Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Masterarbeit

Gregory Föll

Robuste Tiefenbildgewinnung aus Stereobildern

Fakultät Technik und Informatik
Department Informatik

Faculty of Engineering and Computer Science
Department of Computer Science

Gregory Föll

Robuste Tiefenbildgewinnung aus Stereobildern

Masterarbeit eingereicht im Rahmen der Masterprüfung
im Studiengang Informatik (Master)
am Department Informatik
der Fakultät Technik und Informatik
der Hochschule für Angewandte Wissenschaften Hamburg

Betreuender Prüfer : Prof. Dr.-Ing. Andreas Meisel
Zweitgutachter : Prof. Dr. rer. nat. Wolfgang Fohl

Abgegeben am 29. Juni 2012

Gregory Föll

Thema der Masterarbeit

Robuste Tiefenbildgewinnung aus Stereobildern

Stichworte

Stereoskopie, Korrespondenzproblem, dynamische Programmierung, Ground Control Point, GCP, SIFT, SURF, Delaunay-Triangulation, projektive Transformation

Kurzzusammenfassung

In dieser Arbeit wird ein robustes Verfahren zur Lösung des Stereokorrespondenzproblems vorgestellt. Das Verfahren basiert auf dem Verfahren der dynamischen Programmierung. Die Qualität der dynamischen Programmierung wird dabei durch den Einsatz von hochrobusten korrespondierenden Bildpunkten (GCP) erhöht.

Gregory Foell

Title of the paper

Robust range image acquisition from stereo images

Keywords

stereo vision, correspondence problem, dynamic programming, ground control point, GCP, SIFT, SURF, Delaunay triangulation, homography

Abstract

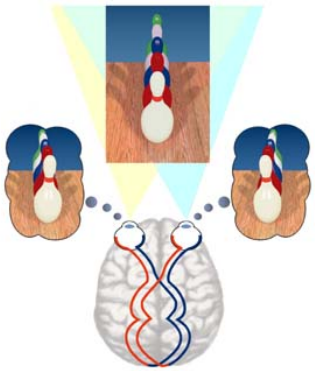
A robust method for solving the stereo matching problem is presented in this thesis. The method is based on dynamic programming. The dynamic programming stereo matching method is improved by using of ground control points (GCP).

Inhaltsverzeichnis

Tabellenverzeichnis

Abbildungsverzeichnis

1. Einführung



Stereoskopie oder Stereo Vision ist ein seit vielen Jahren erforschtes Gebiet der digitalen Bildverarbeitung. Das Ziel ist es, Tiefeninformationen aus mindestens zwei Bildern, die dasselbe Objekt zur selben Zeit aus unterschiedlichen Blickwinkeln zeigen, zu gewinnen. Durch die perspektivischen Differenzen in den beiden Bildern gelingt es, die Entfernung zu dem abgebildeten Objekt zu berechnen. Dieses Prinzip liegt auch dem menschlichen Sehapparat zugrunde, denn auch der Mensch nimmt seine Umgebung mit zwei Augen war, deren Abstand voneinander und deren Lage, die umliegenden Objekte aus zwei unterschiedlichen Perspektiven betrachten. Das Gehirn schätzt daraus die Entfernung zu dem betrachteten Objekt und seine räumliche Ausdehnung. Das Sehen mit zwei Augen oder Kameras wird als räumliches, dreidimensionales oder als Stereosehen bezeichnet.

Die Gewinnung von Tiefeninformationen aus mehreren zweidimensionalen Bildern ist nicht trivial. Viele Algorithmen ermöglichen keine Ausführung in Echtzeit oder setzen sehr leistungsstarke Hardware-Ressourcen voraus. Die rasante technologische Entwicklung bietet aber die Möglichkeit immer komplexere Verfahren zu entwickeln und schafft dadurch neue Einsatzgebiete.

Die Einsatzgebiete sind sehr vielfältig. Vor allem in der Robotik und bei der Navigation autonomer Fahrzeuge spielen stereoskopische Systeme eine zunehmend große Rolle. Das räumliche Sehen bringt für Roboter viele Vorteile. So können sie den Abstand zu den umliegenden Objekten berechnen und dadurch die Objekte genauer im Raum lokalisieren und greifen. Räumliches Sehen erweitert das Blickfeld. Verdrehte oder teilweise verdeckte Objekte könne leichter erkannt werden und auch die Navigation, die Hinderniserkennung und die Szenenerkennung werden dadurch enorm verbessert.

Im Rahmen dieser Arbeit wird ein Verfahren entwickelt und implementiert, welches eine robuste Extraktion von Tiefeninformationen aus Stereobildern ermöglicht. In den folgenden Kapiteln werden die Grundlagen der Stereoskopie und das entwickelte Verfahren genau beschrieben.

2. Grundlagen

In diesem Kapitel werden die für das Verständnis des vorgestellten Verfahrens benötigten Grundlagen erläutert. Die Theorie und die Problemstellung werden schrittweise verdeutlicht. Zunächst werden Grundlagen der Bildaufnahme, die Abbildungsgeometrie und die Kamerakalibrierung vorgestellt. Dann wird die Theorie der Stereoskopie mit den damit verbundenen Problemstellungen beschrieben. Zum Schluss werden grundlegende Ansätze zur Tiefenbildgewinnung aus Stereobildern kurz beschrieben.

2.1. Kameramodelle

2.1.1. Lochkamera

Das grundlegende Arbeitsprinzip einer Kamera kann sehr anschaulich anhand eines Lochkameramodells erläutert werden. Wie in Abbildung 2.1 zu sehen ist, stellt eine Lochkamera eine lichtdichte Box mit einem kleinen kreisförmigen Loch (Lochblende) an der Vorderseite dar. Die Mitte der Lochblende ist das sogenannte *Projektionszentrum* oder *optisches Zentrum* der Kamera, in dem sich auftreffende Projektionsstrahlen schneiden.

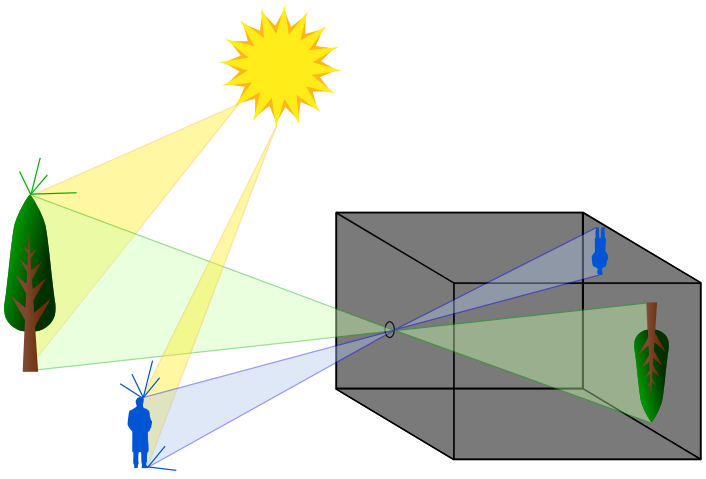


Abbildung 2.1.: Modell einer Lochkamera

Wird ein Objekt (in Abbildung 2.1 Baum und Person) von einer Lichtquelle (hier Sonne) beleuchtet, dann reflektiert das beleuchtete Objekt die Lichtstrahlen in viele verschiedene Richtungen. Trifft ein Teil der reflektierten Strahlen auf die Kamera, dann werden diese Strahlen von der winzigen Lochblende so geordnet, dass man auf der inneren Rückseite der Kamera eine Abbildung des beleuchteten Objektes erhält. Die zweidimensionale Abbildung zeigt das beleuchtete Objekt unter einem bestimmten Blickwinkel, denn da ein Objekt das auftreffende Licht in viele Richtungen reflektiert, kann es aus unterschiedlichen Blickwinkeln beobachtet werden. Das entstehende Bild lässt sich festhalten, indem man auf der inneren Rückseite der Kamera lichtempfindliches Material anbringt und es lange genug belichtet. Da das Projektionszentrum zwischen der betrachteten Szene und der Bildebene liegt, entsteht eine um $180°$ gedrehte Abbildung.

Bildgröße

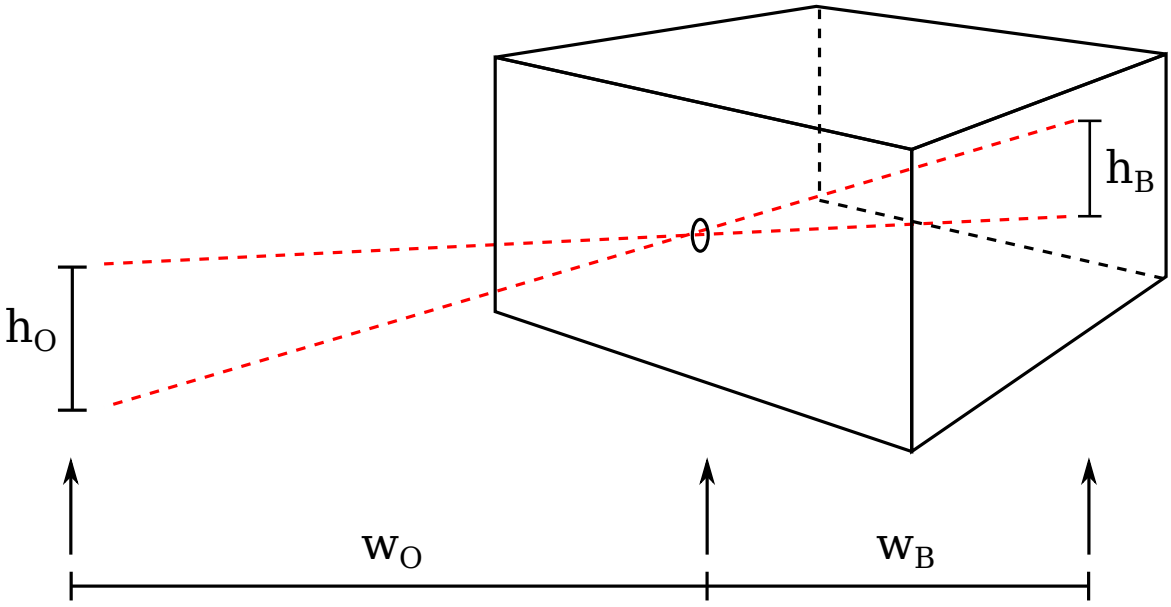


Abbildung 2.2.: Größe des mit Hilfe einer Lochkamera aufgenommenen Bildes.

Mit Hilfe des Strahlensatzes lässt sich zeigen, dass die Größe h_B der zweidimensionalen Abbildung eines Objektes im Raum von der Bildweite w_B (Abstand von der Lochblende zur Rückseite der Kamera) und von der Objektweite w_O (Abstand zwischen Lochblende und Objekt) abhängt.

$$\frac{h_B}{w_B} = \frac{h_O}{w_O} \Rightarrow h_B = \frac{h_O \cdot w_B}{w_O} \tag{2.1}$$

In Gleichung 2.1 bezeichnet h_O die Objektgröße.

2.1.2. Linsenkamera

Um ein scharfes Bild zu erzeugen, muss die Lochblende einer Lochkamera sehr klein gewählt werden. Im andern Fall treten im Bild sogenannte Zerstreuungskreise auf, die das Bild unscharf werden lassen.

Mit einer kleiner werdenden Lochblende wird die Kamera lichtschwächer. Das heißt, da nur wenig Licht durch das kleine Loch das lichtempfindliche Material auf der Rückseite der Kamera erreicht, verlängern sich die Belichtungszeiten. Durch Bewegungen des aufgenommenen Objektes während der Belichtung, entstehen Lichtspuren, das führt wieder zu unscharfen Bildern.

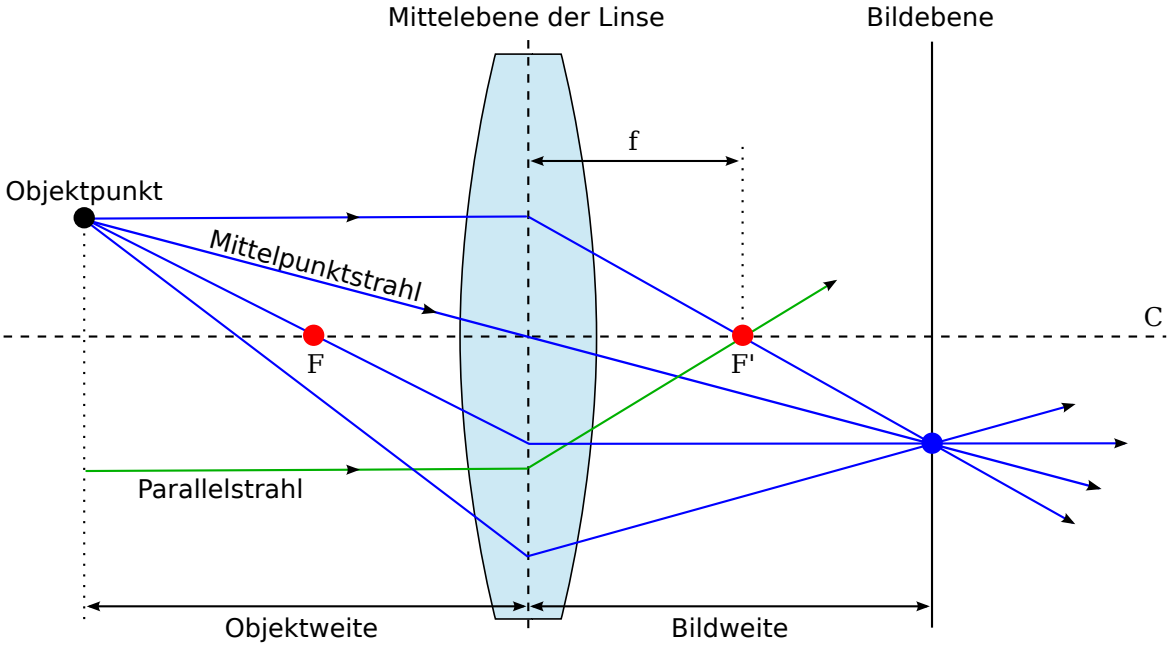


Abbildung 2.3.: Brechverhalten einer Sammellinse

Um die Lichtstärke der Kamera ohne Nebeneffekte (wie zum Beispiel Zerstreuungskreise) zu erhöhen, wird anstelle der Lochblende eine sogenannte Sammellinse eingesetzt. Eine Sammellinse lenkt alle parallel zur *optischen Achse* C einfallenden Lichtstrahlen so um, dass sie alle im Punkt F' gebündelt werden. Dieser Punkt wird als *Brennpunkt* oder *Fokus* bezeichnet. Der Abstand f zwischen der Mittelebene der Linse und dem Brennpunkt, bezeichnet man als *Brennweite* (Focal Length). Strahlen, die vor der Linse den Brennpunkt schneiden und dann auf die Linse treffen, werden so gebrochen, dass sie parallel zur optischen Achse verlaufen. Ein Strahl, der von einem Objektpunkt ausgeht und durch den Mittelpunkt der Linse verläuft, wird von der Linse nicht gebrochen. Dort, wo sich ein parallel zur optischen Achse verlaufender Strahl und der Mittelpunktstrahl eines Objektpunktes schneiden, schneiden sich

alle von diesem Objektpunkt reflektierten Strahlen. Dieser Schnittpunkt ist die Abbildung des Objektpunktes.

Die Lichtstärke der Kamera wird durch die größere Öffnung und durch die Lichtbündelung erhöht. Im Gegensatz zur Lochkamera, bei der die Bildschärfe unabhängig von der Objektweite ist, bildet die Linsenkamera nur die Objekte scharf ab, die sich auf einer bestimmten Entfernung zur Kamera befinden.

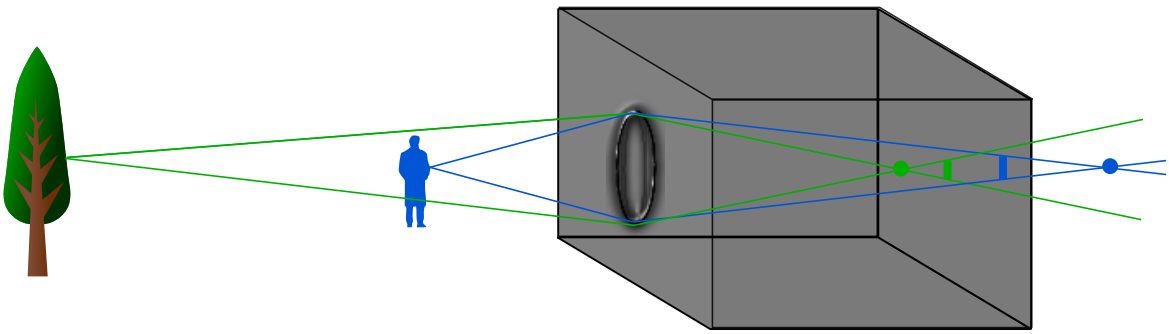


Abbildung 2.4.: Die Linsenkamera bildet zu weit oder zu nah liegende Objekte unscharf ab.

Befindet sich ein Objekt zu weit von der Kamera entfernt, dann schneiden sich die von ihm ausgehenden Lichtstrahlen noch vor der Bildebene. Die Projektionsstrahlen eines sehr nah liegenden Objektes haben ihren Schnittpunkt hinter der Bildebene (siehe Abbildung 2.4). In beiden Fällen entstehen im Bild anstelle von Punkten Kreise, die das Bild unscharf werden lassen. Linsenkameras haben meistens nicht nur eine Linse, sondern ein Objektiv das aus mehreren Linsen besteht. Mit einem Objektiv können Objekte scharf gestellt (fokussiert) werden, indem die Bildweite verändert wird.

2.2. Koordinatensysteme

Bei der Abbildung eines Raumpunktes durch eine Kamera sind mehrere Koordinatensysteme involviert. Man unterscheidet zwischen dem sogenannten Weltkoordinatensystem, dem Kamerakoordinatensystem und dem Bildkoordinatensystem. Die Beziehung zwischen diesen drei Koordinatensystemen ist in Abbildung 2.5 verdeutlicht.

Das 3D-Weltkoordinatensystem ist das Referenzsystem auf das man sich bezieht wenn Objektpositionen im Raum beschrieben werden. Es bezieht sich auf die betrachtete Szene, also auf im Raum liegende Objekte. Die Einheiten dieses Systems werden in m angegeben.

Das Kamerakoordinatensystem ist mit der aufnehmenden Kamera verknüpft. Ihre Z-Achse entspricht dabei der optischen Achse des Kamerasystems. Ihr Ursprung liegt im Projektionszentrum der Kamera.

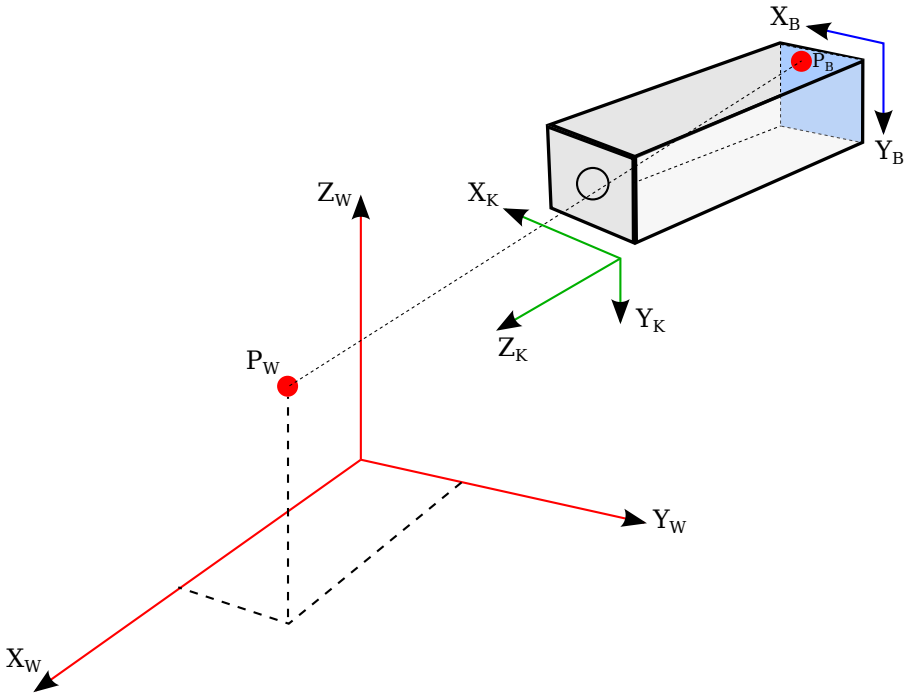


Abbildung 2.5.: Koordinatensysteme: Weltkoordinatensystem (rot), Kamerakoordinatensystem (grün), Bildkoordinatensystem (blau)

Das Bildkoordinatensystem ist zweidimensional. Seine x- und y-Achse verlaufen parallel zu den entsprechenden Achsen des Kamerakoordinatensystems. Das Bildkoordinatensystem beschreibt die Position eines abgebildeten Raumpunktes im Bild. Sein Ursprung liegt in der Mitte des Bildes, die Einheiten werden in m oder in μm angegeben.

Ein weiteres, für die Abbildungsgeometrie irrelevantes System, ist das Pixelkoordinatensystem. Es ist das Koordinatensystem eines digitalen Bildes, bei dem die Zeilen- und Spaltennummer eines Pixel als dessen Koordinaten verwendet werden. Der Ursprung dieses Koordinatensystems liegt in der linken oberen Ecke des Bildes. Die Einheit des Systems ist Pixel.

Die Beziehungen zwischen den genannten Systemen werden mit Hilfe von Matrizen beschrieben. Die Umrechnungsvorschriften werden im Abschnitt 2.3 vorgestellt.

2.3. Kamerakalibrierung

Das Modell einer Lochkamera liefert eine ideale Abbildung eines betrachteten Objektes. Das heißt, ein Objektpunkt wird ohne Verzerrungen in einen Bildpunkt überführt. Eine reale Kamera besitzt aber ein Objektiv, welches aus mehreren einzelnen Linsen besteht. Eine Linse kann einen Objektpunkt nicht fehlerfrei abbilden. Durch sogenannte *Linsenverzeichnungen* werden zum Beispiel gerade Linien gekrümmt dargestellt. Die Verzeichnung jeder Linse ist individuell. Das bedeutet, dass jedes Kamerasystem einen Objektpunkt auf eine unterschiedliche Art und Weise verzerrt abbildet. Diese Abbildungseigenschaften zusammen mit der Brennweite f und der genauen Position des Bildmittelpunktes, werden als *intrinsische* oder *innere* Kameraparameter bezeichnet.

Möchte man wissen wo sich ein Objektpunkt im Raum befindet, muss die relative Position und Orientierung der Kamera bezüglich eines vordefinierten Weltkoordinatensystems bekannt sein. Diese Größen bezeichnet man als *extrinsische* oder *äußere* Kameraparameter. Das Weltkoordinatensystem wird mit Hilfe eines bekannten Kalibrierkörpers bestimmt.

Der Vorgang bei dem die intrinsischen und extrinsischen Kameraparameter ermittelt werden, wird *Kalibrierung* genannt. Das Ziel der Kamerakalibrierung ist es, die Transformation zwischen dem 3D-Weltkoordinatensystem und dem 2D-Bildkoordinatensystem zu ermittelt. Die Kalibrierung wird im Folgenden genauer beschrieben.

2.3.1. Intrinsische Parameter

Die intrinsischen Parameter stellen den Zusammenhand zwischen Kamera- und Bildkoordinatensystem dar. Diese Parameter beschreiben die Eigenschaften des optischen Systems und bleiben deswegen immer gleich, auch wenn die Kamera bewegt wird. Sie müssen also nur einmal für jede Kamera bestimmt werden.

Eine Linse weist unterschiedliche Verzeichnungen auf. Verzeichnung bedeutet, dass die Lage eines Bildpunktes vom Abstand zum Bildmittelpunkt beeinflusst wird und sich dadurch nichtlinear zur Lage des zugehörigen Objektpunktes verhält. Der Bildmittelpunkt ist der Punkt, in dem die optische Achse die Bildebene schneidet.

Radiale Verzeichnung

Die radialsymmetrische Verzeichnung macht sich am deutlichsten im resultierenden Bild bemerkbar. Sie wirkt sich in Richtung des von der Bildmitte ausgehenden Vektors aus und führt zu kissenförmigen oder tonnenförmigen Verzerrungen. Durch Überlagerungen verschiedener Verzerrungsformen können auch wellenförmige Verzerrungen auftreten.

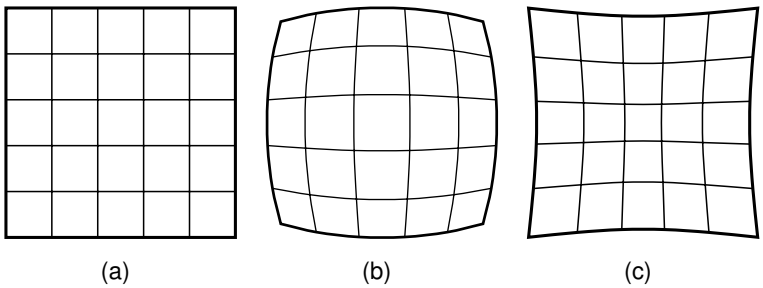


Abbildung 2.6.: Verzerrungsformen: (a) verzerrungsfrei (b) tonnenförmig (c) kissenförmig

Der Einfluss $R(\tilde{r})$ der radialen Verzeichnung wächst mit zunehmendem Abstand zum Bildmittelpunkt. $\tilde{r}$ steht für den Radius des verzerrten Punktes $\tilde{P}$ mit seinen verzerrten Koordinaten $\tilde{x}$ und $\tilde{y}$. r bezeichnet den Radius des unverzerrten Punktes P.

$$\tilde{r} = \sqrt{\tilde{x}^2 + \tilde{y}^2} \tag{2.2}$$

$$R(\tilde{r}) = 1 + k_1\tilde{r}^2 + k_2\tilde{r}^4 + k_3\tilde{r}^6 \tag{2.3}$$

In Gleichung 2.3 stehen k_1, k_2 und k_3 für die kameraabhängigen Verzerrungskoeffizienten.

