 8.18: Algorithmisch detektierte Linie der dritten i-Reihe

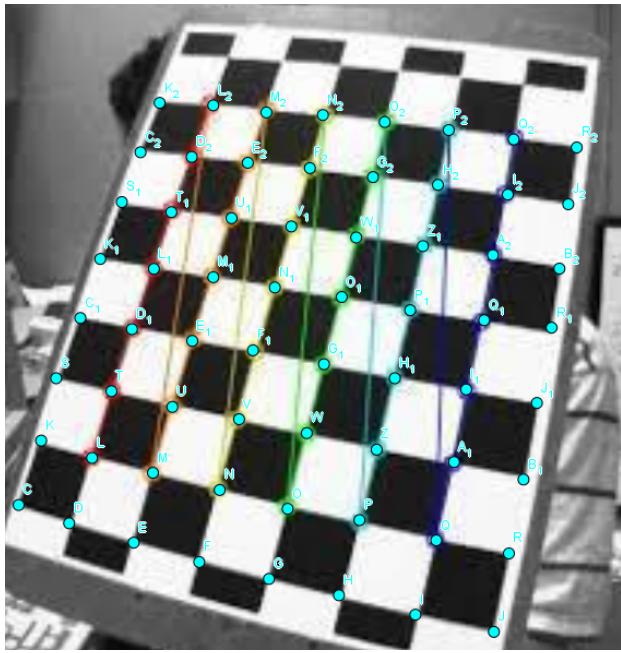


Abbildung 8.19: Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts(GRFIK AUSTAUSCHEN BILD IS KACKE)