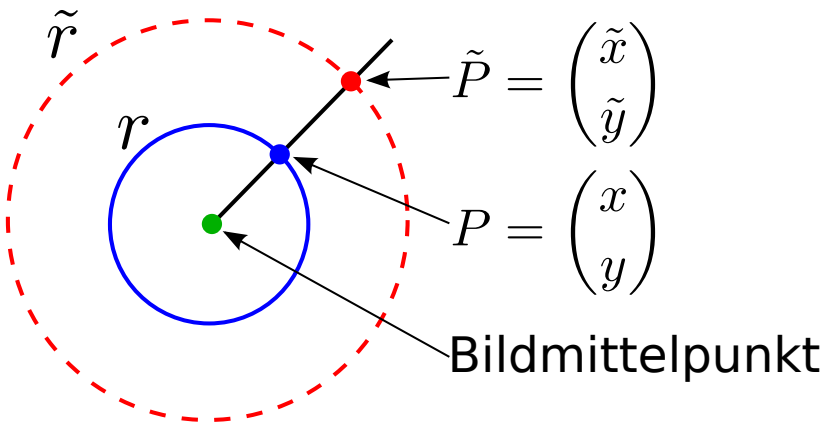


Abbildung 2.7.: Radiale Verzeichnung einer Linse

Tangentiale Verzeichnung

Die tangentiale Verzeichnung $T(\tilde{x}, \tilde{y})$ beeinflusst einen Bildpunkt in Richtung des Vektors, der tangential zu r verläuft.

$$T(\tilde{x}, \tilde{y}) = \begin{pmatrix} 2k_4\tilde{x}\tilde{y} + k_5(\tilde{r}^2 + 2\tilde{x}^2) \\ k_4(\tilde{r}^2 + 2\tilde{y}^2) + 2k_5\tilde{x}\tilde{y} \end{pmatrix} \tag{2.4}$$

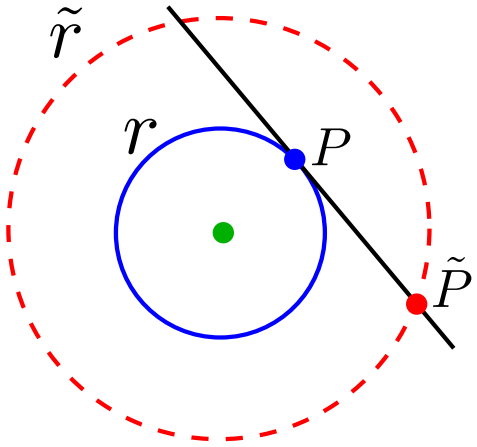


Abbildung 2.8.: Tangentiale Verzeichnung einer Linse. Diese Verzeichnung wird durch die Dezentralisierung der Linse im Objektiv verursacht.

Berechnung der entzerrten Bildpunktkoordinaten

Zur Verdeutlichung der Abbildungsgeometrie kann wieder das ideale Modell einer Lochkamera herangezogen werden. Mit $P_w \mapsto P_B$ beschreibt man die Abbildung eines Objektpunktes in einen Bildpunkt, also die Überführung eines Punktes aus $\mathbb{P}^3$ nach $\mathbb{P}^2$.

Unter Verwendung der Strahlensätze lässt sich zeigen, dass die Bildkoordinaten des Punktes P_B wie folgt berechnet werden:

$$x_B = f\frac{x_W}{z_W} \tag{2.5}$$

$$y_B = f\frac{y_W}{z_W} \tag{2.6}$$

Da die Lochkamera keine Linse besitzt und die Projektionsstrahlen somit nicht gebrochen werden, entspricht die Bildebene einer Lochkamera der Brennebene. Die Brennweite entspricht also dem Abstand von der Lochblende zur Bildebene.

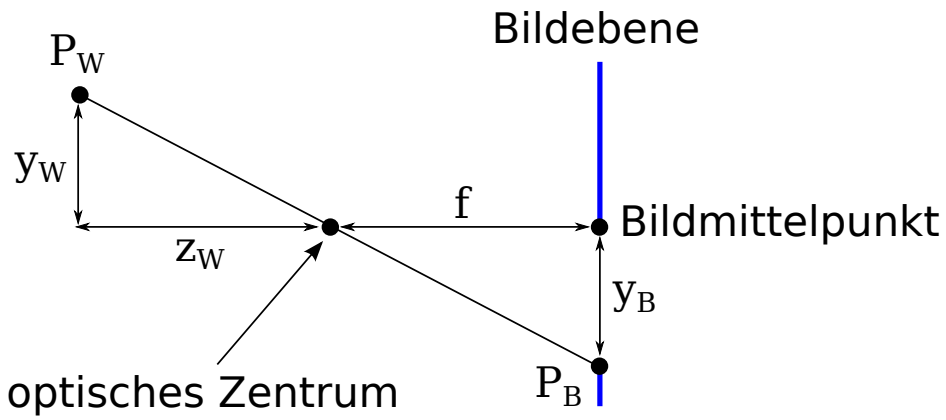


Abbildung 2.9.: Projektionsgeometrie einer Lochkamera

In homogenen Koordinaten gilt:

$$P_W = (x_W, y_W, z_W, 1)^T \tag{2.7}$$

$$P_B = \left(f\frac{x_W}{z_W}, f\frac{y_W}{z_W}, 1 \right)^T \tag{2.8}$$

Die Überführung kann auch in Matrixdarstellung geschrieben werden:

$$\begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} f\frac{x_W}{z_W} \\ f\frac{y_W}{z_W} \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix} \tag{2.9}$$

Das Modell einer Lochkamera beschreibt eine ideale Abbildungsgeometrie, bei der die optische Achse durch die Mitte der Bildebene verläuft. Ein reales Kamerasystem weist jedoch einen bestimmten Versatz zum Bildmittelpunkt auf. Der Vektor $(p_x, p_y, 1)^T$ beschreibt die homogenen Koordinaten des Bildmittelpunktes, das heißt ein abgebildeter Punkt wird beschrieben durch:

$$P_B = \left(f\frac{x_W}{z_W} + p_x, f\frac{y_W}{z_W} + p_y, 1 \right)^T \tag{2.10}$$

In Matrixdarstellung schreibt man dann:

$$\begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix} \mapsto \begin{pmatrix} f\frac{x_W}{z_W} + p_x \\ f\frac{y_W}{z_W} + p_y \\ 1 \end{pmatrix} = \begin{pmatrix} f & 0 & p_x & 0 \\ 0 & f & p_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix} \tag{2.11}$$

Aus dieser Betrachtung entsteht die sogenannte *Kalibrierungsmatrix* K.

$$K = \begin{pmatrix} f & 0 & p_x \\ 0 & f & p_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.12}$$

Bei der Überführung aus dem Bildkoordinatensystem in das Pixelkoordinatensystem, muss darauf geachtet werden, dass die Pixel auf dem Chip einer digitalen Kamera nicht immer quadratisch sein müssen. Das Seitenverhältnis eines Pixel $\frac{m_x}{m_y}$ muss also in die Matrix K einbezogen werden. Dabei wird auch die Brennweite f, bezogen auf die Pixelgröße, skaliert.

Die Achsen des Bildkoordinatensystems einiger Kameras verlaufen nicht orthogonal zueinander. Das bedeutet, der von den beiden Achsen eingeschlossene Winkel weicht von $90°$ ab. Dieser Fehler wird durch den Parameter s (*Skew*) ausgeglichen. Bei der Kalibrierung wird dieser Parameter aber meistens nicht berücksichtigt und somit auf 0 gesetzt.

Durch Einbeziehen der genannten Parameter ergibt sich eine Kalibrierungsmatrix, die die Abbildungsgeometrie des Kameramodells vollständig beschreibt.

$$K = \begin{pmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.13}$$

Mit $f_x = f \cdot m_x$, $f_y = f \cdot m_y$ und $c_x = p_x \cdot m_x$, $c_y = p_y \cdot m_y$.

Da die radiale und die tangentiale Verzerrungen keine linearen Größen sind, können sie nicht in die Kalibrierungsmatrix einbezogen werden. Aus diesem Grund werden zunächst die verzerrten Koordinaten eines Bildpunktes korrigiert. Das Ergebnis wird dann mit der Kalibrierungsmatrix K multipliziert. Daraus resultieren schließlich die entzerrten Bildkoordinaten x und y.

$$\begin{pmatrix} x \\ y \end{pmatrix} = K \left(R(\tilde{r}) \cdot \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} + T(\tilde{x}, \tilde{y}) \right) \tag{2.14}$$

2.3.2. Extrinsische Parameter

Wie in Abbildung 2.5 zu sehen ist, befindet sich die Kamera an einer bestimmten Position im Weltkoordinatensystem. Die Matrix 2.12 kann nur dann angewendet werden, wenn das Welt- und das Kamerakoordinatensystem übereinstimmen, was in der Regel nicht der Fall ist. Die beiden Koordinatensysteme lassen sich durch Translation und Rotation ineinander überführen (siehe Abbildung 2.10).

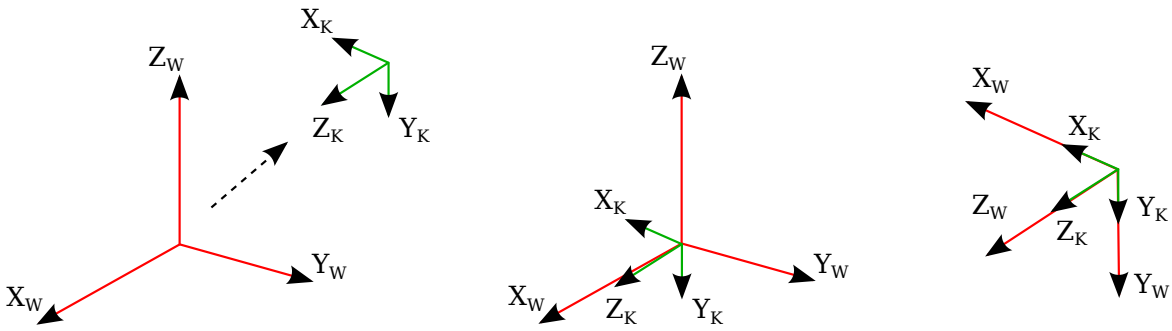


Abbildung 2.10.: Überführung des Kamerakoordinatensystems in das Weltkoordinatensystem durch Translation und Rotation.

Zuerst wird der Ursprung des Weltkoordinatensystems durch den Translationsvektor t in den Ursprung des Kamerakoordinatensystems verschoben. Anschließend wird das Weltkoordinatensystem unter Verwendung der Rotationsmatrix R so gedreht, dass es mit dem Kamerakoordinatensystem zusammenfällt.

Die Weltkoordinaten eines 3D-Punktes $P_W = (x_W, y_W, z_W)^T$ werden in Kamerakoordinaten desselben Punktes $P_K = (x_K, y_K, z_K)^T$ überführt mit:

$$P_K = R \cdot P_W + t = \begin{pmatrix} x_K \\ y_K \\ z_K \\ 1 \end{pmatrix} = \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix} \tag{2.15}$$

Die gesamte Abbildung vom Weltkoordinatensystem in das Bildkoordinatensystem lässt sich mit der Projektionsmatrix P beschreiben.

$$P = K \begin{pmatrix} R & t \\ 0 & 1 \end{pmatrix} \tag{2.16}$$

Die Koordinaten des $3x1$ Translationsvektors $t = (t_x, t_y, t_z)^T$ geben die parallele Verschie-

bung des Weltkoordinatensystems zum Kamerakoordinatensystem entlang der entsprechenden Achsen an.

Die Rotation des verschobenen Weltkoordinatensystems wird in drei Schritten durchgeführt. Es wird um die drei positiven Achsen gedreht. Die Winkel entgegen dem Uhrzeigersinn werden als positiv betrachtet.

1. Rotation um die x-Achse um den Winkel α:

$$R(\alpha) = \begin{pmatrix} 1 & 0 & 0 \\ 0 & cos(\alpha) & sin(\alpha) \\ 0 & -sin(\alpha) & cos(\alpha) \end{pmatrix} \tag{2.17}$$

2. Rotation um die y-Achse um den Winkel β:

$$R(\beta) = \begin{pmatrix} cos(\beta) & 0 & -sin(\beta) \\ 0 & 1 & 0 \\ sin(\beta) & 0 & cos(\beta) \end{pmatrix} \tag{2.18}$$

3. Rotation um die z-Achse um den Winkel γ:

$$R(\gamma) = \begin{pmatrix} cos(\gamma) & sind(\gamma) & 0 \\ -sin(\gamma) & cos(\gamma) & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.19}$$

Insgesamt ergibt sich aus den drei Rotationen $R(\alpha)$, $R(\beta)$, und $R(\gamma)$ die $3x3$ Rotationsmatrix R.

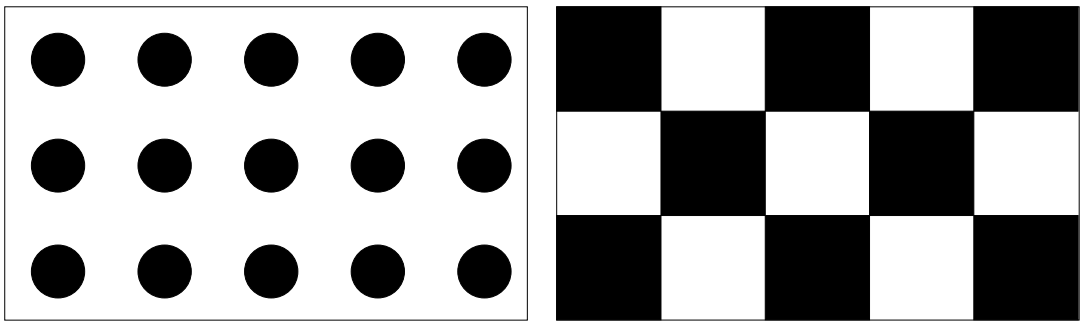
$$R = \begin{pmatrix} cos(\beta)cos(\gamma) & cos(\alpha)sin(\gamma)+sin(\alpha)sin(\beta)cos(\gamma) & sin(\alpha)sin(\gamma)-cos(\alpha)sin(\beta)cos(\gamma) \\ -cos(\beta)sin(\gamma) & cos(\alpha)cos(\gamma)-sin(\alpha)sin(\beta)sin(\gamma) & sin(\alpha)cos(\gamma)-cos(\alpha)sin(\beta)sin(\gamma) \\ sin(\beta) & -sin(\alpha)cos(\beta) & cos(\alpha)cos(\beta) \end{pmatrix} \tag{2.20}$$

Unter Beachtung der intrinsischen und extrinsischen Kameraparameter (Gleichung 2.16) gilt für die Überführung eines 3D-Objektpunktes in einen 2D-Bildpunkt folgender Zusammenhang:

$$\begin{pmatrix} x_B \\ y_B \\ 1 \end{pmatrix} = \underbrace{\begin{pmatrix} L_{11} & L_{12} & L_{13} & L_{14} \\ L_{21} & L_{22} & L_{23} & L_{24} \\ L_{31} & L_{32} & L_{33} & L_{34} \\ 0 & 0 & 0 & 1 \end{pmatrix}}_{Projektionsmatrix P} \begin{pmatrix} x_W \\ y_W \\ z_W \\ 1 \end{pmatrix} \tag{2.21}$$

Die Projektionsmatrix P aus Gleichung 2.21 enthält 12 Unbekannte. Betrachtet man diese Unbekannten als ein lineares Gleichungssystem, dann hat das Gleichungssystem einen Freiheitsgrad von 11. Das heißt man kann den Parameter L_{34} zum Beispiel auf 1 setzen und die anderen 11 frei wählen. Man erhält danach 11 Unbekannte ($L_{11} \ldots L_{33}$). Für die Lösung werden also mindestens 11 Gleichungen benötigt.

Die unbekannten Parameter werden mit Hilfe eines Kalibriermusters bestimmt. Es handelt sich dabei um ein Muster mit genau bekannten Abmessungen. Das Muster enthält markante Punkte. Nach der Aufnahme des Kalibriermusters werden zu diesen Punkten im Weltkoordinatensystem ihre korrespondierenden Bildpunkte bestimmt. Üblicherweise werden Schachbrettmuster oder Kreismuster verwendet (siehe Abbildung 2.11), Schachbrettmuster werden in der Regel bevorzugt, da die Ecken des Musters sehr effizient mit Hilfe von Kantenfiltern bestimmt werden können und somit als markante Kalibrierpunkte dienen.



(a) (b)

Abbildung 2.11.: Übliche Muster für die Kamerakalibrierung: (a) Kreismuster (b) Schachbrettmuster

Anstelle eines Musters kann auch ein dreidimensionales Objekt mit bekannten Weltkoordinaten für die Kalibrierung benutzt werden ((siehe Abbildung 2.12). Ein Kalibrierobjekt enthält jedoch weitaus weniger markante Kalibrierpunkte als beispielsweise ein Schachbrettmuster.

Gemäß Gleichung 2.21 ergeben sich für jedes korrespondierende Punktepaar ($P_W \leftrightarrow P_B$) jeweils zwei Gleichungen:

$$x_B = \frac{L_{11}x_W + L_{12}y_W + L_{13}z_W + L_{14}}{L_{31}x_W + L_{32}y_W + L_{33}z_W + 1} \tag{2.22}$$

$$y_B = \frac{L_{21}x_W + L_{22}y_W + L_{23}z_W + L_{24}}{L_{31}x_W + L_{32}y_W + L_{33}z_W + 1} \tag{2.23}$$

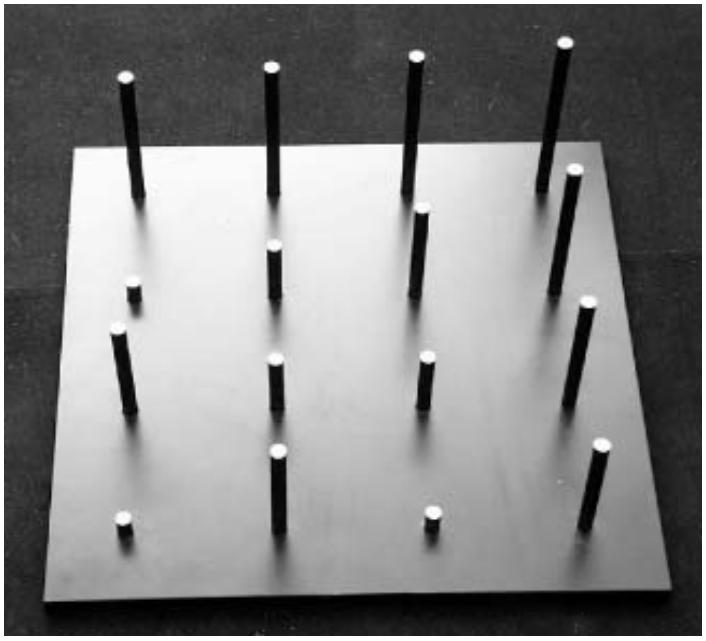


Abbildung 2.12.: Kalibrierkörper für die Kamerakalibrierung mir bekannten Weltkoordinaten

Multipliziert man die beiden Gleichungen mit dem Nenner und stellt sie nach x_B und y_B um, erhält man die folgenden zwei Gleichungen:

$$x_B = L_{11}x_W + L_{12}y_W + L_{13}z_W + L_{14} - L_{31}x_Bx_W - L_{32}x_By_W - L_{33}x_Bz_W \tag{2.24}$$

$$y_B = L_{21}x_W + L_{22}y_W + L_{23}z_W + L_{24} - L_{31}y_Bx_W - L_{32}y_By_W - L_{33}y_Bz_W \tag{2.25}$$

Theoretisch reichen bereits 6 Punktkorrespondenzen ($\Rightarrow$ 12 Gleichungen) für die Lösung des Gleichungssystems mit den 11 Unbekannten Parametern aus. Da bei einem realen Kamerasystem, zum Beispiel durch Bildrauschen, Messungenauigkeiten entstehen, werden in der Praxis mehr Punktkorrespondenzen benötigt. Dazu wird das Kalibriermuster mehrmals aus unterschiedlichen Perspektiven aufgenommen. Eine höhere Anzahl von Punktkorrespondenzen führt zu einem überbestimmten linearen Gleichungssystem und zu einem genaueren Ergebnis.

Mit n Punktkorrespondenzen erhält man unter Verwendung der Gleichungen 2.24 und 2.25 das folgende lineare Gleichungssystem:

$$\underbrace{\begin{pmatrix} x_{W_1} & y_{W_1} & z_{W_1} & 1 & 0 & 0 & 0 & 0 & -x_{B_1}x_{W_1} & -x_{B_1}y_{W_1} & -x_{B_1}z_{W_1} \\ 0 & 0 & 0 & 0 & x_{W_1} & y_{W_1} & z_{W_1} & 1 & -y_{B_1}x_{W_1} & -y_{B_1}y_{W_1} & -y_{B_1}z_{W_1} \\ x_{W_2} & y_{W_2} & z_{W_2} & 1 & 0 & 0 & 0 & 0 & -x_{B_2}x_{W_2} & -x_{B_2}y_{W_2} & -x_{B_2}z_{W_2} \\ 0 & 0 & 0 & 0 & x_{W_2} & y_{W_2} & z_{W_2} & 1 & -y_{B_2}x_{W_2} & -y_{B_2}y_{W_2} & -y_{B_2}z_{W_2} \\ . & . & . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . & . & . \\ x_{W_n} & y_{W_n} & z_{W_n} & 1 & 0 & 0 & 0 & 0 & -x_{B_n}x_{W_n} & -x_{B_n}y_{W_n} & -x_{B_n}z_{W_n} \\ 0 & 0 & 0 & 0 & x_{W_n} & y_{W_n} & z_{W_n} & 1 & -y_{B_n}x_{W_n} & -y_{B_n}y_{W_n} & -y_{B_n}z_{W_n} \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} L_{11} \\ L_{12} \\ L_{13} \\ L_{14} \\ L_{21} \\ L_{22} \\ L_{23} \\ L_{24} \\ L_{31} \\ L_{32} \\ L_{33} \end{pmatrix}}_{T^*} = \underbrace{\begin{pmatrix} x_{B_1} \\ y_{B_1} \\ x_{B_2} \\ y_{B_2} \\ . \\ . \\ . \\ x_{B_n} \\ y_{B_n} \end{pmatrix}}_{D} \tag{2.26}$$

Das überbestimmte lineare Gleichungssystem

$$A \cdot T^* = D \tag{2.27}$$

lässt sich mit Hilfe der Methode der kleinsten Quadrate (*Least Squares*) nach Gauß lösen. Da durch die Messungenauigkeiten das Gleichungssystem nicht exakt lösbar ist, sucht man nach einer Lösung, bei der der quadratische Fehler am kleinsten ist. Die Lösung lautet dann:

$$T^* = (A^T A)^{-1} A^T D \tag{2.28}$$

Die Lösung des Gleichungssystems führt zur Bestimmung aller intrinsischen und extrinsischen Kameraparametern.

2.3.3. Kamerakalibrierung nach Tsai

Für die Kalibrierung von Kamerasystemen existieren mehrere Verfahren [González u. a. (2007)]. In der Praxis wird oft das Verfahren von Roger Tsai [Tsai (1992)] eingesetzt.

Tsai unterscheidet zwischen 6 extrinsischen und 5 intrinsischen Kameraparametern. Zu den extrinsischen Parametern gehören die drei Rotationswinkel (siehe Gleichungen 2.17, 2.18 und 2.19) und die Koordinaten des Translationsvektors t.

Zu den intrinsischen Parametern gehören die Brennweite f, die radialen Verzerrungskoeffizienten k_n (die tangentiale Verzerrung wird wegen ihren geringen Einfluss vernachlässigt), der *Skew*-Parameter s und die beiden Koordinaten des Bildmittelpunktes c_x und c_y.

Das Verfahren wird in vier Schritten durchgeführt:

1. Transformation des Weltkoordinatensystem in das Kamerakoordinatensystem (siehe Gleichung 2.15). Gesucht sind hier die Parameter R und t.
2. Transformation der Kamerakoordinaten in Bildkoordinaten, ohne Berücksichtigung von Linsenverzeichnungen (da das Welt- und das Kamerakoordinatensystem jetzt übereinstimmen, gelten die Gleichungen 2.5 und 2.6). Gesucht ist der Parameter f.
3. Berücksichtigung der radialen Linsenverzeichnung (siehe Gleichungen 2.2, 2.3 und 2.14). Gesucht sind die radialen Verzerrungskoeffizienten k_1, k_2, und k_3.
4. Transformation der entzerrten Bildkoordinaten in Pixelkoordinaten (siehe Matrix 2.13). Gesucht ist der Parameter s.

2.4. Stereoskopie

Bei der Stereoskopie (Stereo Vision) wird eine Szene gleichzeitig aus verschiedenen Perspektiven aufgenommen. Für eine Stereoaufnahme benötigt man also mindestens zwei Kameras. Anhand der Unterschiede (Disparitäten) in den einzelnen zweidimensionalen Aufnahmen, kann ein Tiefenbild der betrachteten Szene berechnet werden.

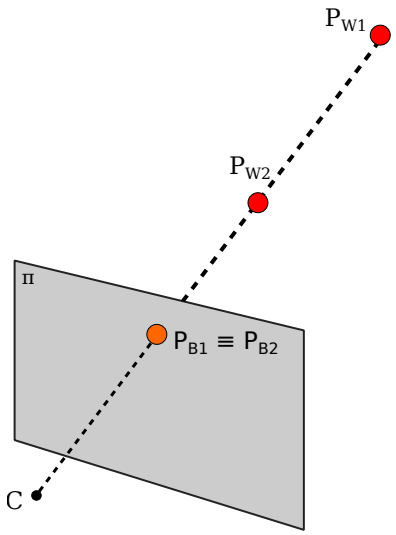


Abbildung 2.13.: Abbildung bei der Aufnahme einer Szene mit einer Kamera

In Abbildung 2.13 liegen die beiden Weltpunkte P_{W1} und P_{W2} auf einer Projektionsgeraden. Da P_{W1} durch P_{W2} verdeckt wird, entsteht auf der Bildebene π für die beiden Weltpunkte nur ein Bildpunkt. Dieser Bildpunkt bildet nur den Punkt P_{W2} ab.

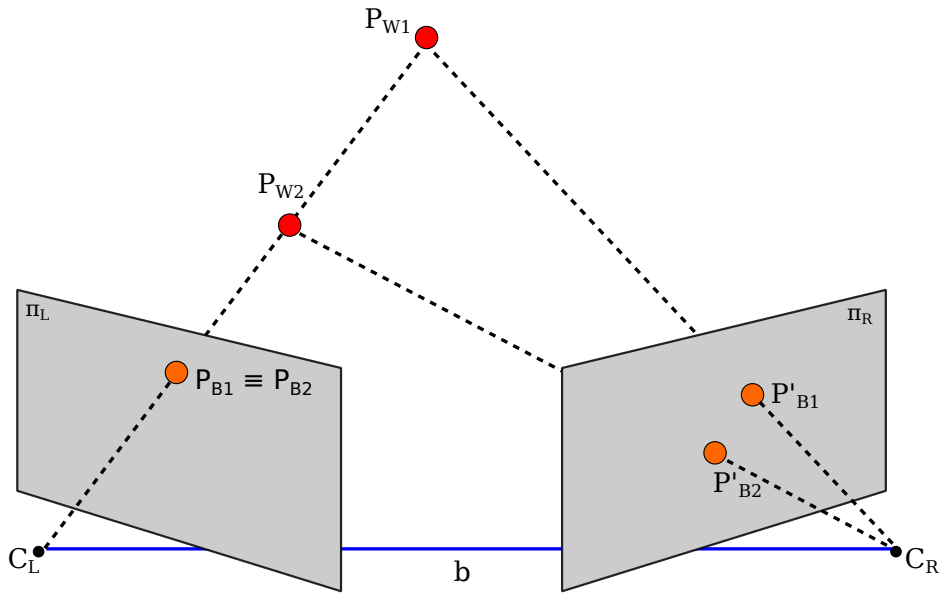


Abbildung 2.14.: Abbildung bei der Aufnahme einer Szene mit zwei Kameras

Werden zwei Kameras für die Aufnahme einer Szene verwendet, die durch eine Basislinie b voneinander getrennt sind, kann durch die Berechnung des Dreiecks (Triangulation), das durch die Punkte P_{W2}, P_{B2} und P'_{B2} gebildet wird, die Entfernung zum Punkt P_{W2} bestimmt werden. Für die Berechnung müssen, außer der Bildkoordinaten der korrespondierenden Punkte P_{B2} und P'_{B2}, die Kameraparameter f (Brennweite) und b (Basislinie) bekannt sein. Die Berechnung wird in Abschnitt 2.4.4 genau erklärt.

2.4.1. Korrespondenzproblem

Die für die Triangulation benötigten Kameraparameter f und b werden bei der Kalibrierung des Kamerasystems ermittelt. Das große Problem der Stereoskopie ist es, zu einem Punkt eines Bildes einer Stereoaufnahme, die korrespondierenden Punkte in den anderen Bildern dieser Stereoaufnahme zu finden.

Durchsucht man das gesamte korrespondierende Bild nach ähnlichen Punkten, ergibt sich ein zweidimensionaler Suchraum. Die Korrespondenzsuche in einem 2D-Raum ist sehr komplex, da es dabei mit großer Wahrscheinlichkeit zu Mehrdeutigkeiten kommt. Unter Verwendung der *Epipolargeometrie* (siehe nächster Abschnitt), kann die Suche auf einen eindimen-

sionalen Raum beschränkt werden. Dadurch wird das Korrespondenzproblem stark vereinfacht.

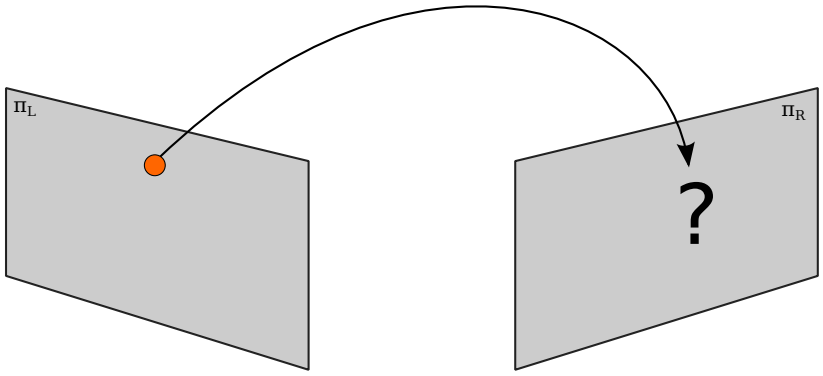


Abbildung 2.15.: Korrespondenzproblem

2.4.2. Epipolargeometrie

Die Epipolargeometrie beschreibt die Zusammenhänge zwischen den Aufnahmen einer Szene, die von mindestens zwei Kameras zur selben Zeit aufgenommen werden.

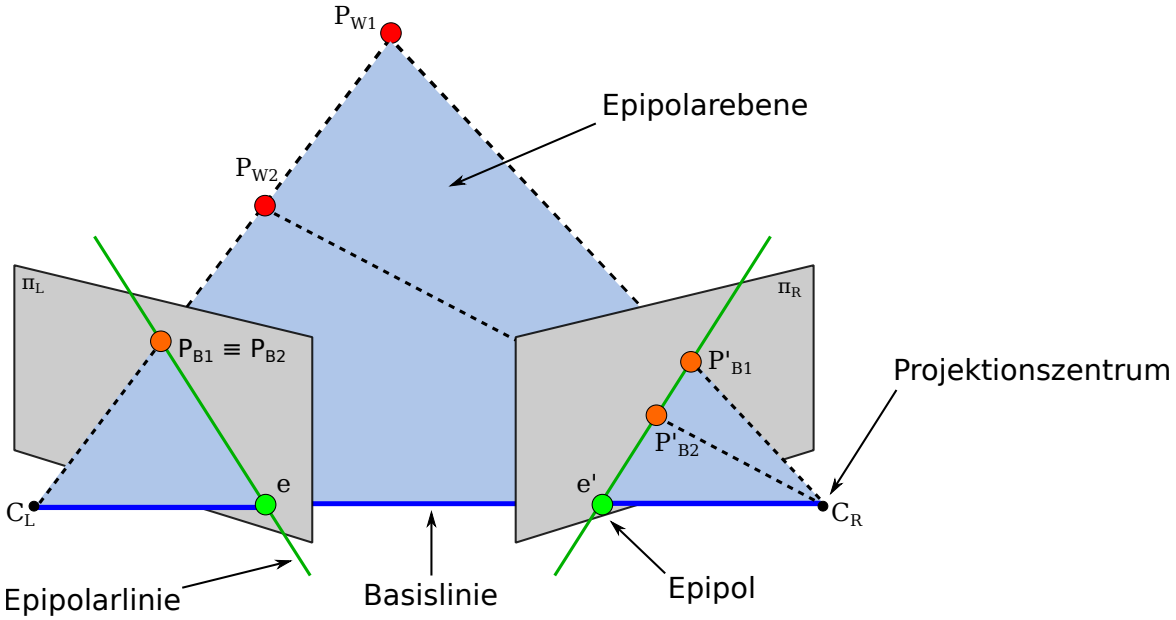


Abbildung 2.16.: Epipolargeometrie

Abbildung 2.16 zeigt die Epipolargeometrie für die Aufnahme mit zwei Kameras. Die Projektionszentren, C_L und C_R, der beiden Kameras und ein Weltpunkt, zum Beispiel der Punkt

P_{W1}, spannen die sogenannte *Epipolarebene* auf. Die Punkte e und e', in denen die Basislinie d die Bildebenen durchstößt, werden als *Epipole* bezeichnet. Geraden, die durch die Schnittpunkte der Epipolarebene mit der jeweiligen Bildebene verlaufen, bezeichnet man als *Epipolarlinien*.

Jeder Punkt innerhalb der Epipolarebene, wird auf dieselben Epipolarlinien in den Bildebenen π_L und π_R projiziert. Die Abbildung der Geraden, die durch die Punkte C_L, P_{B1}, P_{B2}, P_{W2} und P_{W1} verläuft, bestimmt in der Bildebene π_R der rechten Kamera die Epipolarlinie, die die korrespondierenden Bildpunkte P'_{B1} und P'_{B2} schneidet. Diese Eigenschaft wird als *Epipolarbedingung* bezeichnet.

Das Koordinatensystem der einen Kamera kann durch eine Translation und eine Rotation in das Koordinatensystem der anderen Kamera überführt werden. Der Translationsvektor t entspricht in diesem Fall der Basislinie b. Die Beziehung zwischen den beiden Kameras, also die Epipolargeometrie, wird mit Hilfe der *Essential-Matrix* und der *Fundamental-Matrix* modelliert.

Die Essential- und die Fundamental-Matrix

Die $3x3$ Essential-Matrix E beinhaltet die extrinsischen Parameter des aufnehmenden Kamerasystems. Sie modelliert die Epipolarbedingung, indem sie einen Bildpunkt P_B der einen Kamera auf die zugehörige Epipolarlinie l' der anderen Kamera abbildet und umgekehrt:

$$l' = EP_B \Longleftrightarrow l = E^T P'_B \Rightarrow {P'_B}^T E P_B = 0 \tag{2.29}$$

Die Fundamental-Matrix F enthält außer der extrinsischen auch die intrinsischen Kameraparameter. Sie kann somit als Erweiterung der Essential-Matrix angesehen werden. Sie modelliert ebenfalls die Epipolarbedingung:

$$l' = FP_B \Longleftrightarrow l = F^T P'_B \Rightarrow {P'_B}^T F P_B = 0 \tag{2.30}$$

Auch für die Fundamental-Matrix gilt: $F \in \mathbb{R}^{3x3}$. Im Gegensatz zur Matrix E bezieht sich F nicht auf Bild- sondern auf Pixelkoordinaten. Das hat den Vorteil, dass falls genügend robuste Punktkorrespondenzen in den Bildern gefunden werden können, die Fundamental-Matrix durch Lösen eines linearen Gleichungssystems bestimmt werden kann. Dazu werden keine Informationen über intrinsische oder extrinsische Kameraparameter benötigt. Für ${P'_B}^T F P_B = 0$ schreibt man unter Verwendung homogener Koordinaten:

$$(x', y', 1) \cdot \begin{pmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{pmatrix} \cdot \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = 0 \tag{2.31}$$

Nach dem Ausmultiplizieren von 2.31 bekommt man die Gleichung 2.33, die für jedes korrespondierende Punktepaar aufgestellt wird:

$$(x', y', 1) \cdot \begin{pmatrix} f_{11} \cdot x + f_{12} \cdot y + f_{13} \\ f_{21} \cdot x + f_{22} \cdot y + f_{23} \\ f_{31} \cdot x + f_{32} \cdot y + f_{33} \end{pmatrix} = 0 \tag{2.32}$$

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} + = 0 \tag{2.33}$$

Zur Bestimmung von F werden mindestens 8 Punktkorrespondenzen benötigt. Für n korrespondierende Punkte ergibt sich das folgende lineare, homogene Gleichungssystem:

$$\underbrace{\begin{pmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . & . \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{pmatrix}}_{A} \cdot \underbrace{\begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix}}_{f} = 0 \tag{2.34}$$

Das Aufstellen und Lösen dieses Gleichungssystems wird als *8-Punkt-Algorithmus* bezeichnet.

Durch Ungenauigkeiten in den Punktkorrespondenzen (verursacht zum Beispiel durch Bildrauschen) werden mehr als 8 Korrespondenzen für die Lösung benötigt. Das führt zu einem überbestimmten Gleichungssystem. Um in diesem Fall das Gleichungssystem trotzdem lösen zu können, wird die *Singulärwertzerlegung* (SVD) eingesetzt (siehe Anhang B). Eine Randbedingung der F-Matrix ist, dass sie nach dem Ausrechnen einen Rang von 2 aufweist. Um diese Eigenschaft sicherzustellen, wird wieder die Singulärwertzerlegung verwendet.

Ein weiterer Nachteil des 8-Punkt-Algorithmus ist die numerische Instabilität. Da der Algorithmus mit Bildpunktkorrespondenzen arbeitet und da die Bildkoordinaten korrespondierender Punkte, bei großen Bildern, weit auseinander liegen können, weichen die Einträge der Matrix A, in ihrer Größenordnung stark voneinander ab. Um die Einträge von A einander anzupassen, können die Bildkoordinaten der Punktkorrespondenzen durch eine Translation und eine Skalierung normalisiert werden.

Bei fehlerbehafteten Punktkorrespondenzen kann die Fundamental-Matrix mit Hilfe des *RANSAC*-Verfahrens (siehe Abschnitt 3.1.3) approximiert werden. Dabei wird wie folgt vorgegangen:

1. Suche mit einem geeigneten Verfahren nach Punktkorrespondenzen.
2. Teile die gefundenen Korrespondenzen in n Mengen auf, wobei jede Menge aus 8 Korrespondenzen besteht.
3. Berechne mit allen Korrespondenzpaaren jeder der n Mengen die F-Matrix.
4. Berechne für einen Punkt jedes Korrespondenzpaares die zugehörige Epipolarlinie im korrespondierenden Bild und prüfe, ob der andere Punkt auf dieser Linie liegt. Lasse dabei für den Fahler einen bestimmten Schwellwert zu.
5. Berechne F mit allen Punktkorrespondenzen die den Test in Punkt 4 bestanden haben.

Aus einer bekannten Fundamental-Matrix können der Translationsvektor und die Rotationsmatrix der Kameras zueinander, die Essential-Matrix und die Projektionsmatrizen der einzelnen Kameras berechnet werden. Für eine detaillierte Einführung in die Epipolargeometrie und die Berechnung der F-Matrix siehe [Hartley und Zisserman (2003)].

2.4.3. Rektifizierung

Die Epipolargeometrie vereinfacht enorm die Suche nach korrespondierenden Punkten in gleichzeitig aufgenommenen Bilder einer Szene, indem sie den Suchraum von 2D auf 1D reduziert. Durch eine Rektifizierung der Bilder kann die Suche weiter vereinfacht werden.

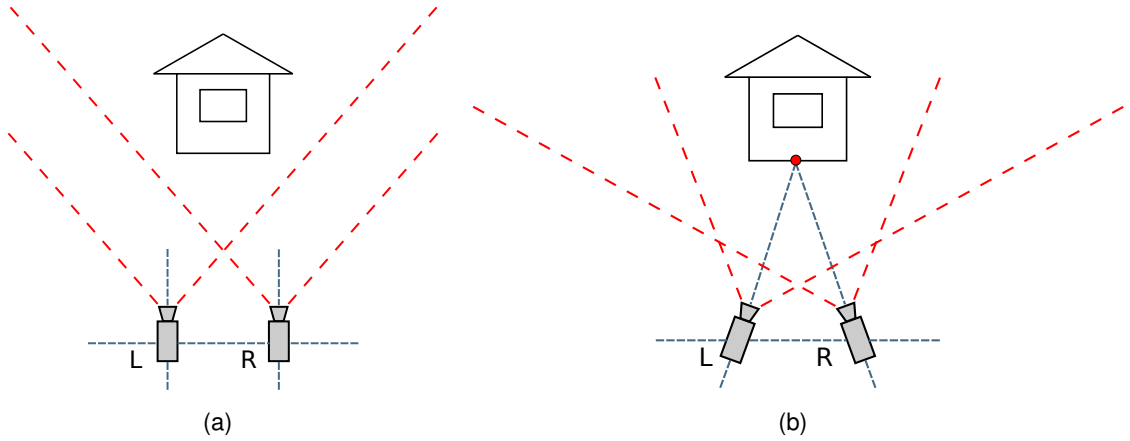


Abbildung 2.17.: Positionierung der Kameras in einem Stereokamerasystem: (a) parallel (b) zueinander gedreht

Üblicherweise sind die Kameras eines Stereokamerasystems so angeordnet, dass ihre optischen Achsen sich in einem Weltpunkt schneiden (siehe Abbildung 2.17(b)). Die Kameras sind also nicht parallel ausgerichtet, sonder sind zueinander gedreht. Aus diesem Grund haben die Epipolarlinien der einzelnen Bilder einen schrägen Verlauf (siehe Abbildung 2.16). Bei parallel ausgerichteten Kameras verlaufen die Epipolarlinien horizontal (siehe Abbildung 2.18). Das vereinfacht die Korrespondenzsuche, da die Suche auf nur eine Bildzeile beschränkt wird. In der Praxis ist es schwierig die Kameras exakt parallel zu positionieren. Außerdem ergibt sich dadurch ein kleinerer Ausschnitt der betrachtete Szene, der von beiden Kameras erfasst wird (vergleiche die Abbildungen 2.17(a) und 2.17(b)).

Sind die Projektionsmatrizen aller beteiligten Kameras bekannt, dann können die aufgenommenen Bilder rektifiziert werden. Die Rektifizierung entspricht einer virtuellen Drehung der Kameras nach der alle Kameras parallel zueinander sind und in derselben Ebene liegen. Mit Hilfe der Projektionsmatrizen werden bei der Rektifizierung auch die Linsenverzerrungen der Kameras ausgeglichen. Nach der Rektifizierung sind die Bilder in der sogenannten *Standardform* oder *kanonischen Form*.

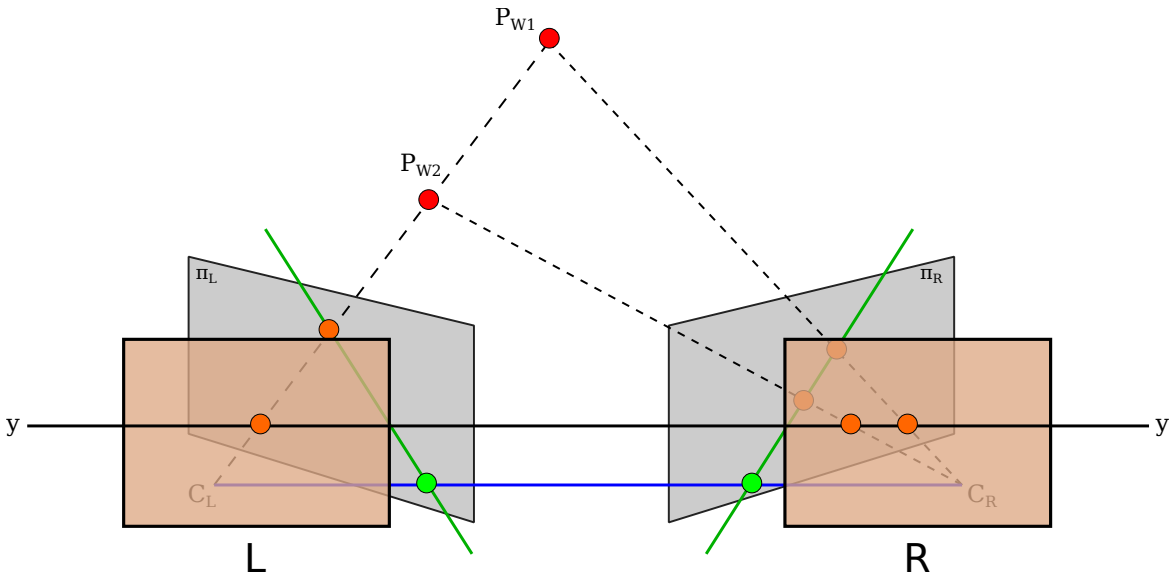


Abbildung 2.18.: Nach der Rektifizierung liegen korrespondierende Punkte in allen Bildern in der gleichen Bildzeile y.

2.4.4. Disparität und Abstand

Befinden sich die aufgenommenen Bilder in Standardform und sind korrespondierende Punkte bereits gefunden, können mit Hilfe der Triangulation aus den zweidimensionalen Bildern Tiefeninformationen gewonnen werden.

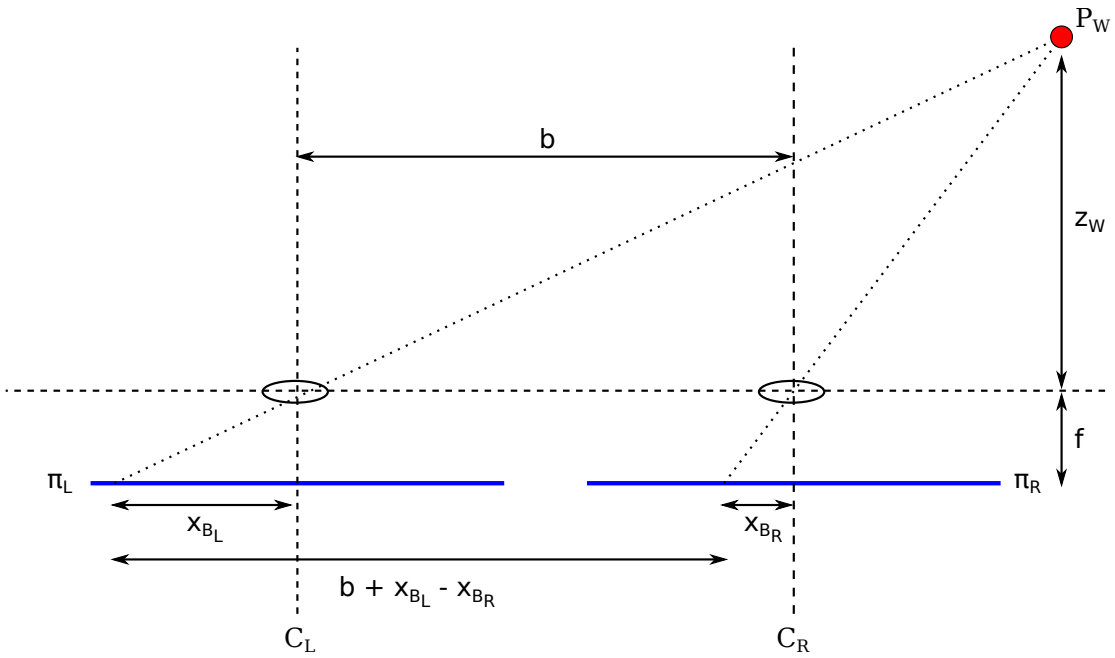


Abbildung 2.19.: Messen des Abstandes z_W mittels Triangulation

Unter Verwendung der Strahlensätze und mit $x_{B_L} - x_{B_R} = d$ gilt:

$$\frac{z_W}{z_W + f} = \frac{b}{b + x_{B_L} - x_{B_R}} \tag{2.35}$$

$$\Rightarrow z_W \cdot (b + x_{B_L} - x_{B_R}) = b \cdot (z_W + f) \tag{2.36}$$

$$\Rightarrow z_W \cdot (x_{B_L} - x_{B_R}) = b \cdot f \tag{2.37}$$

$$\Rightarrow z_W = \frac{b \cdot f}{x_{B_L} - x_{B_R}} = \frac{b \cdot f}{d} \tag{2.38}$$

z_W - z-Koordinate des Weltpunktes P_W (Abstand des betrachten Objektes im Raum zum Stereokamerasystem)

f - Brennweite

C_L, C_R - Optische Zentren der Kameras

b - Basislinie (Abstand der beiden Kameras voneinander)

π_L, π_R - Bildebenen der beiden Kameras

x_L, x_R - x-Bildkoordinaten der beiden Kameras

d - Disparität (Unterschied zwischen den x-Bildkoordinaten der beiden Kameras)

2.4.5. Tiefenkarte

Die Disparität ist der Unterschied zwischen den x-Koordinaten zweier korrespondierender Bildpunkte. Je näher sich dabei ein betrachteter Punkt im Raum an der Kamera befindet, desto größer ist die Disparität.

Algorithmen die Stereobilder verarbeiten, speichern ihre Ergebnisse in sogenannten *Tiefenkarten*. Die Disparität eines korrespondierenden Punktepaares wird auf eine Skala mit 255 Werten normiert. Die Ergebnisse der Normierung aller Punktkorrespondenzen werden anschließend als ein Graustufenbild gespeichert. Eine Tiefenkarte ist also ein Graustufenbild das die Tiefe in Helligkeitsstufen kodiert. Dabei werden näher liegende Objekte heller dargestellt (siehe Abbildung 3.26).

2.5. Korrespondenzsuche in Stereobildern

Zur Lösung des Korrespondenzproblems existieren mittlerweile viele unterschiedliche Verfahren. In [Scharstein und Szeliski (2002)] wird die Vorgehensweise der üblichen Verfahren genau untersucht. Demnach können die meisten Stereoalgorithmen in folgende vier Schritte unterteilt werden:

1. Kostenberechnung
2. Kostenaggregation
3. Berechnung der Disparitäten
4. Verfeinerung der Disparitäten

2.5.1. Kostenberechnung

Um korrespondierende Bildpunkte in Stereoaufnahmen zu finden, werden bestimmte Bildausschnitte/Regionen der einzelnen Bilder nach ähnlichen Bildpunkten durchsucht. Wie weiter oben bereits beschrieben, liegen korrespondierende Bildpunkte einer rektifizierten Stereoaufnahme in der gleichen Zeile y. Im Fall der dynamischen Programmierung (wird im Abschnitt 3.5 vorgestellt) sind korrespondierende Bildzeilen die Regionen, die nach Korrespondenzen durchsucht werden. Solche Regionen werden auch Fenster genannt.

Um zu bestimmen, ob zwei Punkte sich ähnlich sind, muss ein Ähnlichkeitsmaß als Vergleichskriterium definiert werden. Ein Ähnlichkeitsmaß ist eine Funktion, die zwei Punkte miteinander vergleicht. Das Ergebnis dieser Funktion sind die Kosten die den Unterschied

zwischen den beiden Punkten beschreiben. Je kleiner dabei die Kosten, das heißt je kleiner der Unterschied, desto ähnlicher sind sich die Punkte. Ein Ähnlichkeitsmaß wird auch als *Kostenfunktion* oder *Energiefunktion* bezeichnet.

In der Praxis werden, wegen ihrer einfachen Berechnung, oft die beiden Kostenfunktionen *Summe absoluter Differenzen* (sum of absolute differences) (SAD) und *Summe der quadrierten Differenzen* (sum of squared differences) (SSD) verwendet:

$$SAD(y,d) = \Sigma_j \Sigma_i |I_R(y,i) - I_L(y,j+d)| \tag{2.39}$$

$$SSD(y,d) = \Sigma_j \Sigma_i (I_R(y,i) - I_L(y,j+d))^2 \tag{2.40}$$

Die Gleichungen 2.39 und 2.40 beschreiben die Berechnungsvorschriften für die Zeile y eines Stereobildpaares. I_L und I_R entsprechen dabei dem linken und dem rechten Bild, i und j geben den Spaltenindex an und der Parameter d steht für die Disparität.

Das Thema „Kostenfunktionen“ wird sehr ausführlich in [Hermann und Klette] und in [Hirschmüller und Scharstein (2007)] behandelt.

2.5.2. Kostenaggregation

Da benachbarte Bildpunkte (Pixel) sich bezüglich ihrer Farbwerte sehr ähnlich sein können, gelingt es allein mit den Kosten einzelner Pixel nicht, die richtigen Korrespondenzen zu finden. Deswegen werden die Kosten aller Pixel im Suchfenster aufsummiert. Anschließend wird eine Lösung gesucht, bei der die globalen Kosten minimal sind. Die Lösung mit den geringsten Kosten liefert die bestmögliche Zuordnung korrespondierender Bildpunkte.