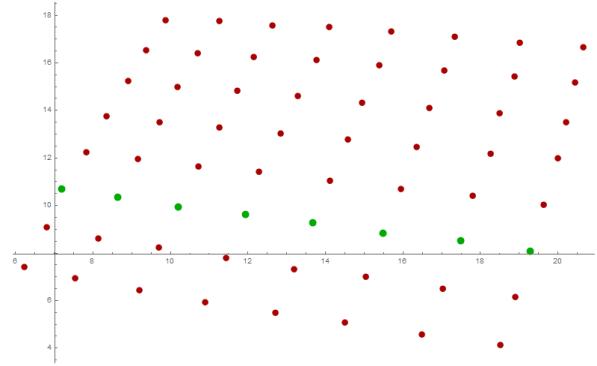


Abbildung 8.20: Algorithmisch detektierte Linie der dritten i-Reihe

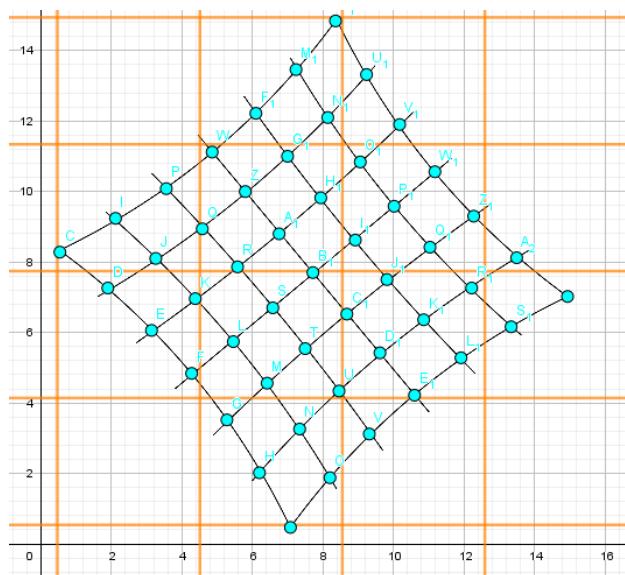


Abbildung 8.21: Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts

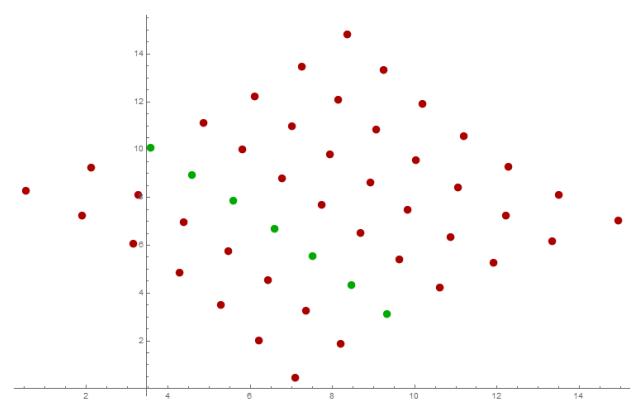


Abbildung 8.22: Algorithmisch detektierte Linie der dritten i-Reihe

9 Fazit - Conclusion

10 Nächste Schritte - next steps

11 Protocol - 10.11.2015

12 Abkürzungsverzeichnis - List of Abbreviations

Abbildungsverzeichnis

2.1	Schematik eines Abbildenden Systems. Ein Punkt M im Weltkoordinatensystem O wird durch eine Kamera C aufgenommen. Diese Aufnahme wird durch eine Projektion, die als Verbindungsline von M zu C dargestellt ist und M auf m abbildet, beschrieben.	7
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	8
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	9
2.4	Das Schaubild zeigt die einzelnen Koordinatensysteme in einem Lochkameramodell. Das Weltkoordinatensystem (O, δ) mit $\delta = (\hat{d}_1, \hat{d}_2, \hat{d}_3, O)$, das Kamerakoordinatensystem (C, β) mit $\beta = (\hat{b}_1, \hat{b}_2, \hat{b}_3, C)$, das Bildebenenkoordinatensystem (I, τ) mit $\tau = (\hat{t}_1, \hat{t}_2, I)$ und das Sensorkoordinatensystem (S, σ) mit $\sigma = (\hat{u}, \hat{v})$	12
3.1	Veranschaulichung der Homographie bei zwei verschieden translatierten und rotierten Kameras.	15
3.2	C und C' sind die Projektionszentren zweier Kameras. Beide Kameras besitzen jeweils eine Bildebene. Die Basislinien verbindet die Projektionszentren der Kameras. Der Punkt an welchem die Basislinie die Bildebenen schneidet, wird als Epipol bezeichnet. Durch den Epipol verlaufen alle Epipolarlinien des Bildes. M ist der Objektpunkt im 3D-Raum und m_1 und m_2 sind die jeweiligen Abbildungen dieses Punktes auf den Bildebenen. Die Verbindungsvektoren zwischen C, C' und M bilden die sogenannte Epipolarebene[5, 3, 1].	17
3.3	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.	18
4.1	Arbeitsprozesse der Stereoanalyse bei Verwendung von eigens erstellten synthetischen Bilddaten	22
4.2	In der Abbildung ist der vereinfachte Stereoaufbau in einer Top-Down-Ansicht zu sehen	23
4.3	In Grün ist die Abbildung des Quaders auf der Bildebenen I von C und in rot ist die Abbildung des Quaders auf der Bildebenen I' von C' zu sehen	23
4.4	Achsen falsch!!! In Blau und Rot sind jeweils das Welt- und Kamerakoordinatensystem von Kamera eins zu sehen . In grün ist das gedrehte Koordiantensystem von Kamera 2 zu sehen.	23
4.5	Die Abbildung zeigt die vier möglichen Lösungen für R .	28
4.6	Optimale Triangulierung: Beide Geraden Treffen sich in einem Punkt im 3D-Raum	28
4.7	Durch Ungenauigkeiten in der korrespondierenden Punkte, verfehlten sich die Linien und es kommt zu keinem Schnittpunkt	29
4.8	Veranschaulichung der Skaleninvarianz und dessen Auswirkung auf die geometrische Form der Objekte	31
4.9	auf Originalgröße skalierte rekonstruierte Szene	31
5.1	Rechteckiger Bildsensor mit darauf sich befindenden quadratischen Pixeln. Vlg [10]	34