2.5.3. Berechnung der Disparitäten

In diesem Schritt wird mit einem geeigneten Verfahren versucht, die Kostenfunktion zu minimieren. Die Minimierung der Kostenfunktion, also die eigentliche Korrespondenzsuche, ist in der Regel nicht trivial. Aus diesem Grund müssen für die jeweiligen Verfahren bestimmte Bedingungen erfüllt sein.

Neben der weiter oben beschriebenen Epipolarbedingung, ist die bedeutendste Bedingung für das in dieser Arbeit verwendete Verfahren (dynamische Programmierung), die sogenannte *Reihenfolgebedingung* (ordering constraint).

Die Reihenfolgebedingung besagt, dass korrespondierende Punkte in den Bildern einer Stereoaufnahme, in gleicher Reihenfolge auftreten müssen. Abbildung 2.20 zeigt jedoch, dass

diese Bedingung nicht immer erfüllt werden kann. In Abbildung 2.20 führt das kleinere Objekt im Vordergrund zu Fehlkorrespondenzen, denn die Reihenfolgebedingung ist dort nicht erfüllt.

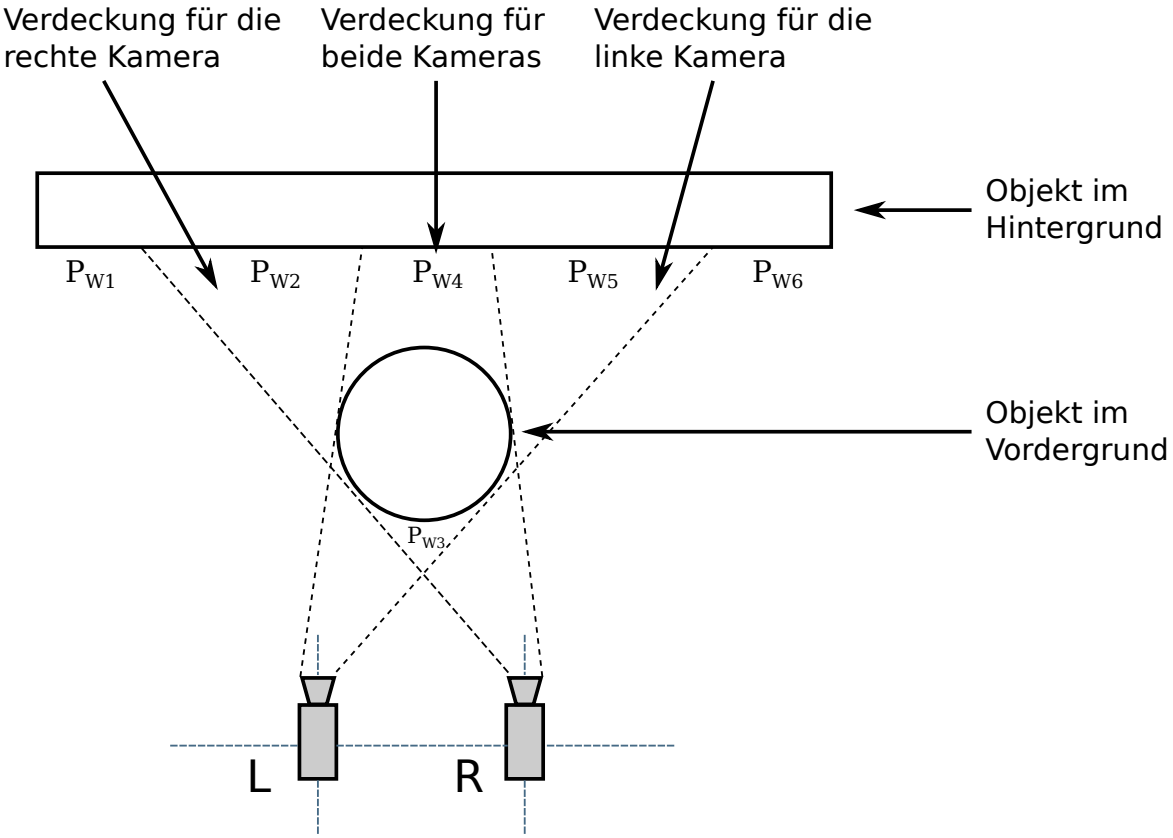


Abbildung 2.20.: Verletzung der Reihenfolgebedingung: Reihenfolge der Bildpunkte der linken Kamera: $P_{L_{B1}}$, $P_{L_{B2}}$, $P_{L_{B3}}$, $P_{L_{B6}}$ Reihenfolge der Bildpunkte der rechten Kamera: $P_{R_{B1}}$, $P_{R_{B3}}$, $P_{R_{B5}}$, $P_{R_{B6}}$

Grundsätzlich können die Algorithmen/Verfahren für die Korrespondenzsuche in zwei Kategorien unterteilt werden. Es gibt lokale und globale Ansätze.

Lokale Ansätze

Lokale Ansätze führen die Kostenaggregation und die Korrespondenzsuche lokal in einem sogenannten *Korrelationsfenster* durch. Ein Korrelationsfenster ist ein kleiner Ausschnitt eines Bildes (zum Beispiel $8x8$ Pixel). Um lokale Mehrdeutigkeiten zu reduzieren, müssen bei diesen Ansätzen die Fenstergröße und die Kostenfunktion gut gewählt werden, denn als Ergebnis wird einfach das Minimum der lokalen Kosten angenommen.

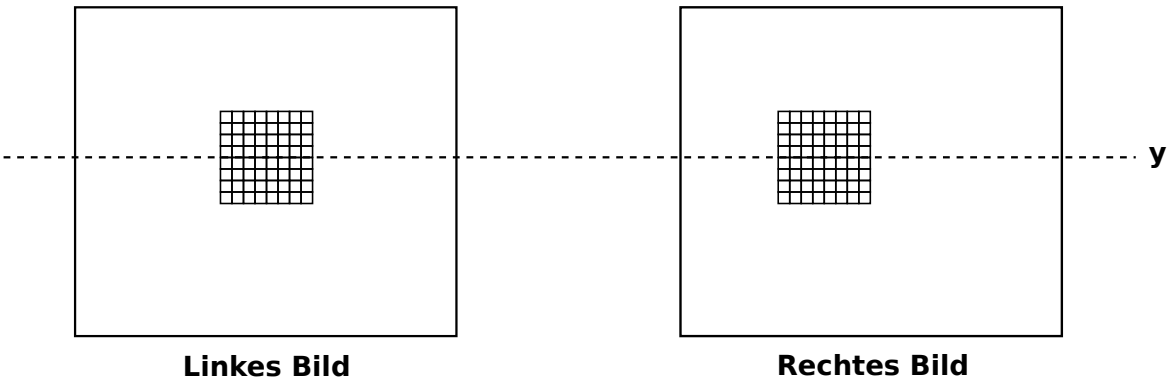


Abbildung 2.21.: Korrespondenzsuche mit Hilfe eines Korrelationsfensters

Globale Ansätze

Globale Ansätze durchsuchen das gesamte Bild pixelweise. Es wird versucht die Kostenfunktion über alle Pixel der Bilder einer Stereoaufnahme zu minimieren. Globale Ansätze weisen eine hohe Komplexität auf. Durch die Berücksichtigung der Gesamtkosten, können aber lokale Mehrdeutigkeiten gut aufgelöst werden.

Auch die in dieser Arbeit verwendete dynamische Programmierung gehört zu den globalen Ansätzen, obwohl sie nur mit einem Teil der Bilder (zeilenweise) arbeitet. Solche Ansätze werden auch *semi-globale* Verfahren genannt.

2.5.4. Verfeinerung der Disparitäten

Einige Verfahren versuchen vor oder nach der Korrespondenzsuche die Korrespondenzen nicht nur pixel- sondern subpixelgenau zu bestimmen. Eine genauere Bestimmung der Positionen korrespondierender Punkte resultiert in einer feiner aufgelösten Disparitätskarte. Für Disparitätsverfeinerungen existieren verschiedene Ansätze, als Beispiel siehe Abschnitt 3.6.6.

3. Beschreibung des Verfahrens

Das in dieser Arbeit vorgestellte Verfahren zur Tiefenbildgewinnung aus Stereobildern basiert auf dem Verfahren der *dynamischen Programmierung*. Wie bereits im Anhang A erläutert, wird die dynamische Programmierung in verschiedenen Bereichen zum Lösen von Optimierungsproblemen eingesetzt. Das Verfahren hat sich auch in der Stereoskopie bewährt und wird dort aufgrund seiner Eigenschaften oft verwendet.

Für die Verarbeitung von Bilddaten benötigt die dynamische Programmierung einen Ähnlichkeitsmaß, um zwei Bildpunkte (Pixel) miteinander vergleichen zu können. Es existieren mehrere Standardfunktionen (siehe Abschnitt 2.5.1) die als Ähnlichkeitsmaß fungieren und als Funktionsparameter den Disparitätswert zweier korrespondierender Punkte erwarten. Für jedes Punktepaar das miteinander verglichen werden soll, muss also die Disparität zwischen diesen beiden Punkten bekannt sein. Da die Höhe der Disparität abhängig ist vom Abstand der abgebildeten Objekte zur Kamera, kann die Disparität auch innerhalb eines Bildpaares stark variieren. Aus diesem Grund wird ein Verfahren benötigt, welches eine automatische Ermittlung der Disparität ermöglicht.

Das vorgestellte Verfahren gliedert sich grob in 5 Schritte.

1. Im ersten Schritt werden, mit einer geeigneten Methode, einige robuste Punktkorrespondenzen zwischen beiden Bildern ermittelt. Aus den Koordinaten der ermittelten Punktepaare können dann die zugehörigen Disparitäten errechnet werden. Mit Hilfe dieser Disparitäten erhält man eine automatische Bestimmung der durchschnittlichen Disparität für beliebige Aufnahmen. Als Voraussetzung muss das Bildmaterial genügend Merkmale enthalten, so dass es einem gelingt, einige Punktkorrespondenzen zu ermittelt.

2. Die im ersten Schritt gefundenen Punktkorrespondenzen können weiter ausgenutzt werden, indem aus der Punktmenge des linken Bildes, mit Hilfe der *Delaunay Triangulation*, ein Dreiecksnetz erstellt wird. Da das Netz auf Basis korrespondierender Punkte erzeugt wird, ist sichergestellt, dass jedes Teildreieck des Netzes auch im rechten Bild, aufgrund der anderen Perspektive verzerrt, enthalten ist.

3. Im nächsten schritt wird für jedes Dreieck im linken Bild geprüft, ob alle von diesem Dreieck eingeschlossenen Pixel, lokal auf derselben Objektebene liegen und ob das korrespondierende Dreieck im rechten Bild dieselbe Objektebene beschreibt.

4. Nach bestandener Prüfung, wird für jedes Pixel eines Dreiecks, unter Verwendung der *projektiven Transformation*, das zugehörige Pixel im rechten Bild errechnet. In diesem Schritt werden auf eine „einfache“ Art und Weise für ganze Pixelbereiche, die sich über mehrere Zeilen erstrecken, zuverlässige Korrespondenzen ermittelt.

5. Durch diesen Ansatz können bereits in der Vorverarbeitung viele robuste Korrespondenzen gefunden werden. Anschließend werden die übriggebliebenen Lücken mit Punkten, die nicht im Laufe der Vorverarbeitung berechnet werden konnten, mit Hilfe der dynamischen Programmierung geschlossen. Die vielen bereits berechneten Korrespondenzen dienen der dynamischen Programmierung als Anhaltspunkte[1], an denen sich die Berechnung orientieren kann. Das führt zu einem besseren Ergebnis und, da nicht mehr alle Punkte berechnet werden müssen, zur Beschleunigung des gesamten Verfahrens.

Zur Übersicht skizziert Abbildung 3.1 noch einmal die einzelnen Schritte des Verfahrens. Im Folgenden wird jeder Schritt im Detail beschrieben.

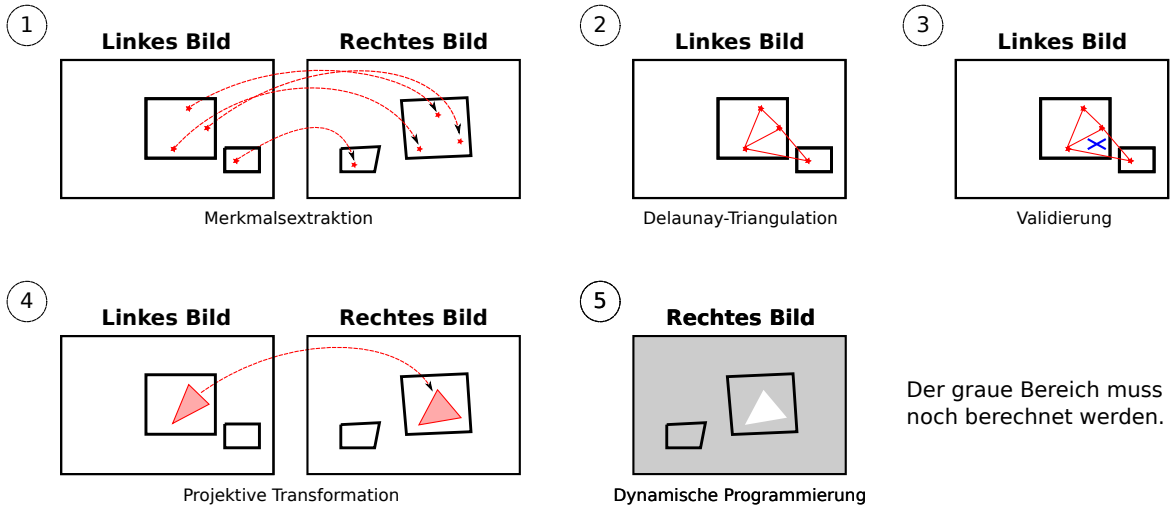


Abbildung 3.1.: Die einzelnen Schritte des Verfahrens im Überblick

3.1. Merkmalsextraktion

In der Bildverarbeitung sind eine Reihe von Verfahren zur Extraktion markanter Merkmale bekannt. Markante Merkmale können etwa Ecken, Kanten, Muster oder Konturen sein. Anhand dieser Merkmale ist es beispielsweise möglich, bestimmte Objekte in verschiedenen Bildern zu identifizieren.

[1] *Ground Control Points* (GCP) (3.5.5)

Es existiert keine allgemeingültige Methode. Für jede Aufgabe müssen passende Merkmale ausgewählt werden. Für die gegebene Aufgabenstellung wird ein Extraktionsverfahren benötigt, welches die gefundenen Merkmale pixelgenau und sehr zuverlässig Objekten zuordnen kann. Zwei bekannte Verfahren die diese Voraussetzungen erfüllen und oft zum Einsatz kommen sind *Scale Invariant Feature Transform* (SIFT) [Lowe (2004)] und *Speeded Up Robust Features* (SURF) [Bay u. a. (2006)]. Die durch diese Verfahren extrahierten Merkmale, verfügen über eine große Invarianz gegenüber Rauschen, Beleuchtung, Skalierung, Translation und Rotation.

Beide Verfahren zeigen eine gute Ausführungsgeschwindigkeit. Eigene (Tabelle 3.1) und andere Untersuchungen (siehe zum Beispiel [Juan und Gwon (2009)]) haben gezeigt, dass SURF deutlich schneller als SIFT arbeitet, SIFT aber im Gegensatz zu SURF, in den meisten Fällen, wesentlich mehr Merkmale findet. Die modulare Implementierung des in dieser Arbeit vorgestellten Verfahrens, ermöglicht den Einsatz eines beliebigen Verfahrens zum Auffinden von robusten Merkmalen. Dem Programm wird über Parameter mitgeteilt, welches Verfahren zum Einsatz kommen soll. Die beiden im Folgenden beschriebenen Verfahren sind bereits implementiert.

3.1.1. SIFT - Scale Invariant Feature Transform

SIFT kann grob in vier Schritte unterteilt werden: Suchen nach potentiellen Merkmalskandidaten, Filterung und Lokalisierung von Merkmalen, Bestimmung der Orientierung gefundener Merkmale und Beschreibung der Merkmale durch Deskriptoren.

Suche nach potentiellen Merkmalskandidaten

Im ersten Schritt wird nach Extremstellen im sogenannten *Skalenraum* (scale-space) $L(x, y, \sigma)$ gesucht. Der Skalenraum wird folgendermaßen erstellt:

$$L(x, y, \sigma) = G(x, y, \sigma) \cdot I(x, y) \tag{3.1}$$

Zuerst wird die Auflösung des zu analysierenden Bildes $I(x, y)$ n-mal, jeweils um den Faktor 2, verkleinert. Die Zwischenergebnisse der Skalierung werden übereinandergelegt. Jede der n verschiedenen Auflösungen des Bildes wird dann k-mal mit einem Gaußfilter $G(x, y, \sigma)$ geglättet. x und y sind die Koordinaten eines Pixels im Bild und der Parameter σ gibt die Varianz, also die beim Glätten entstehende Streuung bezüglich x und y an. So entsteht eine n-schichtige Pyramide mit k Glättungsstufen in jeder Schicht. Eine Schicht der Pyramide wird *Oktave* genannt.

Anschließend werden in jeder Oktave die jeweils benachbarten Glättungsstufen voneinander abgezogen. Da bei diesem Vorgang der Unterschied zwischen zwei aufeinander folgenden Gaußfilterungen berechnet wird, bezeichnet man diese Subtraktion als *Difference of Gaussian* (DOG) $D(x, y, \sigma)$. In Abbildung 3.2 ist der beschriebene Vorgang dargestellt.

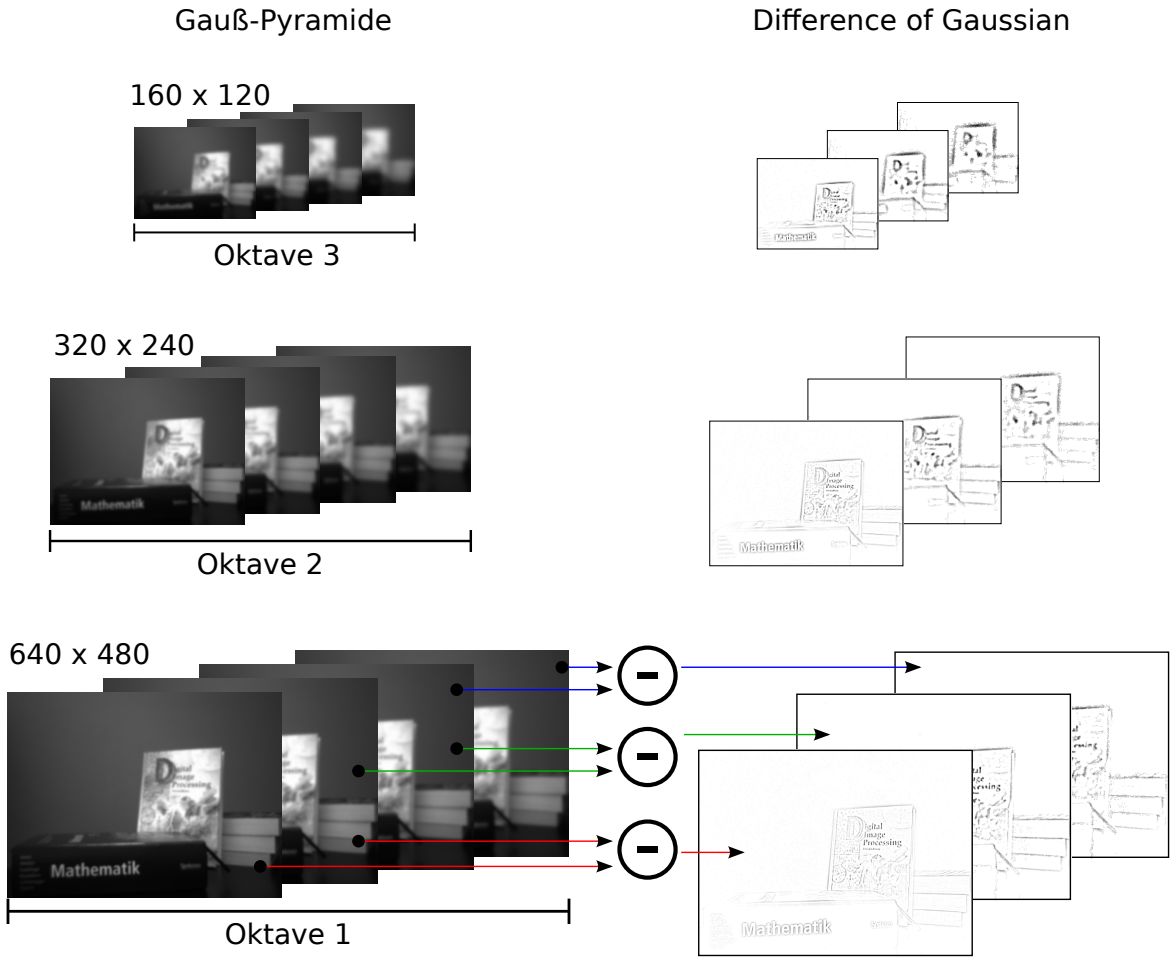


Abbildung 3.2.: Skalenraum (links) und der dazugehörige differentielle Skalenraum (rechts)

DOG ist eine Annäherung an den *Laplacian of Gaussian Operator* (LOG). SIFT benutzt DOG aufgrund der höheren Ausführungsgeschwindigkeit gegenüber LOG.

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \cdot I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma) \quad (3.2)$$

Nachdem die Gauß-Pyramide und die zugehörige DOG-Pyramide (in Abbildung 3.2 rechts dargestellt) erstellt worden sind, wird in der DOG-Pyramide, also im differentiellen Skalenraum, nach Extremstellen gesucht. Extremstellen werden als potentielle Merkmalskandidaten betrachtet und sind die Stellen, die ein Minimum oder ein Maximum bezüglich benach-

barter Pixel aufweisen. Zu den benachbarten Pixeln gehören die 8 umliegenden Pixel auf der aktuellen Ebene inerhalb einer Oktave und jeweils 9 Pixel der beiden angrenzenden Ebenen. Insgesamt ergeben sich 26 Nachbarpixel bei zwei angrenzenden Ebenen, siehe Abbildung 3.3. Ein Minimum wird gefunden, falls die Werte aller Nachbarpixel größer als die des aktuelle Pixels sind. Ein Maximum ergibt sich, wenn alle benachbarten Pixel einen kleineren Wert aufweisen.

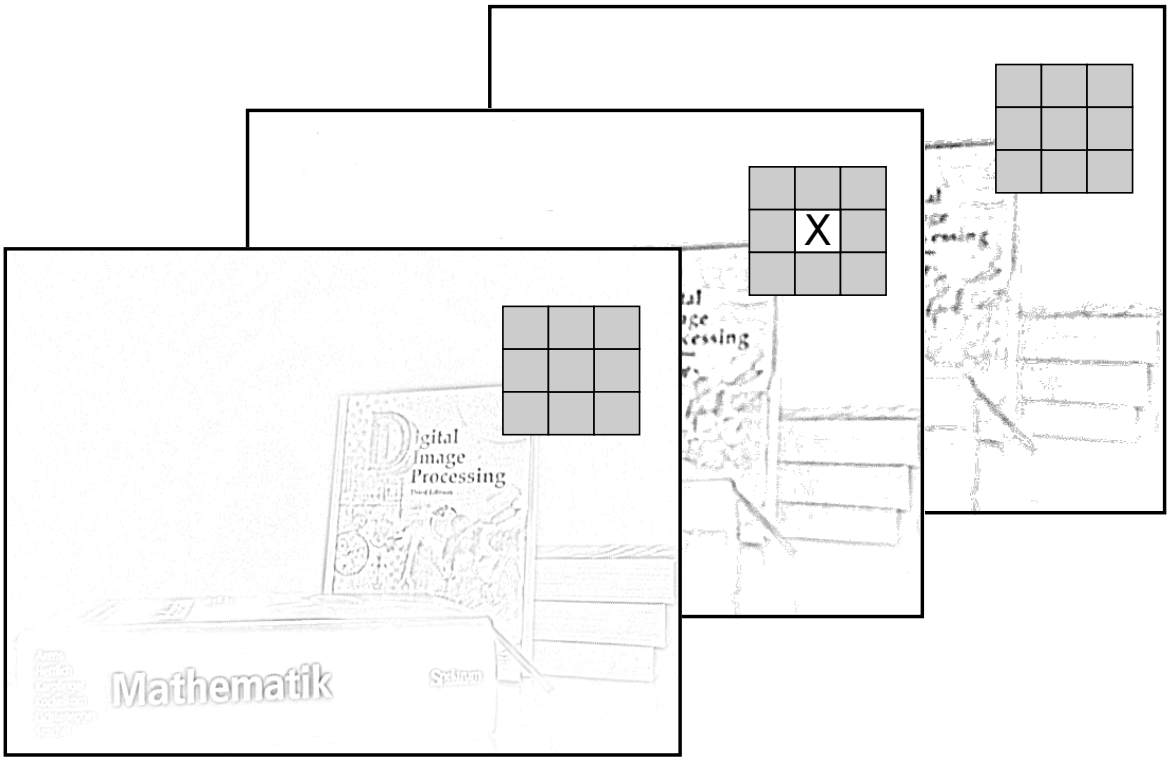


Abbildung 3.3.: 26 Nachbarn eines Pixels bei der Suche nach potentiellen Merkmalskandidaten im differentiellen Skalenraum. Das aktuelle Pixel ist mit X gekennzeichnet, die Nachbarpixel sind grau dargestellt.

Filterung und Lokalisierung von Merkmalen

Alle im ersten Schritt gefundenen Extremstellen werden einer Stabilitätsprüfung unterzogen. Besteht ein Kandidat die Prüfung, wird angenommen, dass es sich um ein robustes Merkmal handelt und es wird seine Subpixelposition und Skalierung bestimmt. Kandidaten die beim Stabilitätstest durchfallen, werden verworfen.

Bei dem Stabilitätstest wird geprüft, ob der Kontrast des Kandidaten einen bestimmten Schwellwert nicht überschreitet und ob das Bildrauschen eine eindeutige Lokalisierung des

Kandidaten zulässt. Des Weiteren wird bestimmt, ob die Extremstelle zu einer Ecke, einer Kante oder zu einem homogenen Bereich gehört.

Bestimmung der Orientierung

In diesem Schritt wird jedem Merkmal seine Orientierung zugeordnet. Durch diese Information zusammen mit dem Merkmalsvektor, wird Invarianz gegenüber Bilddrehungen erreicht.

Die Orientierung des Merkmals kann anhand seiner umliegenden Gradienten ermittelt werden. Ein Gradient ist ein Vektor, mit der Länge $m(x, y)$ und der Richtung $\theta(x, y)$, der den steilsten Anstieg der Helligkeit eines Bildpunktes bestimmt.

Um die Berechnung der Gradienten skalierungsinvariant durchzuführen, wird aus dem Skalenraum $L(x, y, \sigma)$ ein Bild L ausgewählt, welches der Skalierung des Merkmalspunktes am nächsten kommt. Die Berechnung der Länge und der Orientierung des Gradienten bezüglich eines Bildpunktes $L(x, y)$ in der entsprechenden Skalierung, wird wie folgt durchgeführt:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{3.3}$$

$$\theta(x, y) = tan^{-1}((L(x, y+1) - L(x, y-1))/(L(x+1, y) - L(x-1, y))) \tag{3.4}$$

Aus den Orientierungen der Gradienten der Punkte, die ein Merkmalspunkt umgeben, wird ein Histogramm aufgestellt. Das Orientierungshistogramm beinhaltet 36 Werte, die den gesamten Winkelbereich von $360°$ abdecken. Jeder Histogrammwert hat somit eine Abdeckung (Breite) von $\frac{360°}{36} = 10°$ und wird mit seiner Gradientenlänge $m(x, y)$ und mit einer Gaußfunktion gewichtet. Der Parameter σ der Gaußfunktion ist dabei $1,5$ mal größer als die Skalierung des Merkmalspunktes. Der höchste Histogrammwert bestimmt schließlich die Orientierung des Merkmalspunktes.

Beschreibung der Merkmale durch Deskriptoren

Als Deskriptor wird ein Merkmalsvektor bezeichnet, der einen Merkmalspunkt eindeutig beschreibt.

Bei SIFT wird der Merkmalsvektor mit Hilfe der umliegenden Pixel eines Merkmalspunktes erstellt. Dazu wird die Nachbarschaft des Merkmalspunktes aus $4 \cdot 4 = 16$ Teilbereichen betrachtet. Jeder Teilbereich deckt jeweils $4 \cdot 4 = 16$ Pixel ab. Zunächst wird für jedes der 16 Pixel der dazugehörige Gradient berechnet. Um den Einfluss vom Merkmalspunkt weiter entfernter Gradienten zu verringern, werden die Gradienten mit einer Gaußfunktion gewichtet, deren σ der Hälfte des Deskriptorfensters entspricht. Dann wird aus den Gradienten eines

Teilbereichs, ähnlich wie im Schritt 3 (3.1.1) beschrieben, ein Orientierungshistogramm erstellt. Die Gradienten werden im Histogramm zu 8 möglichen Richtungen zusammengefasst. Die 8 Richtungen werden auf $360°$ abgebildet, jede Richtung deckt somit einen Winkelbereich von $\frac{360°}{8} = 45°$ ab. Der SIFT-Deskriptor ist also ein 128-dimensionaler Vektor, der sich aus 16 Teilbereichen mit jeweils 8 Winkelbereichen zusammengesetzt $16 \cdot 8 = 128$.

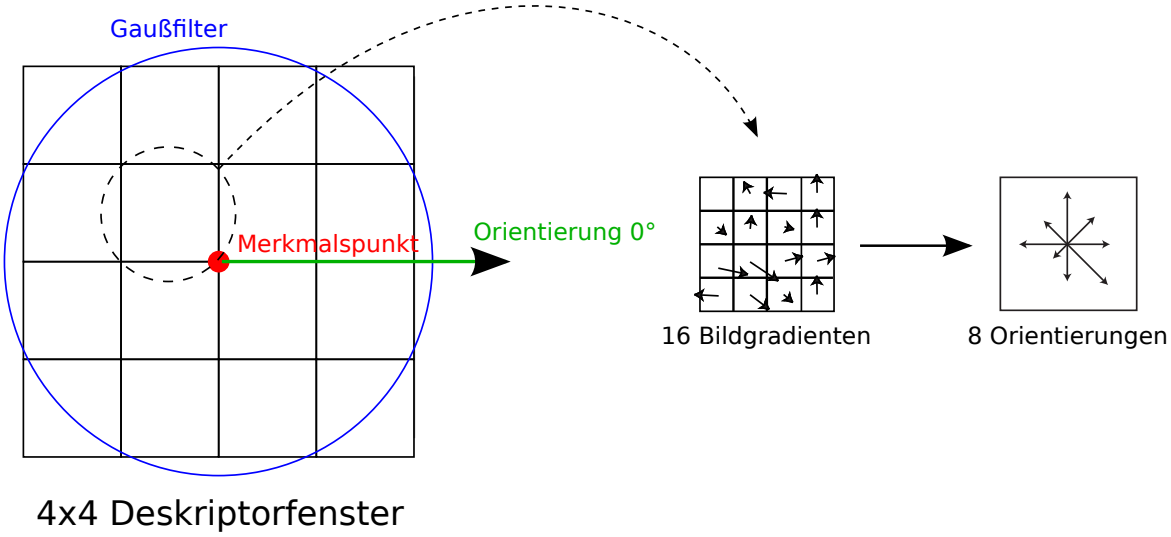


Abbildung 3.4.: Erstellen eines SIFT-Deskriptors für einen Merkmalspunkt

Um eine Invarianz gegenüber Bilddrehungen zu erreichen, werden die Koordinaten des Deskriptors und die Orientierungen der Gradienten, relativ zu der, im Schritt 3 (3.1.1) berechneten, Orientierung des Merkmalspunktes gedreht.

3.1.2. SURF - Speeded Up Robust Features

SURF wurde (2006) 7 Jahre nach SIFT (1999) veröffentlicht. Das Verfahren orientiert sich an SIFT, versucht aber durch einen neuen Ansatz für die Berechnung von Merkmalspunkten und Merkmalsvektoren/Deskriptoren, schneller und robuster als sein Vorgänger zu sein.

Suche nach Merkmalspunkten

Anstelle einer Gaußfilterung benutzt SURF für die Lokalisierung und Skalierung von Merkmalspunkten die Determinante der *Hesse-Matrix*.

Die Hesse-Matrix ist eine quadratische Matrix (Zeilenanzahl entspricht der Spaltenanzahl), die die zweiten partiellen Ableitungen einer reellen Funktion beinhaltet. Es werden also die

lokalen Krümmungen einer Funktion mit mehreren Variablen beschrieben. Die Hesse-Matrix ist für einen Punkt $X = (x, y)$ im Bild I in der Skalierung σ wie folgt definiert:

$$H(X, \sigma) = \begin{pmatrix} L_{xx}(X, \sigma) & L_{xy}(X, \sigma) \\ L_{xy}(X, \sigma) & L_{yy}(X, \sigma) \end{pmatrix} \tag{3.5}$$

Die Einträge in H entsprechen der Faltung der zweiten Ableitung der Gaußfunktion $g(\sigma)$ im Punkt X des Bildes I. So steht zum Beispiel $L_{xx}(X, \sigma)$ für die 2. Ableitung $\frac{\delta^2}{\delta x^2} g(\sigma)$ im Punkt X.

Die Determinante wird berechnet durch:

$$det(H(X, \sigma)) = L_{xx}(X, \sigma) L_{yy}(X, \sigma) - (L_{xy}(X, \sigma))^2 \tag{3.6}$$

Eine Approximation der Determinante kann als einfacher Mittelwertfilter benutzt werden.

Ähnlich wie Lowe[2] mit seinem DOG-Operator den *Laplacian of Gaussian Operator* (LOG) approximiert, approximiert Bay[3] in SURF den *Hessian Laplace Operator* mit Hilfe von Integralbildern. Bay bezeichnet seine Approximation als *Fast Hessian Detector*.

Mit einem Integralbild können schnell Pixelsummen berechnet werden, die sich innerhalb eines rechteckigen Bildausschnitts befinden. Ein Integralbild I ist eine Tabelle mit aufsummierten Pixelwerten eines Bildes. Ein Punkt (x, y) in I entspricht der Summe aller Pixel, die sich innerhalb des Rechtecks befinden, der von den beiden Punkten (x, y) und $(0, 0)$ im Ursprungsbild aufgespannt wird.

$$I(x, y) = \sum_{i=0}^{i<x} \sum_{j=0}^{j<y} I(i, j) \tag{3.7}$$

Ein Integralbild eines Bildes kann in einem Durchlauf über das Bild erzeugt werden. Ein Punkt (x, y) in I wird folgendermaßen berechnet:

$$I(x, y) = i(x, y) + I(x - 1, y) + I(x, y - 1) - I(x - 1, y - 1) \tag{3.8}$$

Gleichung 3.8 wird in Abbildung 3.5 verdeutlicht.

Nachdem das Integralbild einmal berechnet ist, kann die Summe von Intensitätswerten eines beliebigen großen Bildausschnitts in konstanter Zeit berechnet werden. Für die Berechnung

[2] Lowe (2004)
[3] Bay u. a. (2006)

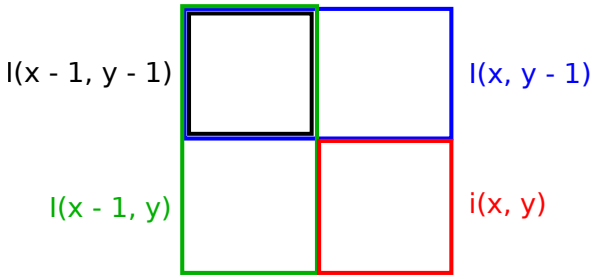


Abbildung 3.5.: Berechnung des Punktes (x, y) eines Integralbildes I. I besteht hier aus $2x2$ Pixel. $I(x-1, y-1)$ muss abgezogen werden, da es durch $I(x-1, y) + I(x, y-1)$ doppelt addiert wird.

werden jeweils lediglich die 4 Punkte benötigt, die den gewünschten Bildausschnitt aufspannen.

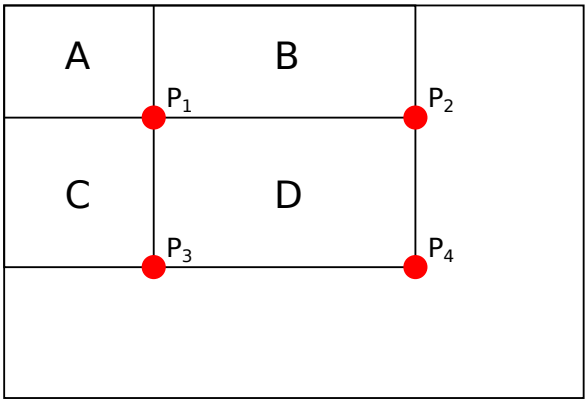


Abbildung 3.6.: Berechnung der Summe von Intensitätswerten eines Bildausschnitts mit Hilfe eines Integralbildes

In Abbildung 3.6 entspricht der Wert des Integralbildes im Punkt P_1 der Summe der Pixel im Rechteck A. Der Wert im Punkt P_2 entspricht $A+B$, $P_3 = A+C$ und $P_4 = A+B+C+D$. Die Summe der Pixel in D kann somit berechnet werden durch:

$$P_4 + P_1 - P_2 - P_3 = A + B + C + D + A - A - B - A - C = D \tag{3.9}$$

Bei SIFT wird der Skalenraum untersucht, indem der gleiche Filter auf unterschiedliche Skalierungen des Originalbildes angewendet wird. Da mit einem Integralbild beliebig große Ausschnitte mit gleichem Aufwand berechnet werden können, wird bei SURF stattdessen die Größe des Filters variiert, der direkt auf das Originalbild angewendet wird.

Berechnung des Deskriptors

Im Vergleich zu anderen Deskriptoren [Mikolajczyk und Schmid (2005)] hat sich der SIFT-Deskriptor als sehr performant erwiesen. Deshalb basiert der SURF-Deskriptor auf ähnlichen Merkmalen und verringert dabei den Berechnungsaufwand. Die Erzeugung des Deskriptors geschieht in zwei Schritten. Im ersten Schritt wird auf Basis eines kreisförmigen Bereichs um den Merkmalspunkt, dessen Orientierung festgestellt. Im zweiten Schritt wird aus einer, in die Orientierungsrichtung ausgerichteten, rechteckigen Region, der Deskriptor gebildet.

Um Invarianz bezüglich Bilddrehungen zu erreichen, wird im Radius von $6s$ um den Merkmalspunkt, in x und y Richtung, die *Haar-Wavelet*[4]-Antwort berechnet. s steht für die Skalierung des Merkmalspunktes. Die Wavelet-Antworten werden auch in der aktuellen Skalierung s berechnet. Aus Geschwindigkeitsgründen werden für die Berechnung der Wavelets wieder Integralbilder benutzt. In einer beliebigen Skalierung werden für die Berechnung der Antwort in x oder y Richtung nur sechs Rechenoperationen benötigt. Die Seitenlänge des Wavelets beträgt $4s$. Die berechneten Wavelet-Antworten werden, ausgehend vom Merkmalspunkt, mit einer Gaußfunktion, die eine Streuung $\sigma = 2,5s$ aufweist, gewichtet. Die gewichteten Antworten werden dann als horizontale und vertikale Vektoren dargestellt. Alle Vektoren innerhalb eines verschiebbaren Orientierungsfensters, das einen Winkel von $\frac{\pi}{3}$ abdeckt, werden aufsummiert. Die aufsummierten Vektoren ergeben jeweils einen neuen Vektor. Der längste dieser neuen Vektoren bestimmt die Orientierung des Merkmalspunktes.

Der Deskriptor wird konstruiert, indem um den Merkmalspunkt, ausgerichtet in die zuvor berechnete Orientierung, ein Quadrat definiert wird. Das Quadrat wird in $4 \cdot 4 = 16$ gleichgroße Bereiche aufgeteilt. Dann wird jeder Bereich in $5 \cdot 5 = 25$ weitere Unterbereiche unterteilt. Für alle 25 Unterbereiche wird die Haar-Wavelet-Antwort in die horizontale d_x und in die vertikale d_y Richtung berechnet. Um robuster gegen geometrische Transformationen zu sein und um kleine lokale Fehler auszugleichen, werden die Antworten d_x und d_y mit einer Gaußfunktion ($\sigma = 3,3s$) gewichtet. d_x und d_y werden anschließend für alle 16 Bereiche aufsummiert. Um die Polarität der Intensitätsänderungen zu berücksichtigen, werden auch die Absolutwerte $|d_x|$ und $|d_y|$ der Antworten aufsummiert. Für die 16 Teilbereiche des Quadrats ergibt sich jeweils ein 4-dimensionaler Vektor v:

$$v = (\sum d_x, \sum d_y, \sum |d_x|, \sum |d_y|) \tag{3.10}$$

[4] Als Wavelet bezeichnet man eine kontinuierliche oder diskrete Wavelet-Transformation. Wavelet-Transformationen sind lineare Zeit-Frequenz-Transformationen. Sie werden zum Beispiel bei der Bild-/Videokompression, zum Lösen von Differentialgleichungen und in der Signalverarbeitung verwendet. Das Haar-Wavelet ist das einfachste bekannte Wavelet. Es wird aus zwei Rechteckfunktionen gebildet und ist dadurch sehr einfach zu implementieren.

Der gesamte Deskriptor setzt sich also aus 16 Teilbereichen mit jeweils 4 Dimensionen zusammen und hat damit eine Länge von $16 \cdot 4 = 64$.

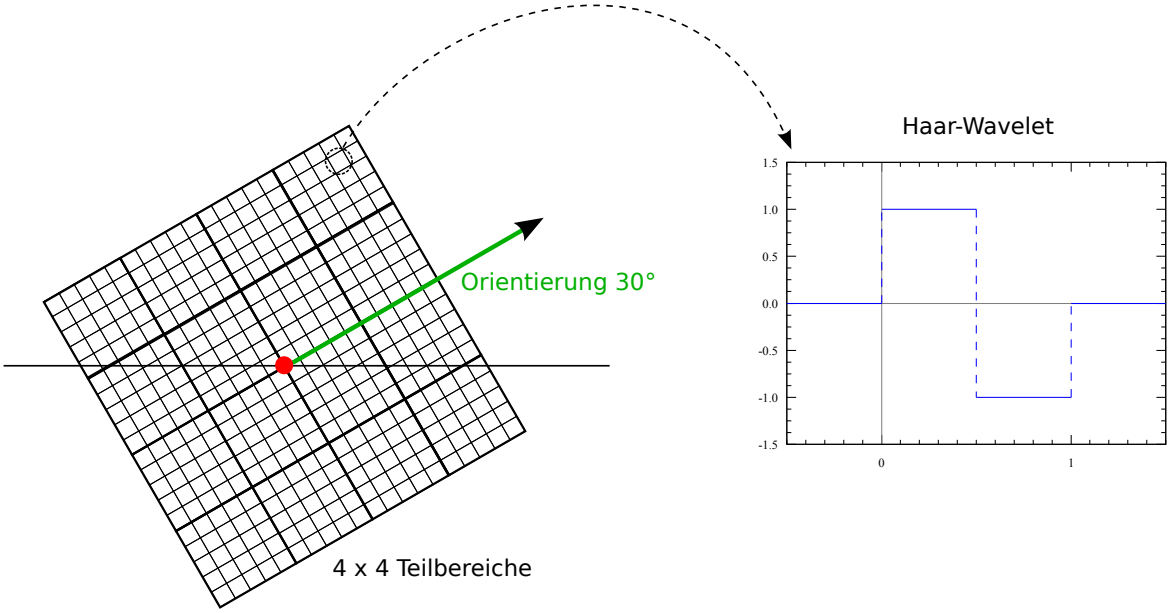


Abbildung 3.7.: Berechnung des SURF-Deskriptors

3.1.3. Korrespondenzsuche (matching)

Robuste Bildmerkmale werden in dieser Arbeit eingesetzt, um Pixelkorrespondenzen zwischen zwei unterschiedlichen Aufnahmen einer Szene zu finden. Ein robustes Merkmal sagt aus, dass dieses Merkmal mit großer Wahrscheinlichkeit in verschiedenen Aufnahmen wiedergefunden werden kann. Um nun korrespondierende Merkmale in verschiedenen Aufnahmen einander zuordnen zu können, wird ein geeignetes Verfahren benötigt.

Suche nach dem nächsten Nachbarn

Das Korrespondenzproblem kann mit Hilfe des *euklidischen Abstandes* gelöst werden, der zwischen einem Merkmalsvektor in einem Bild und einem Merkmalsvektor in einem anderen Bild berechnet wird.

Der euklidische Abstand zwischen zwei Punkten, ist die Länge der Strecke, die diese beiden Punkte verbindet. In der Ebene kann der euklidische Abstand mit Hilfe des Satzes von

Pythagoras berechnet werden. Allgemein wird der euklidische Abstand d zwischen den n-dimensionalen Vektoren x und y definiert durch:

$$d(x, y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \tag{3.11}$$

Ein korrespondierendes Vektorpaar ist gefunden, falls der Abstand zwischen diesen Merkmalsvektoren kleiner als das $0,7$-fache des Abstandes zu dem nächstliegenden Nachbarvektor ist.

Dieses Verfahren wird von SURF bevorzugt. Es wird *Suche nach dem nächsten Nachbarn* (Nearest-Neighbor) oder auch *Brute Force* genannt.

Suche im k-d-Baum

Eine effizientere Lösung ist die Suche nach Korrespondenzen mit Hilfe von sogenannten *k-d-Bäumen* [Bentley (1975)].

Ein k-d-Baum (siehe Abbildung 3.8) ist ein k-dimensionaler binärer Baum. Die Merkmalsvektoren werden von den Blättern des Baumes repräsentiert. Die Brute-Force-Suche führt bei vielen Merkmalspunkten und einer hohen Dimension der Merkmalsvektoren zu einer zeitintensiven Berechnung. Wohingegen ein k-d-Baum bei n Merkmalsvektoren eine maximale Dimension von $log_n 2$ aufweist.

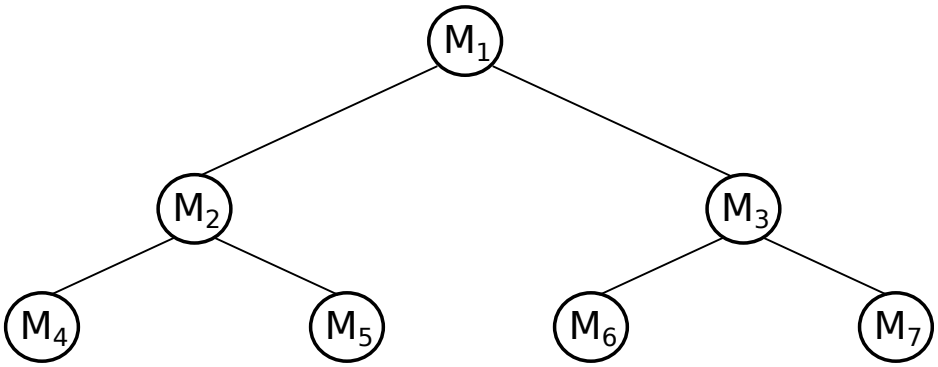


Abbildung 3.8.: k-d-Suchbaum

Aus Geschwindigkeitsgründen benutzt SIFT einen approximierten k-d-Algorithmus. Der Algorithmus wird als *Best-Bin-First* (BBF) [Beis und Lowe (1997)] bezeichnet.

Aussortieren falscher Korrespondenzen

Ein Merkmalsvektor ist so etwas wie ein „Fingerabdruck“ eines Merkmalspunktes. Obwohl SIFT- und SURF-Deskriptoren einen Merkmalspunkt sehr genau beschreiben, kann es beispielsweise durch Bildrauschen oder sehr ähnliche Merkmalsvektoren zu Fehlzuordnungen kommen. Um diese falschen Zuordnungen zu eliminieren, wird ein robustes Ausgleichsverfahren benötigt. Zu den bekannten Verfahren zählen zum Beispiel die *Hough-Transformation* [Hough (1962)] und *Random Sample Consensus* (RANSAC) [Fischler und Bolles (1981)]. Lowe empfiehlt die Hough-Transformation, in dieser Arbeit wird der RANSAC-Algorithmus verwendet.

RANSAC ist ein robuster Algorithmus, welcher verschiedene Ausgleichsverfahren unterstützt, um ein Modell innerhalb einer Reihe von Messwerten mit Ausreißern zu schätzen.

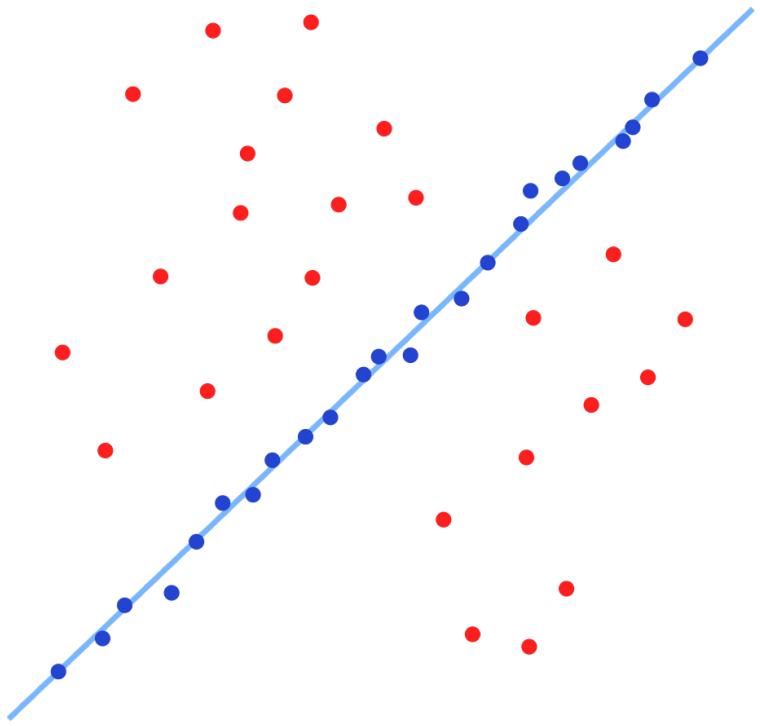


Abbildung 3.9.: RANSAC zum Anpassen einer Geraden an Datenpunkte

Um alle Korrespondenzpaare richtig zuzuordnen, muss die Nachbarsuche in beide Richtungen durchgeführt werden, von links nach rechts und von rechts nach links, bevor RANSAC angewendet wird. Die Schnittmenge enthält dann nur noch korrekte Zuordnungen.

(a) (b)

Abbildung 3.10.: Rektifizierte Aufnahmen einer Stereokamera (a) links (b) rechts

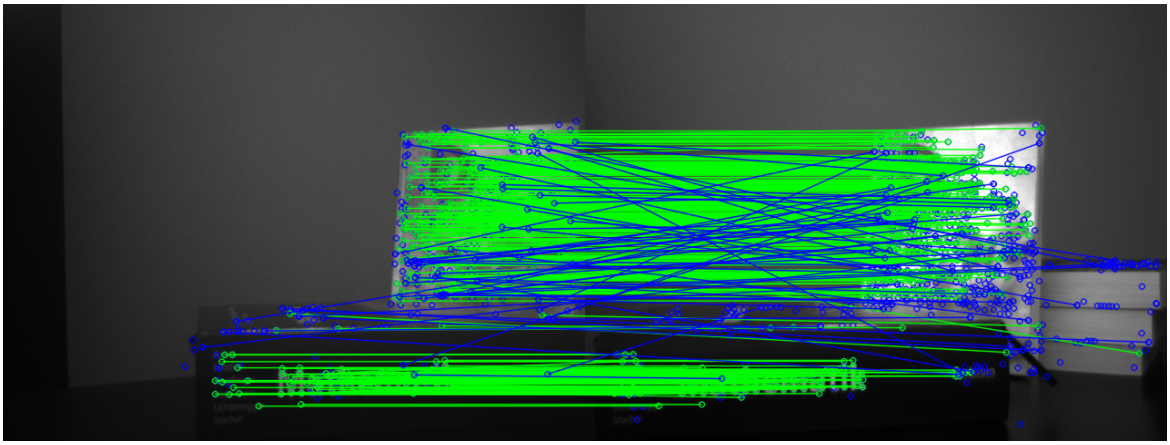


Abbildung 3.11.: SIFT-Merkmale: richtige Zuordnungen (grün) durch RANSAC gefilterte Falschzuordnungen (blau)

	SIFT	SURF
Bildauflösung	640x480	640x480
Merkmale Bild1	680	330
Merkmale Bild2	961	580
Korrespondenzen	377	190
Ausführungszeit	3486.2ms	448.017ms

Tabelle 3.1.: Ergebnisse für Aufnahme 3.10

3.2. Bildsegmentierung

Mit den im ersten Schritt gefundenen robusten Punkten, wird das erste (linke) Bild mit Hilfe der *Delaunay-Triangulation* in kleine Dreieckssegmente aufgeteilt. Für alle Pixel, die sich innerhalb eines Segmentes befinden, werden in einem späteren Schritt, mit Hilfe der *projektiven Transformation*, die korrespondierenden Pixel im zweiten (rechten) Bild ermittelt.