5.2 Bild a) zeigt die Interpolation von Pixeln, wenn bei gleichbleibenden Seitenverältnissen weniger Pixel für das Bild verwendet werden sollen. Die interpolierten Pixel leiten dann alle das selbe Signal weiter. Bild b) zeigt in gelb markiert, den verweneten Bereich des Sensors, wenn sich die Seitenverältnisse ändern und nicht mehr der volle Sensor genutzt wird.	35
5.3 C und C' haben die selbe Auflösung eingestellt	36
5.4 C und C' haben unterschiedliche Auflösungen eingestellt	36
5.5 C und C' haben die selbe Auflösung eingestellt	37
5.6 C und C' haben unterschiedliche Auflösungen eingestellt	37
5.7 C und C' haben die selbe Auflösung eingestellt	37
5.8 C und C' haben unterschiedliche Auflösungen eingestellt	37
5.9 Die rekonstruierten Szenenpunkte und Kamerapositionen bleibt auch bei unterschiedlichen Auflösungen die selben	39
 6.1 Aufnahme der Canon 6D von links	40
6.2 Aufnahme der Canon 60D von rechts	40
6.3 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. Die Canon 60D ist etwas hinter der Canon 6D positioniert. Beide Kameras sind zu Szene hin gedreht und auch leicht nach unten geneigt.	41
6.4 Die mit dem <i>SURF</i> -Algorithmus gefundenen Punkte sind mit den gelben Ziffern im Bild gekennzeichnet	42
6.5 Epipolarlinien ohne <i>Epipolar-constraint</i> im Bild der Canon 6D	44
6.6 Epipolarlinien ohne <i>Epipolar-constraint</i> im Bild der Canon 60D	44
6.7 Epipolarlinien mit <i>Epipolar-constraint</i> im Bild der Canon 6D	45
6.8 Epipolarlinien ohne <i>Epipolar-constraint</i> im Bild der Canon 60D	45
6.9 Epipolarlinien mit <i>Epipolar-constraint</i> im Bild der Canon 6D, nach der denormalisierung von F	45
6.10 Epipolarlinien ohne <i>Epipolar-constraint</i> im Bild der Canon 60D, nach der denormalisierung von F	45
6.11 a)	46
6.12 b)	46
6.13 a) Die rückprojizierten Strahlen der ungenauen korrespondierenden Punkte m und m' sind schief und treffen sich nicht in einem Punkt im 3D-Raum. b) The epipolar geometry for m , m' . The measured points do not satisfy the epipolar constraint. The epipolar line $l' = Fm$ is the image of the ray through n , and $l = F^T m'$ is the image of the ray through m' . Since the rays do not intersect, m' does not lie on l' , and m does not lie on l	46
6.14 Frafische Darstellung der optimalen Punkte \hat{m} und \hat{m}'	47
6.15 Rekonstruierte Szene, unskaliert in Pixeleinheiten	52
6.16 Rekonstruierte Szene, unskaliert, in Pixeleinheiten und in einem 2D-Plot geschrieben .	52
6.17 Zeigt die Die rekonstruierte Matrix R' bei unveränderter Auflösung. Die Auflösungen von C_δ und C'_δ sind die selben.	53
6.18 Zeigt die rekonstruierte Matrix R' wenn K' mit einem Verhältnis von $[5 : 2]$ skaliert wurde	53
6.19 Zeigt die rekonstruierte Matrix R' wenn K' mit einem Verhältnis von $[1 : 2]$ skaliert wurde	53
6.20 Zeigt die rekonstruierte Matrix R' wenn K' mit einem Verhältnis von $[1.2 : 2.3]$ skaliert wurde	53
6.21 Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[5 : 2]$ skaliert wurde	54
6.22 Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[2 : 1]$ skaliert wurde	54
6.23 Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[1 : 2]$ skaliert wurde	55
6.24 Rekonstruierte Szene, wenn K' mit einem Verhältnis von $[1.2 : 2.3]$ skaliert wurde	55

7.1	Durch Ungenauigkeiten in der korrespondierenden Punkte, verfehlen sich die Linien und es kommt zu keinem Schnittpunkt	56
7.2	Beispiel eines rektifizierten Bildes. Quelle: [20]	57
7.3	Beispiel einer einfachen Tiefenkarte eines Stereobildpaars nach der Rektifizierung. Quelle: [21]	57
7.4	Aufnahmen zweier Kameras mit den selben Auflösungen, Kamera eins(Grün) und Kamera(rot) zwei gelten jeweils $\zeta = 1$	63
7.5	Epipole für Kamera eins und Kamera zwei vor der Rektifizierung	64
7.6	Abbildung beider Bilder nach anwenden der Matrizen H_p und H'_p . Die Epipole beider Bilder sind nun im unendlichen. Das die entstehenden parallelen Epipolarlinien auch hier schon horizontal ausgerichtet sind ist Zufall. Die Epipolarlinien sind immer parallel nach dieser Transformation aber die Richtung ist nicht immer automatisch bereits $i = [1,0,0]$	65
7.7	Abbildung beider Bilder nach anwenden der Matrizen $H_r \cdot H_p$ und $H'_r \cdot H'_p$. Die Epipolarlinien sind nun horizontal zueinander ausgerichtet	67
7.8	Die Abbildung verdeutlicht noch mal Schematisch, wie sich die Punkte ausrichten sollen. Bild a) zeigt die durch die Rektifizierung verschobenen Bildkantenmitten. Bild b= zeigt, wie sich die Bildkantenmitten durch die Scherungstransformation wieder ausrichten sollen.	68
7.9	Abbildung beider Bilder nach anwenden der Matrizen $H_s \cdot H_r \cdot H_p$ und $H'_s \cdot H'_r \cdot H'_p$. Die horizontale Verzerrung wurde reduziert.	69
7.10	In dieser Abbildung wurden die Epipolarlinien noch in den Grafikplot mit eingebaut .	70
8.1	Klassendiagramm	72
8.2	Klassendiagramm	72
8.3	Klassendiagramm	73
8.4	Klassendiagramm	73
8.5	Klassendiagramm	73
8.6	74
8.7	74
8.8	74
8.9	74
8.10	Klassendiagramm	75
8.11	Klassendiagramm	75
8.12	Klassendiagramm	76
8.13	Bild eines Tonnenförmig verzeichneten Schachbretts	76
8.14	Algorithmisch detektierte Linie der dritten i-Reihe	76
8.15	Bild eines perspektivisch verzerrtem Schachbretts	77
8.16	Algorithmisch detektierte Linie der dritten i-Reihe	77
8.17	Bild eines Kissenförmig verzeichnetem Schachbretts	77
8.18	Algorithmisch detektierte Linie der dritten i-Reihe	77
8.19	Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts(GRFIK AUSTAUSCHEN BILD IS KACKE)	78
8.20	Algorithmisch detektierte Linie der dritten i-Reihe	78
8.21	Bild eines Tonnenförmig verzeichnetem leicht perspektivisch verzerrtem Schachbretts .	78
8.22	Algorithmisch detektierte Linie der dritten i-Reihe	78

Tabellenverzeichnis

5.1	Auflösungen Canon EOS 6D	34
5.2	Vgl [31]	34

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 Pries. *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] MathWorks. Mathworks documentation, stereo camera calibration app. "<https://de.mathworks.com/help/vision/stereo-camera-calibration.html>".
- [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] Christian Heipke. *Photogrammetrie und Fernerkundung*. 2017 Springer, 1. Auflage.
- [9] Jianzhong Lu. *Ein dreidimensionales Bildverarbeitungssystem für die Automatisierung visueller Prüfvorgänge*.
- [10] Rolf Martin Ekbert Hering. *Photonik, Grundlagen, Technologien und Anwendung*. Springer-Verlag Berlin Heidelberg, 2006.
- [11] Zhengyou Zhang. *Epipolar Geometry*, pages 247–258. Springer US, Boston, MA, 2014.
- [12] 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.
- [13] Jeanne Peiffer and Amy Dahan-Dalmedico. *Wege und Irrwege - Eine Geschichte der Mathematik*. 1986, 1994 Springer Basel AG, Editions du Seuil, Springer Basel AG, aus dem französischen von Klaus Volkert.
- [14] Norbert Köckler Hans Rudolf Schwarz. *Numerische Mathematik*. 2011, Springer Verlag, 8. Auflage.
- [15] Daniel Scholz. *Numerik interaktiv, Grundlagen verstehen, Modelle erforschen und Verfahren anwenden mit taramath*. 2016, Springer Verlag.
- [16] Zhengyou Zhang. Determining the epipolar geometry and its uncertainty: A review. Received July 16, 1996; Accepted February 13, 1997.
- [17] Richard I. Hartley. In defence of the 8-point algorithm. GE-Corporate Research and Development, Schenectady, NY, 12309.
- [18] Cui Guodong Yu Ming Zhang Mandun, Qi Lichao. A triangulation method in 3d reconstruction from image sequences. College of Computer Science and Software, Hebei University of Technology Tianjin, China, 2009, Second International Conference on Intelligent Networks and Intelligent Systems.

- [19] MathWorks. Mathworks documentation, rectify stereo images. "<https://de.mathworks.com/help/vision/ref/rectifystereoimages.html>".
- [20] 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.
- [21] Carlos VILLAGRÁ ARNEDOr Antonio Javier GALLEGÓ SÁNCHEZ, Rafael MOLINA CARMONA. *Scene reconstruction and geometrical rectification from stereo images*. Januar 2005,uploaded by Antonio Javier Gallego Sánchez on 21 May 2014, ResearchGate.
- [22] Luca Irsara Andrea Fusiello. Euclidean epipolar rectification of uncalibrated images. Eurac researc, IT.
- [23] Richard I. Hartley. Euclidean reconstruction from uncalibrated views. G.E. CRD, Schenectady, NY, 12301.
- [24] Andrea Fusiello. Euclidean epipolar rectification. "http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/FUSIELLO2/rectif_cv01.html".
- [25] Dongqing Li, editor. *Encyclopedia of Microfluidics and Nanofluidics*, pages 999–999. Springer US, Boston, MA, 2008.
- [26] S. Margulies. Fitting experimental data using the method of least squares. Department of Physics, University of Illinois at Chicago Circle, Chicago, Illinois 60680, 1967.
- [27] 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.
- [28] James Demmel Dinesh Manocha. Algorithms for interesting parametric and algebraic curves 1: Simple intersections. ACM Transactions of Graphics:73–100, 1994.
- [29] MathWorks. Mathworks documentation,estimate camera parameters. "<https://de.mathworks.com/help/vision/ref/estimatecameraparameters.html>".
- [30] MathWorks. Mathworks documentation, disparity. "<https://de.mathworks.com/help/vision/ref/disparity.html>".
- [31] Canon. Eos 6d, eos 6d (wg), eos 6d (n), instruction manual, pages 188. "http://gdlp01.c-wss.com/gds/6/0300009626/01/EOS_6D_Instruction_Manual_DE.pdf".
- [32] 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.