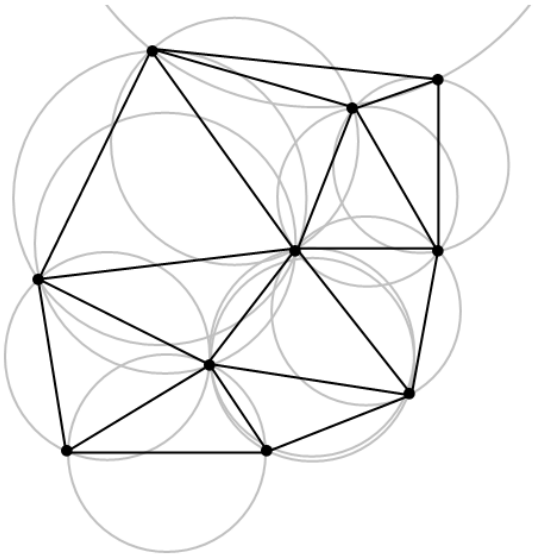


Abbildung 3.12.: Delaunay-Triangulation

Die Delaunay-Triangulation [Boris Nikolajewitsch Delone, 1934] ist ein Verfahren, das aus einer gegebenen Punktmenge ein Dreiecksnetz erstellt. Legt man einen Kreis auf drei beliebige Punkte des Netzes, dann darf innerhalb des Kreises kein anderer Punkt enthalten sein. Diese Eigenschaft wird als *Umkreisbedingung* bezeichnet. Es gibt mehrere Ansätze um eine Delaunay-Triangulation durchzuführen. [Su und Drysdale (1995)] gibt eine gute Übersicht über die einzelnen Verfahren.

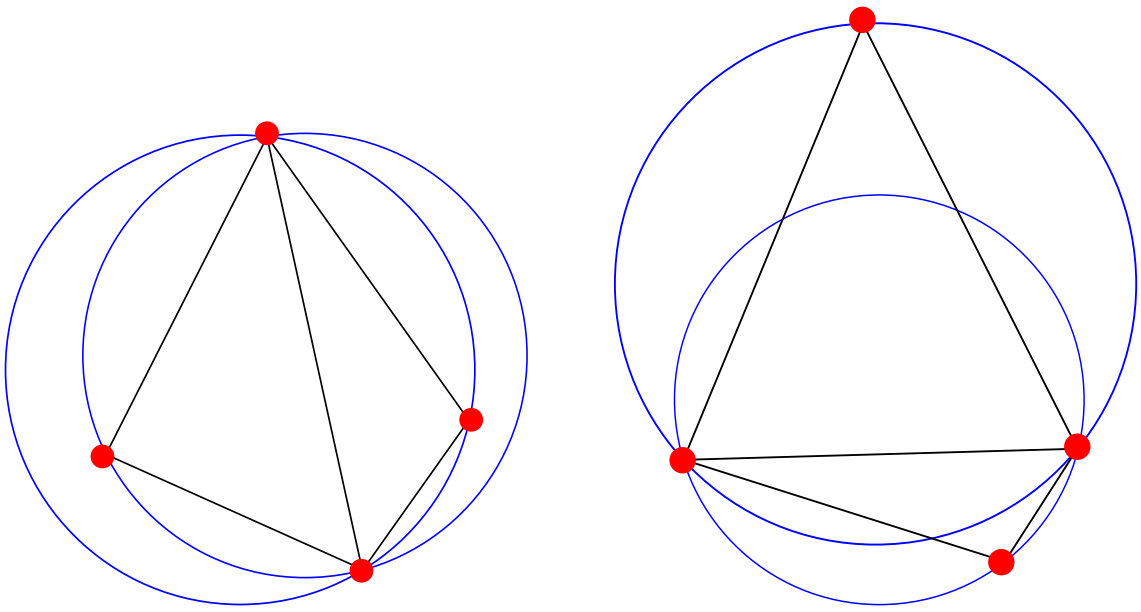


Abbildung 3.13.: Delaunay-Triangulation: (a) erfüllt die Umkreisbedingung nicht, bei (b) ist die Umkreisbedingung erfüllt

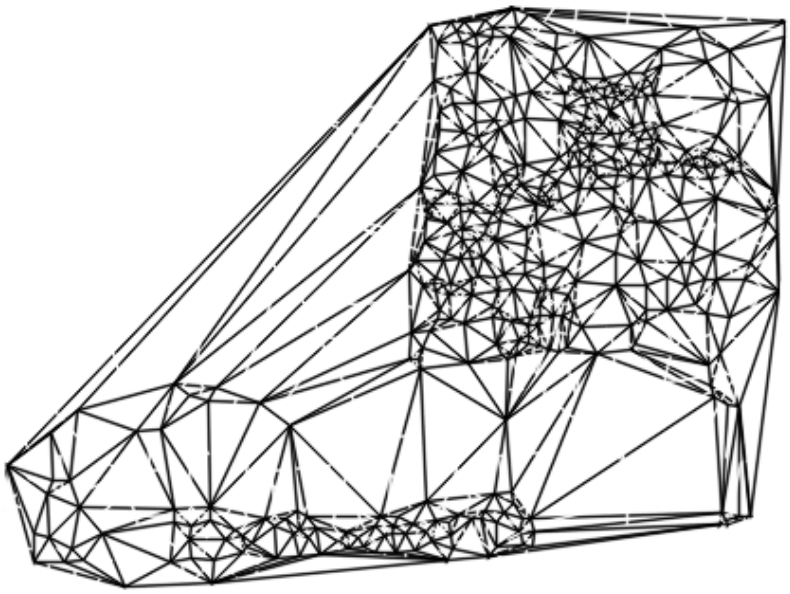


Abbildung 3.14.: Delaunay-Triangulation für das Bild 3.10(a)

3.3. Validierung

Die dynamische Programmierung verlässt sich darauf, dass die Reihenfolgebedingung (2.5.3) erfüllt ist. Das heißt, man geht davon aus, dass korrespondierende Pixel im linken und rechten Bild, in der gleichen Zeile, in gleicher Reihenfolge, versetzt um einen konstanten Disparitätswert liegen. Dies ist nicht der Fall, denn die Disparität hängt von der Entfernung eines 3D-Objektes zur Kamera ab (2.4.4, 2.4.5). Deshalb muss der Disparitätswert dynamisch angepasst werden. Oft werden dafür Kantenfilter eingesetzt. Gehört ein Pixel zu einer Kante, dann geht man davon aus, dass an dieser Stelle ein Disparitätssprung stattfindet. Die Kostenfunktion (2.5.1) muss dann entsprechend angepasst werden. Doch diese Methode versagt spätestens, wenn eine Kante zum Muster desselben Objektes gehört. Es ist also sehr schwierig ein allgemeines Kriterium für das Anpassen des Disparitätswertes zu finden.

Das Transformieren der Delaunay-Dreiecke von einem in das andere Bild, mit Hilfe der projektiven Transformation, ist eine einfache und recheneffiziente Methode um genaue Disparitätswerte für ganze Pixelbereiche zu berechnen. Die projektive Transformation wird im nächsten Abschnitt genau beschrieben.

Um eine projektive Transformation von Punkten des linken Bildes auf das rechte Bild durchzuführen, ist es wichtig, dass alle Punkte im linken Bild lokal auf derselben Raumebene liegen. Es würde sonst zu falschen Korrespondenzzuordnungen kommen (siehe Abbildung 3.15), da durch die andere Perspektive 3D-Objekte auf verschiedene Bereiche in den beiden Bildern (2D-Ebenen) abgebildet werden.

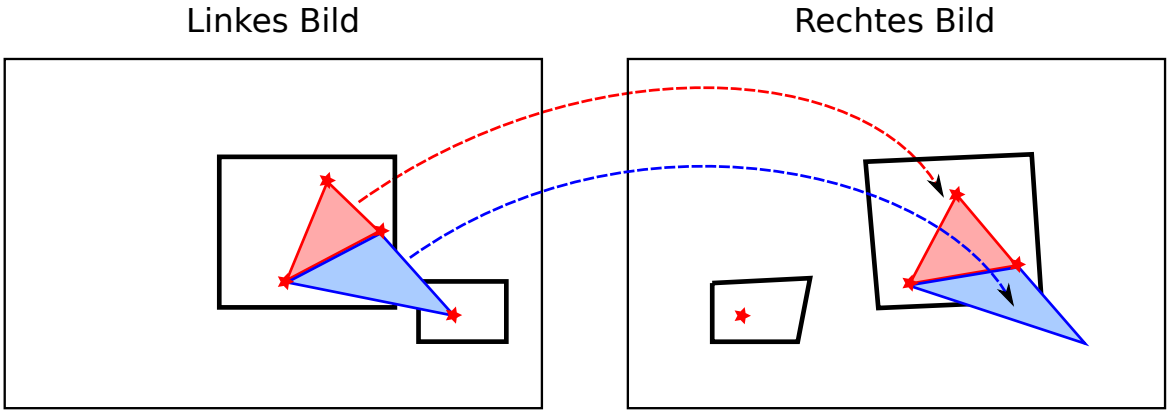


Abbildung 3.15.: Das rote Dreieck ist gültig, die Transformation des blauen Dreiecks führt zu falschen Korrespondenzzuordnungen.

Aus diesem Grund müssen Dreiecke, die die oben genannte Bedingung nicht erfüllen, vor der Transformation aussortiert werden, da die zugehörigen Punkte auf unterschiedlichen Raum-

ebenen liegen. Je kleiner ein Dreieck ist, desto größer ist die Wahrscheinlichkeit, dass alle Punkte des Dreiecks auf demselben Objekt liegen. Ein Dreieck wird geprüft, indem durch Triangulation (2.4.4) die Entfernung zu allen drei Eckpunkten berechnet wird. Der Unterschied der drei Entfernungen darf einen bestimmten Schwellwert nicht überschreiten. Dadurch wird geschätzt, ob die Eckpunkte auf demselben Objekt liegen könnten.

Des Weiteren ist es wichtig, dass korrespondierende Dreiecke im rechen Bild auf derselben lokalen Ebene liegen wie die Dreiecke im linken Bild. Nehmen wir an, wir betrachten zwei unterschiedliche Objekte (Abbildung 3.16). Der kleine Kasten befindet sich näher an der Kamera als der Große. Auf beiden Objekten wurden robuste Merkmale gefunden. Die Delaunay-Triangulation wird nur im linken Bild durchgeführt. Durch die Umkreisbedingung ist sichergestellt, dass sich keine Merkmalspunkte innerhalb eines Dreiecks befinden. Da in der rechten Aufnahme keine Delaunay-Triangulation stattgefunden hat, ist die Umkreisbedingung dort nicht garantiert. Durch den anderen Blickwinkel, wird im rechten Bild der Kasten, der sich weiter hinten befindet, durch den kleineren Kasten verdeckt. Eine Transformation des Dreiecks aus dem linken Bild, das durch die Merkmalspunkte des großen Kastens aufgespannt wird, würde zu falschen Korrespondenzzuordnungen führen. Die Transformation des Dreiecks, das sich auf dem kleinen Kasten befindet, wäre aber gültig.

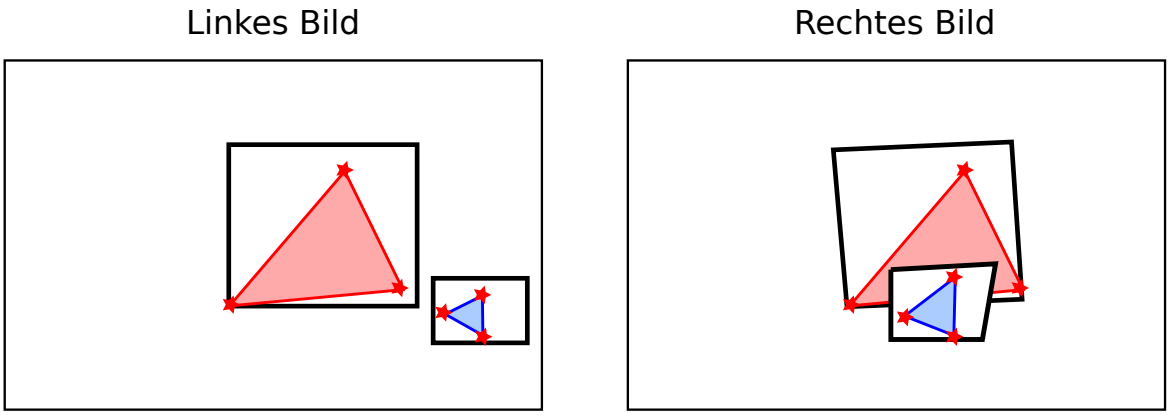


Abbildung 3.16.: Im rechten Bild ist die Umkreisbedingung nicht erfüllt

Dreiecke, deren korrespondierende Dreiecke einen Merkmalspunkt enthalten, können also schon vor dem Test auf lokale Ebenheit ausgeschlossen werden.

Es könnte auch der Fall eintreffen, dass das überdeckende Objekt selbst keine Merkmale enthält (siehe Abbildung 3.17).

In diesem Fall wird für jedes Pixel eines zu transformierenden Dreiecks geprüft, ob das korrespondierende Pixel einen großen Intensitätsunterschied aufweist.

$$I_p \approx I_{p'} \tag{3.12}$$

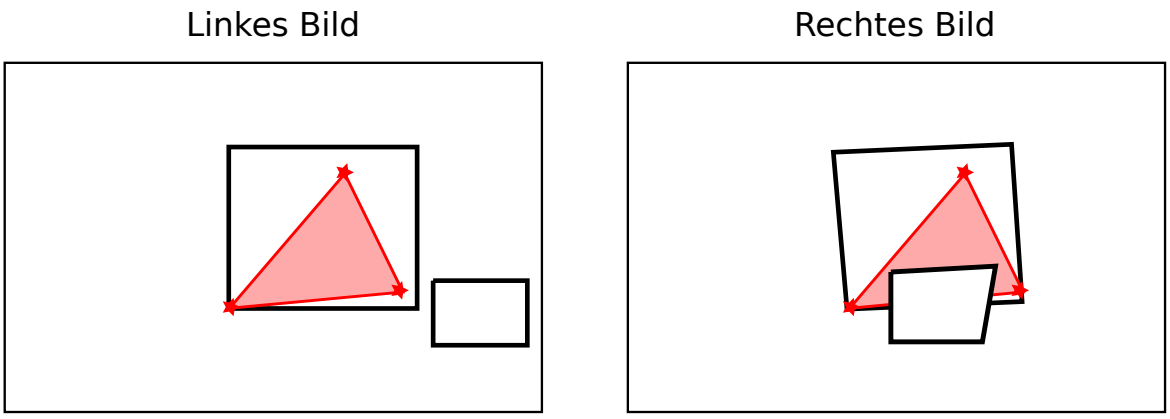


Abbildung 3.17.: Das überdeckende Objekt im rechten Bild enthält keine Merkmalspunkte

Gleichung 3.12 besagt, dass zwei korrespondierende Punkte, ähnliche Disparitätswerte besitzen müssen. Der Unterschied der beiden Werte darf einen Schwellwert nicht überschreiten.

3.4. Projektive Transformation

In diesem Schritt werden für alle Pixel der validierten Delaunay-Segmente, durch Transformation der Koordinaten, die korrespondierenden Pixel bestimmt.

Eine Projektion, in der Mathematik, ist eine umkehrbare Abbildung P eines Vektorraumes V in sich selbst.

$$P : V \mapsto V \tag{3.13}$$

Das Bild der Projektion ist dann eine Teilmenge von V. Die Punkte die vor der Projektion auf einer Geraden liegen, liegen auch nach der Projektion auf einer Geraden.

Wird ein Objekt von zwei Kameras aus unterschiedlichen Blickwinkeln aufgenommen, entstehen durch die Projektion eines 3D Objektes auf eine 2D Bildebene, in den Bildern unterschiedliche perspektivische Verzerrungen. Bei diesen Verzerrungen handelt es sich um *projektive Transformationen*, bei zweidimensionalen Flächen auch als *Homographien* bezeichnet. Die Aufnahmen lassen sich durch eine projektive Rücktransformation wieder entzerren.

Eine projektive Transformation enthält Translation, Rotation, Skalierung, Scherung und Perspektive. Sie erhält Geraden und das Doppelverhältnis zwischen vier kollinearen Punkten

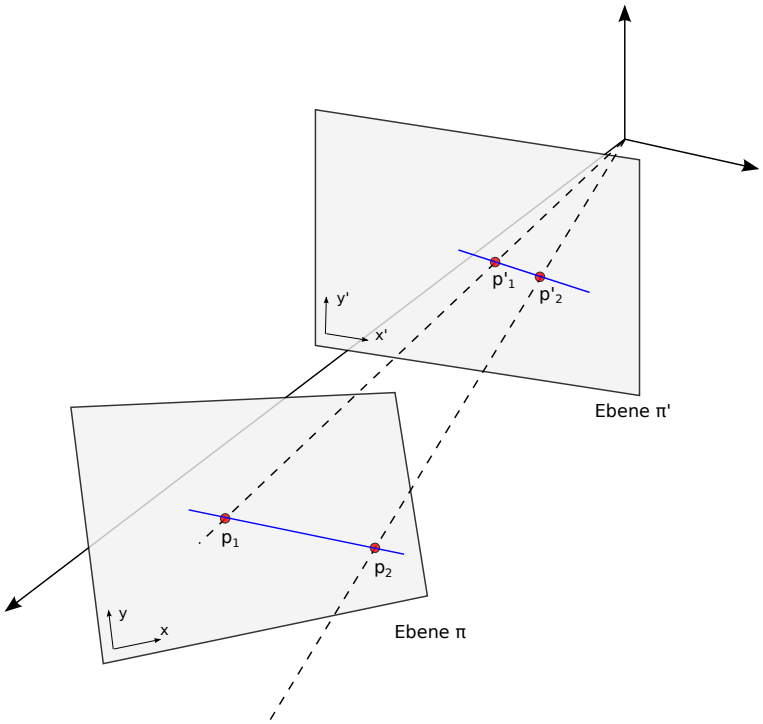


Abbildung 3.18.: Projektive Transformation

(Punkten, die auf einer Geraden liegen). Zu einem Punkt p lässt sich sein korrespondierender, verzerrter Punkt p' berechnen, indem p mit der $3x3$-Homographiematrix H multipliziert wird. Dabei gilt $H \in \mathbb{R}^{3x3}$ und $p, p' \in \mathbb{P}^2$.

$$p' = Hp \tag{3.14}$$

H bildet also $\mathbb{P}^2$ nach $\mathbb{P}^2$ ab.

$$H : \mathbb{P}^2 \mapsto \mathbb{P}^2 \tag{3.15}$$

Da p und p' Punkte des projektiven Raumes sind und dadurch eine homogene Koordinate besitzen, gilt $p = (x, y, 1)^T$ und $p' = (x', y', k')^T$. Nach dem Einsetzen erhält man die folgende Darstellung:

$$\begin{pmatrix} x' \\ y' \\ k' \end{pmatrix} = \begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = \begin{pmatrix} h_{11}x + h_{12}y + h_{13} \\ h_{21}x + h_{22}y + h_{23} \\ h_{31}x + h_{32}y + h_{33} \end{pmatrix} \tag{3.16}$$

Nach dem Ausmultiplizieren wird der 3D-Vektor p' durch Teilen durch die homogene Koordinate k' normalisiert (in euklidische Koordinaten zurückgewandelt). Es ergeben sich folgende

Gleichungen:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{k'} \Rightarrow h_{11}x + h_{12}y + h_{13} - x'k' = 0 \tag{3.17}$$

$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{k'} \Rightarrow h_{21}x + h_{22}y + h_{23} - y'k' = 0 \tag{3.18}$$

$k' = h_{31}x + h_{32}y + h_{33}$ wird in die Gleichungen 3.17 und 3.18 eingesetzt und man erhält:

$$h_{11}x + h_{12}y + h_{13} - x'(h_{31}x + h_{32}y + h_{33}) = 0 \tag{3.19}$$

$$h_{21}x + h_{22}y + h_{23} - y'(h_{31}x + h_{32}y + h_{33}) = 0 \tag{3.20}$$

Durch Umformung ergibt sich ein Gleichungssystem der Art $A * h = 0$:

$$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 & -x'x & -x'y & -x' \\ 0 & 0 & 0 & x & y & 1 & -y'x & -y'y & -y' \end{pmatrix} \begin{pmatrix} h_{11} \\ h_{12} \\ . \\ . \\ . \\ h_{33} \end{pmatrix} = 0 \tag{3.21}$$

Bei n Punktepaaren hat A die Dimension $2n \cdot 9$. Um das Gleichungssystem zu lösen, muss A den Rang 8 haben, d.h. es werden 8 Punktkorrespondenzen benötigt. Mit Hilfe der Singulärwertzerlegung (siehe Anhang B) lässt sich die Homographiematrix H auch mit nur 4 Punktkorrespondenzen berechnen.

Da man für die Berechnung der Homographiematrix mindestens 4 Punktkorrespondenzen benötigt, werden in [Chen u. a. (2008)] beispielsweise mehrere benachbarte Dreiecke zu einer Region zusammengefasst. Dann wird geprüft, ob alle zusammengefassten Dreiecke lokal auf derselben Ebene liegen, denn nur dann kann die projektive Transformation robust angewendet werden. Anschließend wird für alle gültigen Regionen separat eine Homographiematrix berechnet.

In dem Verfahren das in dieser Arbeit vorgestellt wird, wird die Homographiematrix nur einmal, anhand aller im ersten Schritt des Verfahrens ermittelten Punktkorrespondenzen, berechnet. Diese Vorgehensweise vereinfacht und beschleunigt das Verfahren und es können auch die Dreiecke transformiert werden, deren Nachbardreiecke nicht auf derselben lokalen Ebene liegen.

3.5. Dynamische Programmierung

In diesem abschließenden Schritt, werden die Ergebnisse der vorangegangenen Schritte mit Hilfe *dynamischer Programmierung* optimiert.

Abbildung 3.21 verdeutlicht die Vorgehensweise bei der Anwendung dynamischer Programmierung auf Bilddaten. Zwei Grauwertbilder werden dort zeilenweise miteinander verglichen. Das Ziel ist es, die beste Zuordnung der Punkte des einen Bildes zu den Punkten des zweiten Bildes zu finden. Es wird zunächst angenommen, dass die Epipolarbedingung und die Reihenfolgebedingung (2.4.2, 2.5.3) erfüllt sind. Also, dass korrespondierende Punkte in beiden Bildern in der gleichen Zeile y, versetzt um einen konstanten Disparitätswert d, liegen.

3.5.1. Aufstellen der Kostenmatrix

Im Beispiel (Abbildung 3.21) wird als Vergleichskriterium die *Summe absoluter Differenzen* (SAD) (2.5.1) mit dem Disparitätswert $d = d_{max} = 2$ benutzt. Gleichung 3.22 beschreibt die Berechnungsvorschrift. I_R und I_L stehen hier für die beiden Aufnahmen eines Stereobildpaares, i und j geben den Spaltenindex und y gibt den Zeilenindex im jeweiligen Bild an.

$$SAD(y, d) = \Sigma_j \Sigma_i |I_R(y, i) - I_L(y, j + d)| \tag{3.22}$$

Die Korrelationsergebnisse werden in einer sogenannten *Kostenmatrix* C abgelegt (siehe Abbildung 3.21). Die Matrix wird aufgestellt, indem links oben beginnend, spaltenweise iteriert wird. So wird Schritt für Schritt jeder Punkt der Zeile y aus dem rechten Bild mit jedem Punkt der gleichen Zeile aus dem linken Bild verglichen. Der Zeilenindex in C gibt die Position eines Punktes in der Zeile y des rechten Bildes und der Spaltenindex die Position des Punktes in der Zeile y des linken Bildes an. Abbildung 3.22 zeigt die Iterationsschritte für die ersten beiden Spalten.

3.5.2. Beschränkung auf gültige Bildbereiche

Wie im Kapitel 2.4 beschrieben, bildet ein Stereobildpaar eine Szene aus zwei unterschiedlichen Blickwinkeln ab. Die verschiedenen Ansichten einer Szene werden durch die Ausrichtung und den Abstand der beiden Kameras voneinander verursacht. Die betrachteten Objekte einer Szene werden im linken und rechten Bild versetzt um einen Disparitätswert d abgebildet. Die Disparität zwischen zwei korrespondierenden Punkten hängt von der Entfernung des Objektes, auf dem die Punkte liegen, zur Kamera ab. Je näher sich das betrachtete Objekt an der Kamera befindet, desto größer ist die Disparität. Abbildung 3.19 demonstriert

dieses Szenario. Der Einfachheit halber wird hier angenommen, dass die optischen Achsen der beiden Kameras parallel zueinander sind und in derselben Ebene liegen.

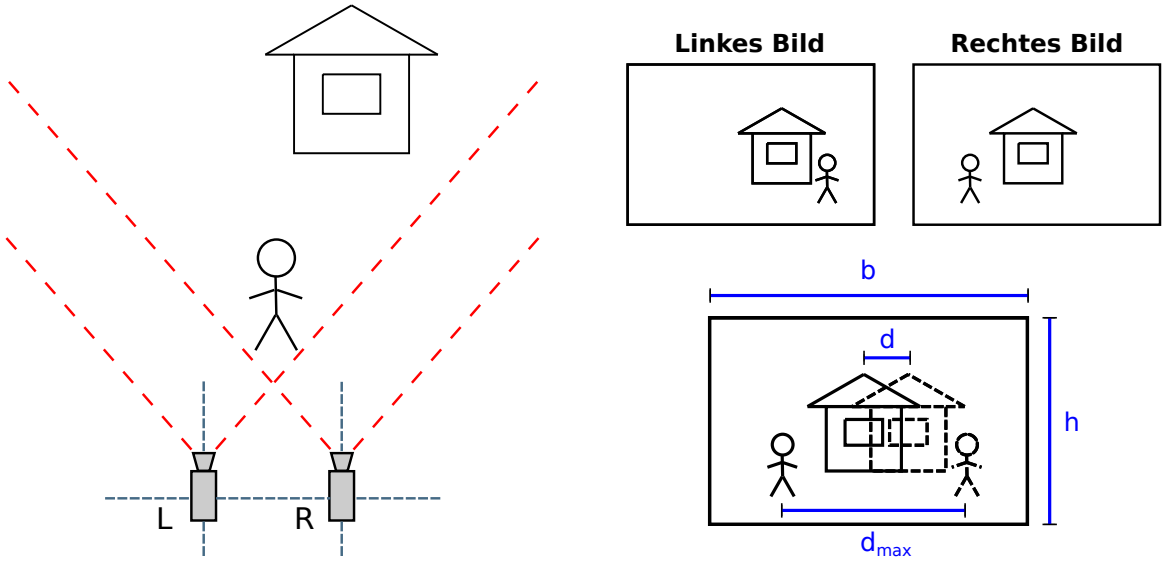


Abbildung 3.19.: Unterschiedliche Ansichten eines Stereobildes

Da eine Szene aus verschiedenen Perspektiven aufgenommen wird, ist nur ein Teilausschnitt der Szene in beiden Bildern sichtbar. Dieser Ausschnitt ist die Schnittmenge der von beiden Kameras erfassten Bereiche (siehe Abbildung 3.19). Während der anschließenden Rektifizierung (2.4.3) der Bilder wird dieser Bereich, durch Umordnung der Pixel, möglicherweise noch weiter eingeschränkt.

Im Beispiel 3.21 werden nur Bereiche berücksichtigt, die in beiden Bildern gültig sind. Diese Einschränkung beschleunigt die Berechnung, da nicht alle Punkte einer Zeile miteinander verglichen werden müssen und führt zu genaueren Ergebnissen, da nur gültige Bereiche verglichen werden. Die gültigen Bereiche lassen sich mit Hilfe des maximalen Disparitätswertes bestimmen. Abbildung 3.20 veranschaulicht die Einschränkung des Bildmaterials auf gültige Bereiche anhand einer realen Stereoaufnahme.

Abbildung 3.20.: Einschränkung der Stereoaufnahme auf den Bereich, der von beiden Kameras erfasst wird

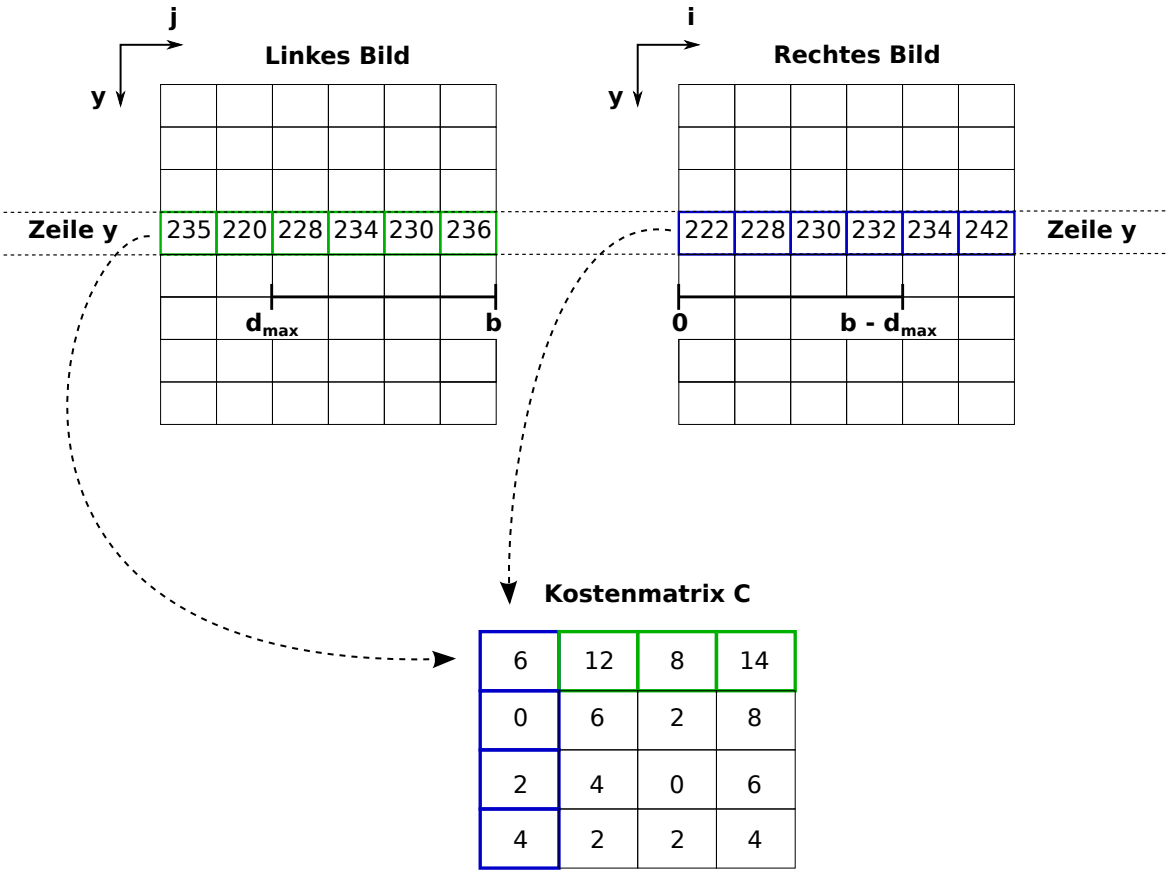


Abbildung 3.21.: Aufstellen der Kostenmatrix

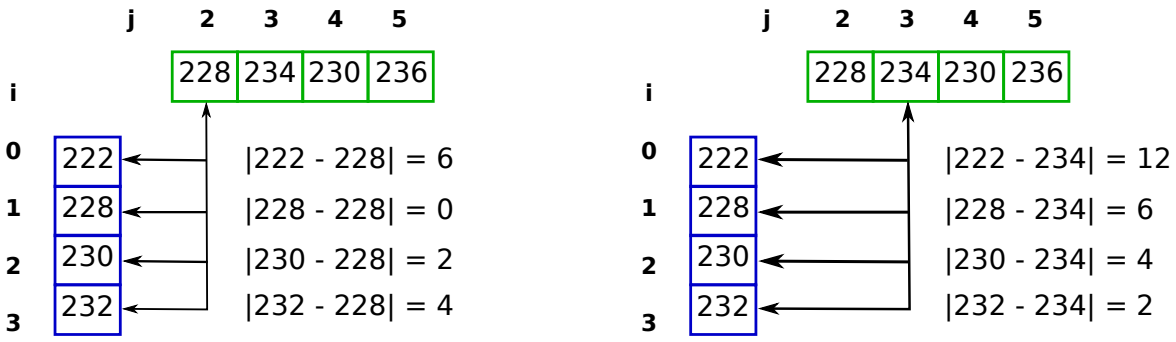


Abbildung 3.22.: Iterationsschritte für die ersten beiden Spalten beim Aufstellen der Kostenmatrix

3.5.3. Kostenaggregation

Die Kostenmatrix C kann als ein Graph betrachtet werden (siehe Abbildung A.1). Die Einträge der Kostenmatrix repräsentieren die Kantengewichte des Graphen. Die Knoten des Graphen werden durch die Position (i, j) der Einträge in C repräsentiert. Das Korrespondenzproblem wird gelöst, indem ein Pfad durch die Matrix gesucht wird, auf dem die Summe der Kosten am geringsten ist.

Um den besten Pfad zu finden, muss jeder Eintrag den Weg zu seinem besten Vorgänger kennen und die Information über die Kosten des besten Vorgängers beinhalten. Der beste Vorgänger ist der Vorgänger mit den geringsten Kosten. Dazu wird von links oben ausgehend, spaltenweise über C iteriert. Zu jedem Eintrag $C(i, j)$ wird der Wert seines besten Vorgängers addiert.

$$C(i,j) = C(i,j) + min \begin{cases} C(i-1,j) \\ C(i,j-1) \\ C(i-1,j-1) \end{cases} \tag{3.23}$$

Zur Verdeutlichung wird die Vorgehensweise in Abbildung 3.23 dargestellt.

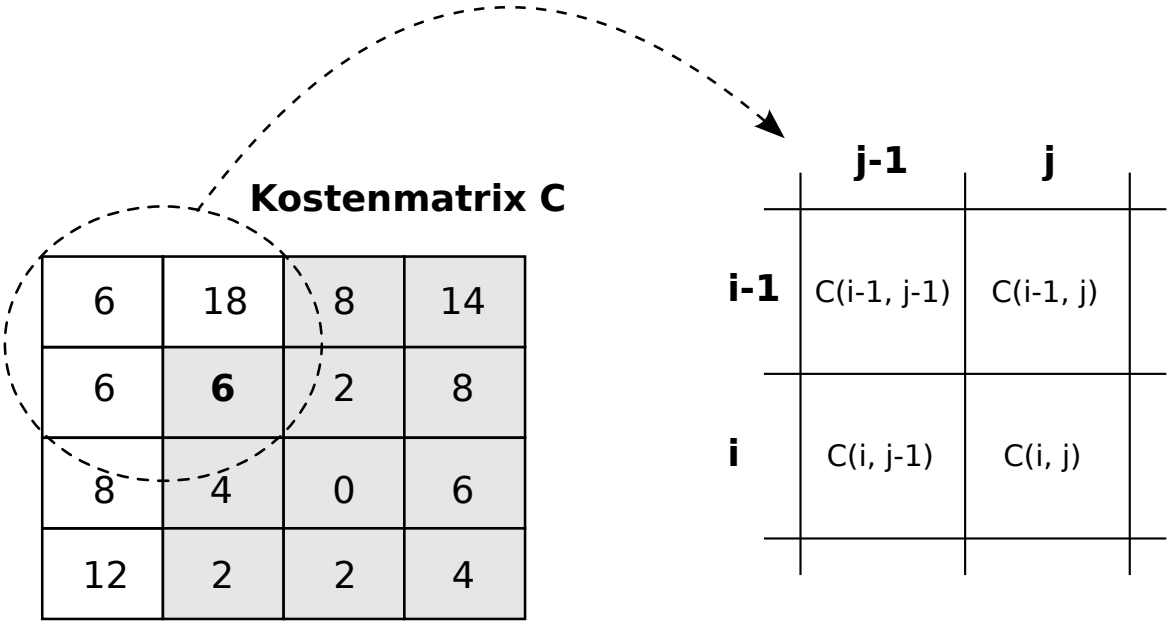


Abbildung 3.23.: Kostenaggregation in der Kostenmatrix: Die Einträge auf dem weißen Hintergrund wurden bereits berechnet. Der aktuell zu berechnende Wert ist fett hervorgehoben. Der neue Wert wird berechnet durch $6 + min(18, 6, 6) = 6 + 6 = 12$

3.5.4. Backtracking

Das Suchen des besten Pfades durch die Matrix wird als *Backtracking* bezeichnet. Man fängt rechts unten in der aggregierten Kostenmatrix an und folgt dem Pfad in Richtung des Vorgängers mit den geringsten Kosten. Für mehrere Vorgänger mit gleichen Kosten wird die diagonale Richtung gewählt. Bei gleichen Kosten in die horizontale und vertikale Richtung und größeren Kosten in die diagonale, wird die horizontale Richtung bevorzugt.

Abbildung 3.24 demonstriert die Rückwärtssuche anhand der weiter oben berechneten Matrix.

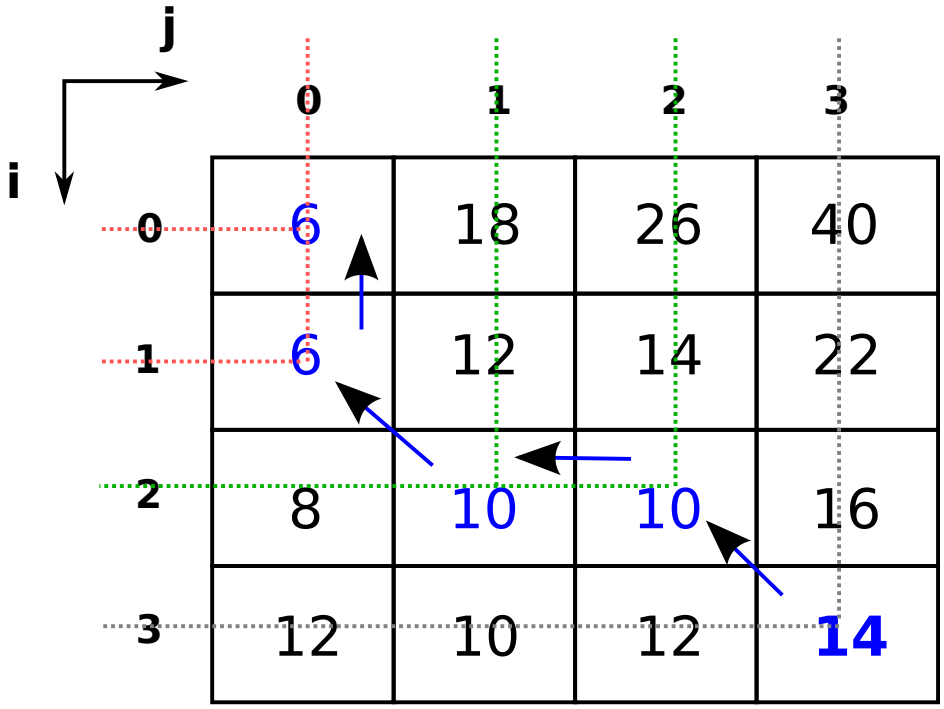


Abbildung 3.24.: Backtracking in der aggregierten Kostenmatrix. Der Startpunkt ist fett hervorgehoben, die gepunkteten Linien zeigen die Pixelzuweisungen der linken und rechen Bildzeile.

Verdeckungen durch im Vordergrund liegende Objekte machen sich bei der Rückwärtssuche dadurch bemerkbar, dass mehrere Pixel in einem Bild, einem einzigen Pixel in dem anderen Bild zugeordnet werden. In Abbildung 3.24 sind Verdeckungen im rechten Bild grün und Verdeckungen im linken Bild rot gekennzeichnet. An diesen Stellen sind aufgrund der Verdeckungen einfach keine korrespondierenden Punkte in den jeweiligen Ansichten vorhanden. Dadurch beinhaltet die generierte Disparitätskarte an diesen Stellen falsche Informationen.

Die Werte der Disparitätskarten $DisparityMapL$ und $DisparityMapR$ für das linke und rechte Bild, werden anhand der horizontalen und vertikalen Unterschiede zur Diagonalen der aggregierten Kostenmatrix, die für die Bildzeile y erstellt wurde, berechnet:

$$DisparityMapL(y,j) = |i - j| \tag{3.24}$$

$$DisparityMapR(y,i) = |j - i| \tag{3.25}$$

Bei einer nxn großen Kostenmatrix C, fängt man also im Punkt $C_{aggregiert}(n,n)$ mit der Rückwärtssuche an. Durch diese Vorgehensweise kommt es zu falschen Informationen in der Disparitätskarte, falls Zeilenabschnitte aus denen C generiert wurde, am Ende einen Disparitätswert $d \neq 0$ besitzen. Die Ergebnisse sind trotzdem akzeptabel, wenn die Bilder relativ groß im Vergleich zu der durchschnittlichen Disparität sind.

Dieser Fehler lässt sich umgehen, indem mit dem Backtracking in der Zeile n und der Spalte, die vom Punkt (n, n) ausgehend die geringsten Kosten enthält, angefangen wird. Abbildung 3.25 zeigt diesen Ansatz an dem oberen Beispiel.

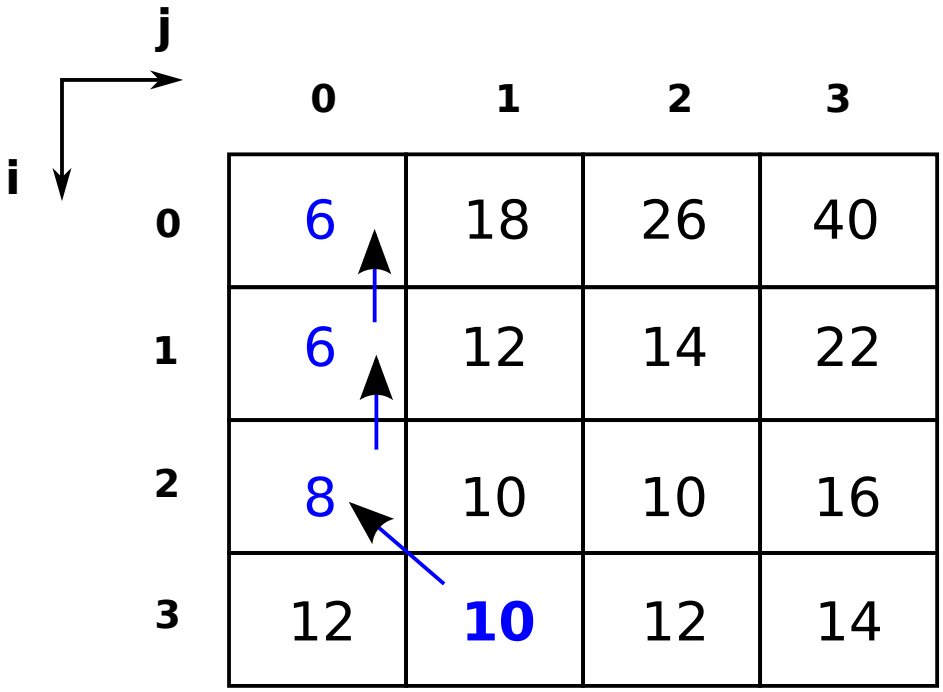
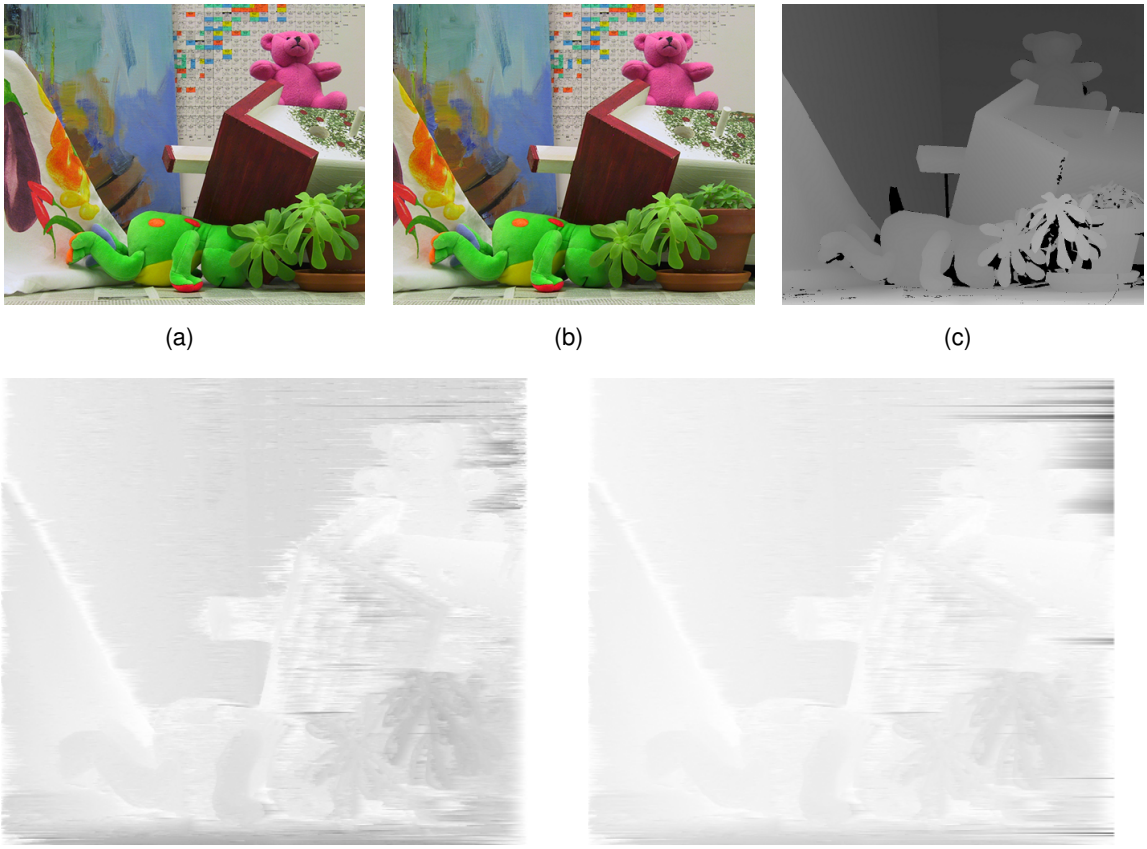


Abbildung 3.25.: Backtracking mit verlagertem Startpunkt

Diese Vorgehensweise liefert ein etwas besseres Ergebnis. Bei diesem Ansatz muss für jede Bildzeile in der untersten Zeile der aggregierten Kostenmatrix, nach dem Minimum gesucht werden. Trotzdem kann dieser zusätzliche Rechenaufwand das Verfahren sogar beschleunigen, da der Weg durch die Matrix dann in der Regel kürzer ist.

Abbildung 3.26 demonstriert die Unterschiede zwischen den beiden Backtrackingstrategien anhand einer bekannten Stereoaufnahme [Scharstein und Szeliski (2003)].



(a) (b) (c)

(d) (e)

Abbildung 3.26.: (a) linkes Bild (b) rechtes Bild (c) Ground Truth (3.7) (d) Backtrackingstartpunkt rechts unten (e) Backtrackingstartpunkt verlagert

3.5.5. Ground Control Points

Hochrobuste korrespondierende Punkte werden als *Ground Control Points* (GCP) bezeichnet. Durch geschickten Einsatz dieser Punkte kann die Qualität und Geschwindigkeit der dynamischen Programmierung erheblich gesteigert werden [Bobick und Intille (1999), Gong und Yang (2003), Kim u. a. (2005)].

Eine einfache Strategie um die im ersten Schritt des vorgestellten Verfahrens gefundenen robusten Korrespondenzen (GCP) auszunutzen ist, den Algorithmus der dynamischen Programmierung so zu beeinflussen, dass der Pfad, beim Backtracking in der aggregierten Kos-

tenmatrix, durch die gefundenen Ground-Control-Points verläuft. Das Backtracking lässt sich beeinflussen, indem die Kosten für einen GCP in der Kostenmatrix auf 0 gesetzt werden und allen anderen Pfaden in derselben Spalte sehr hohe Kosten zugewiesen werden (als Beispiel siehe Abbildung 3.27).

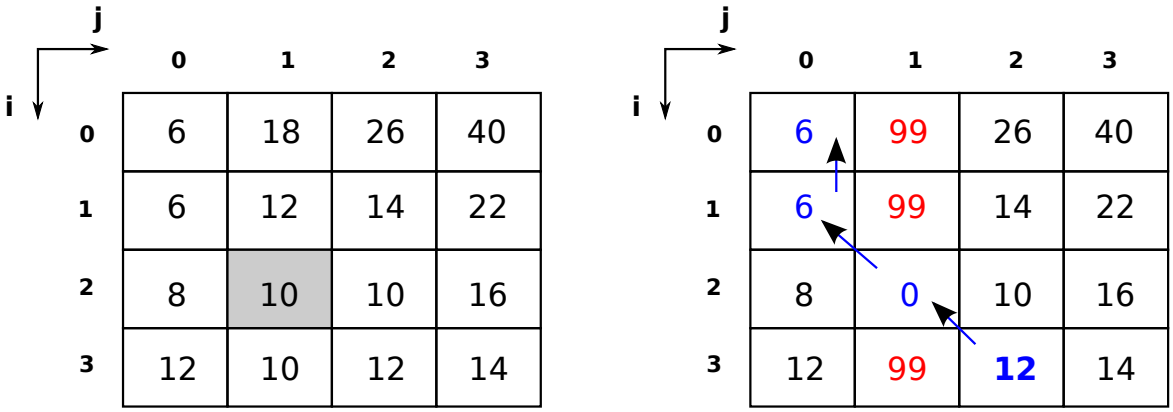


Abbildung 3.27.: Ground Control Points: Der Eintrag in der zweiten Spalte und der dritten Zeile der ersten Matrix wurde als GCP identifiziert. Die Kosten für diesen Eintrag werden auf 0 gesetzt. Allen anderen Pfaden in derselben Spalte wird der hohe Wert 99 zugewiesen. Auf diese Weise wird sichergestellt, dass der Pfad beim Backtracking durch den GCP verläuft.

3.6. Eigenschaften und mögliche Optimierungen

Das oben beschriebene Verfahren ist sehr modular implementiert (siehe Kapitel 4). Die einzelnen Teilschritte des Verfahrens lassen sich leicht durch andere Implementierungen ersetzen. Somit kann das gesamte Verfahren jederzeit durch eine qualitativ bessere oder schnellere Teillösung optimiert werden.

Das Verfahren orientiert sich an [Chen u. a. (2008)]. Die Schritte *projektive Transformation* (3.4) und dadurch auch die *Validierung* der Dreiecke (3.3) wurden vereinfacht. Des Weiteren können verschiedene Merkmalsextraktionsverfahren benutzt werden. Besonderer Wert wurde auf die Implementierung der dynamischen Programmierung gelegt. Viele vorhandene Algorithmen wurden untersucht und die besten Eigenschaften wurden übernommen. Einzelne Optimierungsstufen lassen sich über Parameter aktivieren oder abschalten. Dadurch kann man sich zwischen Qualität und Geschwindigkeit entscheiden und der Algorithmus lässt sich auf eine gegebene Aufgabenstellung anpassen.

In folgenden Abschnitten werden Optimierungsmöglichkeiten für die dynamische Programmierung kurz vorgestellt. Einige Vorschläge sind übernommen aus [Forstmann u. a. (2004)].

3.6.1. Parallelisierung

Dynamische Programmierung verarbeitet zwei Bilder Zeile für Zeile. Für jedes Zeilenpaar wird eine Kostenmatrix erstellt. Nach der Aggregation der Kosten in der Kostenmatrix wird der beste Pfad durch die Matrix gesucht.

Bei großen Bildern (lange Zeilen $\Rightarrow$ große Kostenmatrix) und einem Multicore-Prozessor kann die Berechnung der Zeilen paarweise auf die einzelnen Prozessorkerne verteilt werden (siehe Abbildung 3.28). In diesem Fall steigt die Berechnungsgeschwindigkeit fast[5] linear mit der Anzahl der Prozessorkerne. Bei Bildern mit einer sehr kleinen Auflösung kann die Geschwindigkeit durch den höheren Verwaltungsaufwand vermindert werden. Eine noch größere Geschwindigkeitssteigerung kann erreicht werden, wenn zusätzlich zum Prozessor[6] Grafikkarten[7] oder Programmierbare Logikbausteine[8] für die Berechnung verwendet werden [Wang u. a. (2006); Kalarot und Morris (2010)].

3.6.2. Ground Control Points

Ground Control Points können zur Stabilität und Geschwindigkeit des Algorithmus beitragen. Eine detailierte Beschreibung findet man im Abschnitt 3.5.5.

3.6.3. Verkleinerung der Kostenmatrix

Eine Beschränkung der Zeilenoptimierung nur auf die Bereiche, die in beiden Bildern vorhanden sind, führt in den jeweiligen Zeilen zu einer kleineren Kostenmatrix. Eine kleinere Kostenmatrix bedeutet geringerer Rechenaufwand. Dieser Schritt ist im Abschnitt 3.5.2 ausführlich beschrieben.

[5] dazu kommt der durch die Thread-Verwaltung verursachte Overhead
[6] *central processing unit* (CPU)
[7] *Graphics processing unit* (GPU)
[8] *Field Programmable Gate Array* (FPGA)

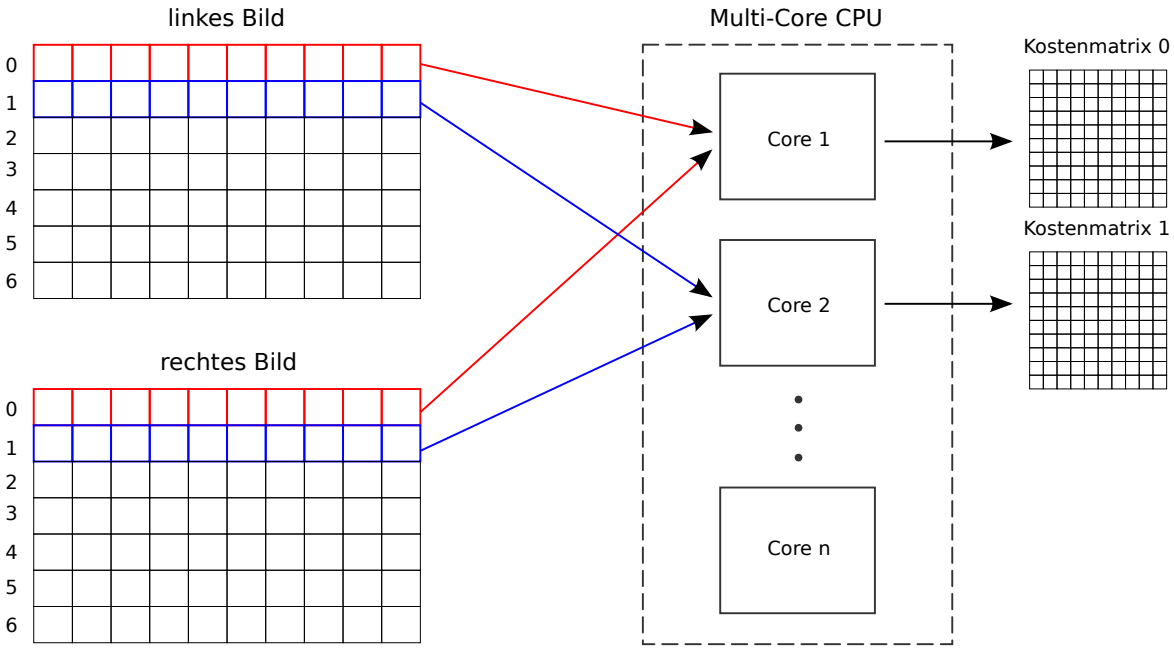


Abbildung 3.28.: Parallele Verarbeitung von Stereoaufnahmen auf einem Multicore-Prozessor

3.6.4. Auswahl einer geeigneten Kostenfunktion

Wesentlich für das Ergebnis ist auch die Wahl der Kostenfunktion. Kostenfunktionen werden im Abschnitt 2.5.1 behandelt.

3.6.5. Vertikale Optimierung

Der herkömmliche Ansatz dynamischer Programmierung verarbeitet zwei gleiche Bildzeilen ganz unabhängig von allen anderen Zeilen. Man geht davon aus, dass korrespondierende Punkte in der gleichen Zeile liegen, versetzt um einen bestimmten Disparitätswert. In realen Aufnahmen liegen korrespondierende Punkte aber oft in benachbarten Bildzeilen. Solche Fälle treten zum Beispiel durch ungenaue Kalibrierung der Kamera, durch unsaubere Rektifizierung oder durch Pixelinterpolationen auf. Diese Ungenauigkeiten können in der Regel nicht vermieden werden.

Deshalb kommt es in einer, mit Hilfe dynamischer Programmierung generierten, Disparitätskarte zu streifenförmigen Mustern entlang der Zeilen. Um das Problem zu lösen, müssen zusätzliche Informationen in die Berechnung einbezogen werden. Üblicherweise werden in einem zusätzlichen Schritt die Spalten der beiden Bilder optimiert. Das geschieht auf die

gleiche Art und Weise wie die Optimierung der Zeilen. Für jedes Spaltenpaar wird also eine Kostenmatrix erstellt und der beste Pfad durch die Matrix gesucht (siehe [Kim u. a. (2005)]). Das Ergebnis ist eine deutlich sauberere Disparitätskarte ohne horizontale Streifen.

3.6.6. Sub-Pixel-Auflösung

Dynamische Programmierung arbeitet pixelorientiert. Eine feinere Auflösung führt zu einer Disparitätskarte mit genauerer Tiefenauflösung. Um dieses zu erreichen, können die Bilder x mal in der Breite skaliert werden, die Höhe bleibt erhalten. Danach wird das skalierte Bildpaar in x-Richtung weichgezeichnet (blurring). Durch das Weichzeichnen werden die ursprünglichen Pixelwerte auf die neue Breite interpoliert. Die Auflösung ist dann x mal feiner als die bisherige.

Diese einfache Methode kann für jeden beliebigen Stereo-Algorithmus verwendet werden. Da sich durch die Methode für jedes Bildzeilenpaar eine größere Kostenmatrix ergibt, wird die Berechnungsgeschwindigkeit erheblich vermindert.

3.6.7. Doppelte Auswertung

Tiefensprünge werden in der Disparitätskarte auch besser aufgelöst, wenn die Kostenmatrix für ein Zeilenpaar zweimal berechnet wird. Einmal von links nach rechts und einmal von rechts nach links. Dazu müssen die Positionen der linken und rechten Zeile (siehe Abbildung 3.22) bei der Generierung der Matrix einmal getauscht werden. Die endgültige Disparitätskarte wird aus der Schnittmenge der beiden Teilergebnisse erzeugt.

3.6.8. Rauschunterdrückung

Aus verschiedenen Gründen, zum Beispiel durch schlechte Belichtung oder verursacht durch den Bildsensor der Kamera, können Aufnahmen verrauscht sein. Also nicht zum eigentlichen Bildinhalt gehörende, störende Informationen enthalten. Das störende Rauschen kann in einem Vorverarbeitungsschritt zum Beispiel durch Rauschfilter entfernt werden. Dieser Schritt ist eine allgemeine Maßnahme zur Verbesserung des Bildmaterials und ist nicht auf dynamische Programmierung beschränkt.

3.7. Ergebnisse

Zur Bestimmung der Qualität eines Stereoalgorithmus können sogenannte *Ground-Truth-Bilder* herangezogen werden. Diese Bilder werden unter Verwendung von *strukturiertem Licht* erzeugt [Scharstein und Szeliski (2003)]. Bei diesem Prozess werden bekannte Lichtmuster, meistens parallele Streifen, auf ein 3-dimensionales Objekt projiziert. Diese bekannten Muster erlauben die Vermessung der Objektoberfläche. Ein Ground-Truth-Bild repräsentiert die optimale Disparitätskarte einer Stereoaufnahme. Ein Algorithmus wird anhand der Unterschiede zwischen seiner Disparitätskarte und des Ground-Truth-Bildes klassifiziert.

Auf der *Middlebury Stereo Vision* Webseite [Scharstein und Szeliski] findet man eine Ranking-Datenbank vieler aktuellen Stereo-Algorithmen. Dort werden mehrere Datensätze von Stereoaufnahmen [Scharstein und Szeliski (2002, 2003); Scharstein und Pal (2007); Hirschmüller und Scharstein (2007)], inklusive zugehöriger Ground-Truth-Bilder zur Verfügung gestellt.

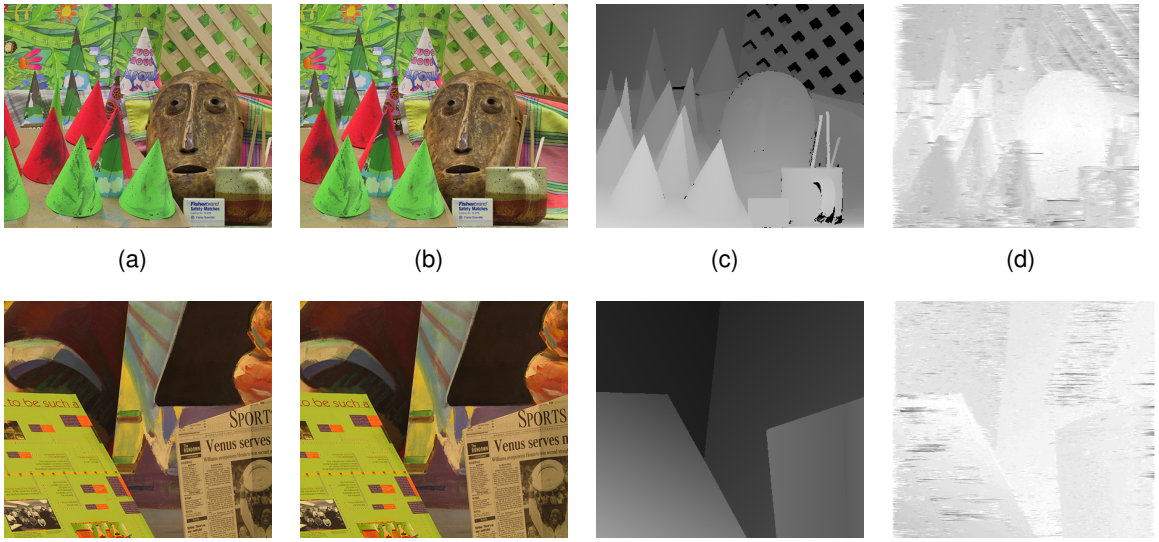
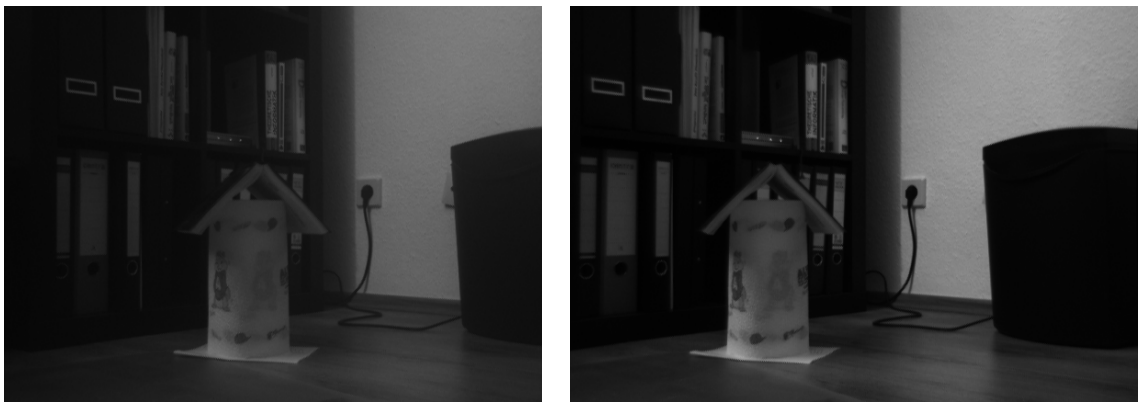


Abbildung 3.29.: Ergebnisse für zwei Stereoaufnahmen von der Middlebury Stereo Vision Webseite (ohne Optimierungen aus Kapitel 3.6). Oben *cones* [Scharstein und Szeliski (2003)] unten *venus* [Scharstein und Szeliski (2002)]. Die Reihenfolge der Abbildungen ist: linke Aufnahme, rechte Aufnahme, Ground Truth, Disparitätskarte

Middlebury bietet auch die Möglichkeit, den eigenen Algorithmus mit allen anderen in der Datenbank zu vergleichen. Bewertet wird die Qualität der Disparitätskarte, die Geschwindigkeit des Algorithmus wird nicht berücksichtigt. Viele der höher eingestuften Algorithmen in der

Rankingdatenbank sind sehr rechenintensiv und im Moment nicht für Echtzeitanwendungen geeignet.

Abbildungen 3.30, 3.31 und 3.32 zeigen das Ergebnis einer realen Szene, aufgenommen mit der *BumbleBee*[9] Stereokamera.



(a) (b)

Abbildung 3.30.: Reale Szene: (a) linkes Bild $586x431$ (b) rechtes Bild $618x431$. (a) und (b) zeigen rektifizierte Aufnahmen, die ursprüngliche Bildauflösung betrug $640x480$

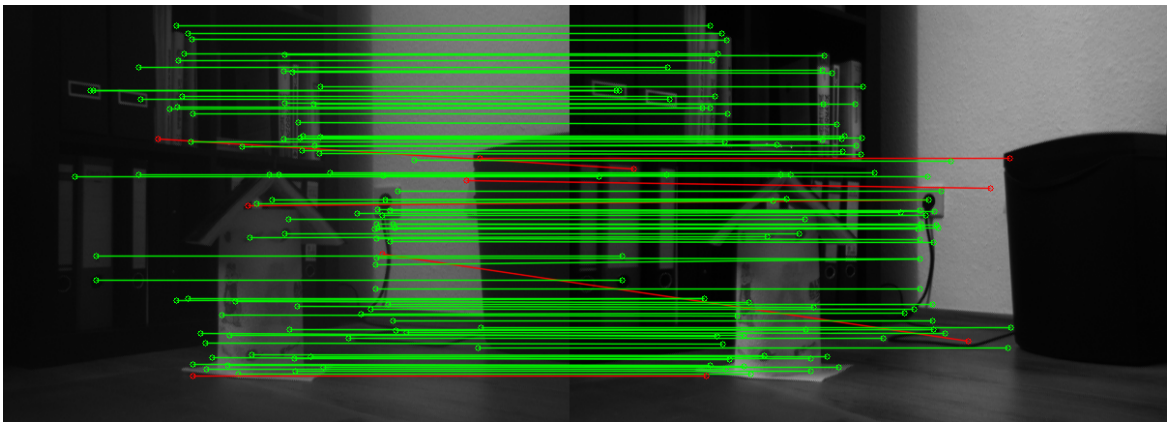
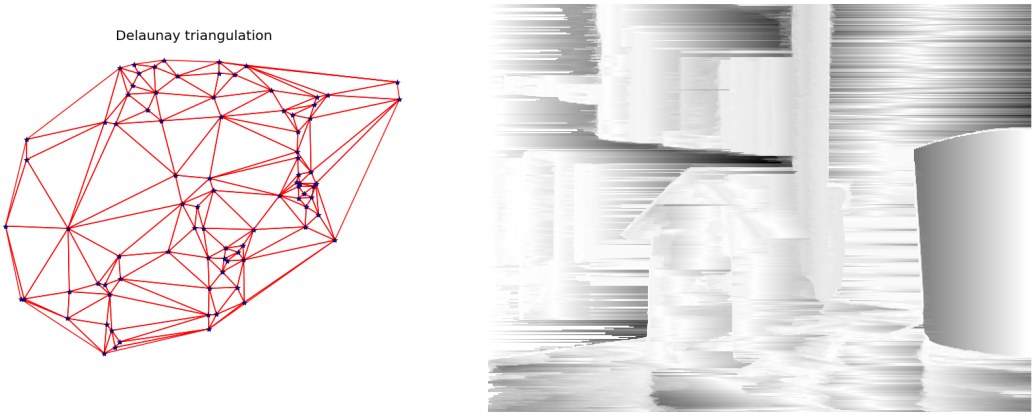


Abbildung 3.31.: Korrespondenzen für die Aufnahme 3.30. SURF-Merkmale: richtige Zuordnungen (grün) durch RANSAC gefilterte Falschzuordnungen (rot)

[9] siehe Abschnitt 4.3



(a) (b)

Abbildung 3.32.: Delaunay-Triangulation und Disparitätskarte für die Aufnahme 3.30. Aufgrund der schlechteren Beleuchtung und einer ungenaueren Kamerakalibrierung, im Vergleich zu den anderen Testbildern (siehe weiter oben), sind die horizontalen Streifen in dieser Disparitätskarte etwas stärker ausgeprägt. Diese streifenartigen Muster lassen sich mit Hilfe der im Abschnitt 3.6.5 beschriebenen Methode deutlich reduzieren.

3.7.1. Auswertung der Ergebnisse

Tabelle 3.2 fasst die Ergebnisse des Verfahrens für die in Abbildung 3.33 dargestellten Bilder und für die BumbleBee-Aufnahme 3.30 zusammen. Die Versuchsergebnisse werden auch als Balkendiagrammen dargestellt. Anhand der Diagramme wird diskutiert, wie die Ausführungszeiten der einzelnen Teilschritte des Verfahrens mit den Eigenschaften des vorliegenden Bildmaterials zusammenhängen.

(a) venus (b) tsukuba (c) cones (d) teddy

Abbildung 3.33.: Testbilder: Dargestellt ist jeweils das linke Bild eines Stereobildpaares.

	venus	**tsukuba**	**cones**	**teddy**	**BumbleBee**
Bildauflösung	434x383	384x288	450x375	450x375	640x480
Merkmalsextraktions-verfahren	SURF	SURF	SURF	SURF	SURF
Merkmale linkes Bild	1128	739	1438	957	240
Merkmale rechtes Bild	1142	784	1434	963	393
Korrespondenzen (gesamt)	429	327	388	305	91
Korrespondenzen (RANSAC gefiltert)	427	321	384	299	87
			Ausführungszeit		
Bildgewinnung	18ms	18ms	22ms	21ms	138ms
Rektifizierung	-	-	-	-	3ms
Merkmalsextraktion	1964ms	946ms	2787ms	1387ms	370ms
Delaunay Validierung	355ms	247ms	324ms	241ms	75ms
Transformation Dynamische Programmierung	391ms	325ms	372ms	385ms	1099ms

Tabelle 3.2.: Detaillierte Auswertung der Ergebnisse: Dieser Testlauf wurde auf einem Rechner mit einem Intel Core i7 Prozessor und 12GB RAM, ohne die in Kapitel 3.6 genannten Optimierungen, ausgeführt.

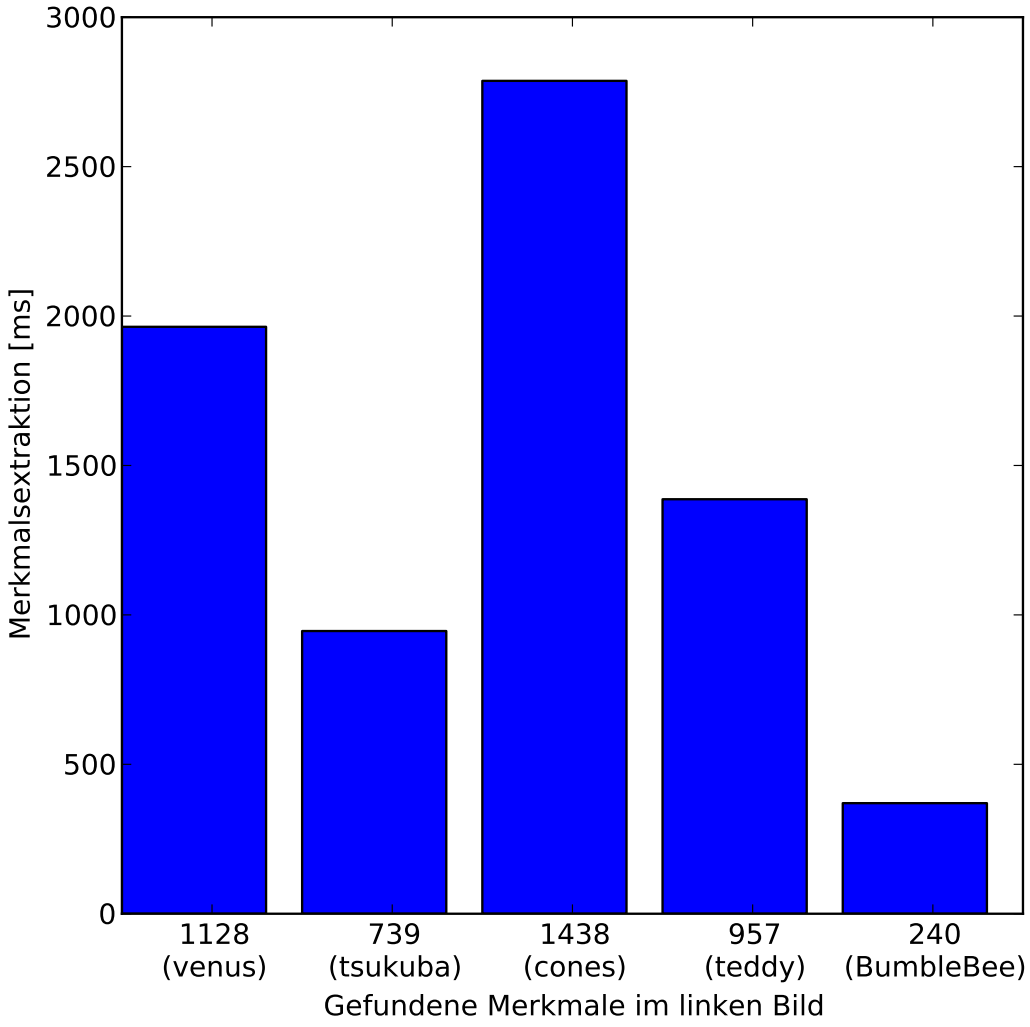


Abbildung 3.34.: Merkmalsextraktion im Verhältnis zur Anzahl gefundener Merkmale

Wie in der oberen Abbildung zu sehen ist, wächst die Ausführungszeit der Merkmalsextraktion mit der Anzahl gefundener Bildmerkmale. Denn je mehr Bildmerkmale gefunden werden, desto mehr Merkmalsdeskriptoren müssen berechnet werden und desto aufwändiger ist die Zuordnung korrespondierender Merkmale.

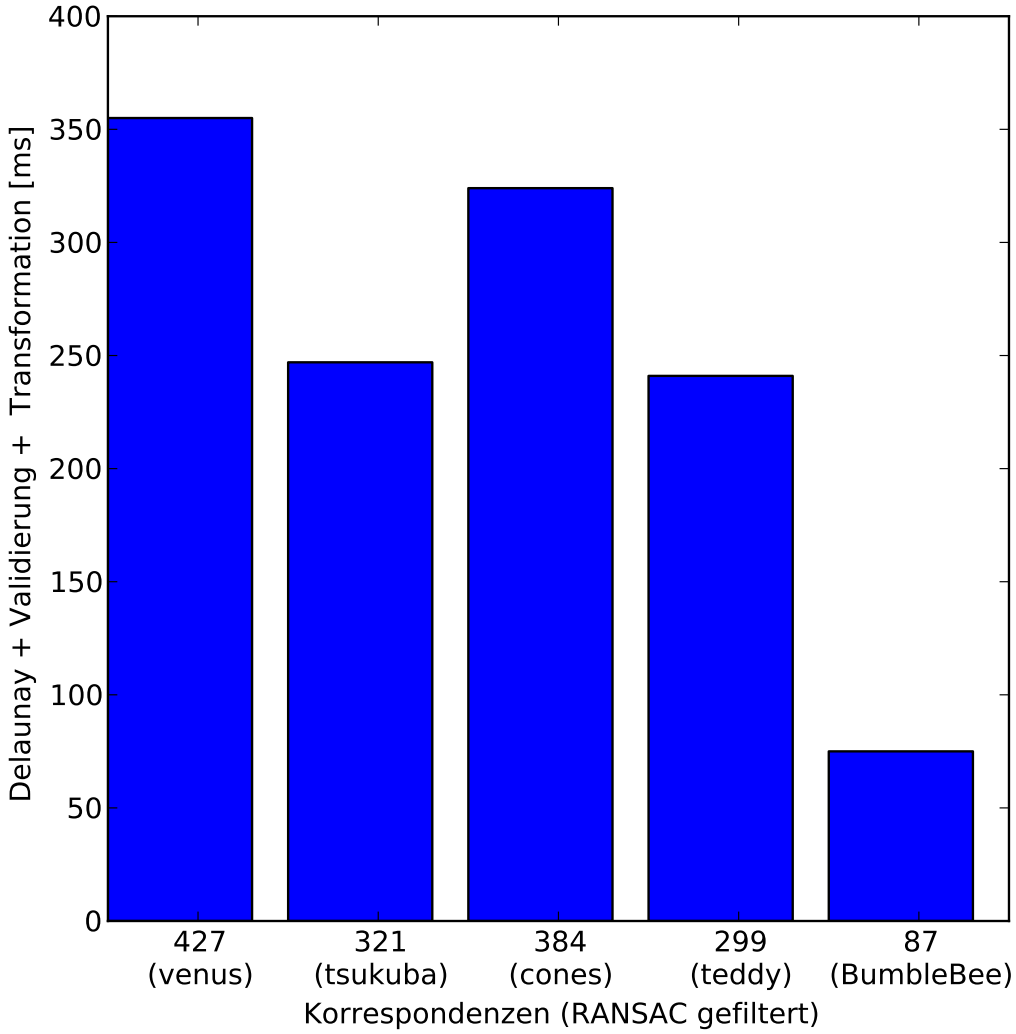


Abbildung 3.35.: Delaunay-Triangulation / Validierung der Dreieckssegmente / Projektive Transformation im Verhältnis zur Anzahl RANSAC-gefilterter Bildpunktkorrespondenzen

Eine höhere Anzahl robuster Bildpunktkorrespondenzen bedeutet, dass die Delaunay-Triangulation mehr Punkte zu einem Dreiecksnetz verbinden muss. Daraus ergibt sich auch eine höhere Anzahl der Dreieckssegmente, von denen jedes validiert und gegebenenfalls in das korrespondierende Bild transformiert werden muss. Aus diesem Grund steigt die Ausführungszeit der genannten Teilschritte des Verfahrens mit der Anzahl robuster Bildpunktkorrespondenzen.

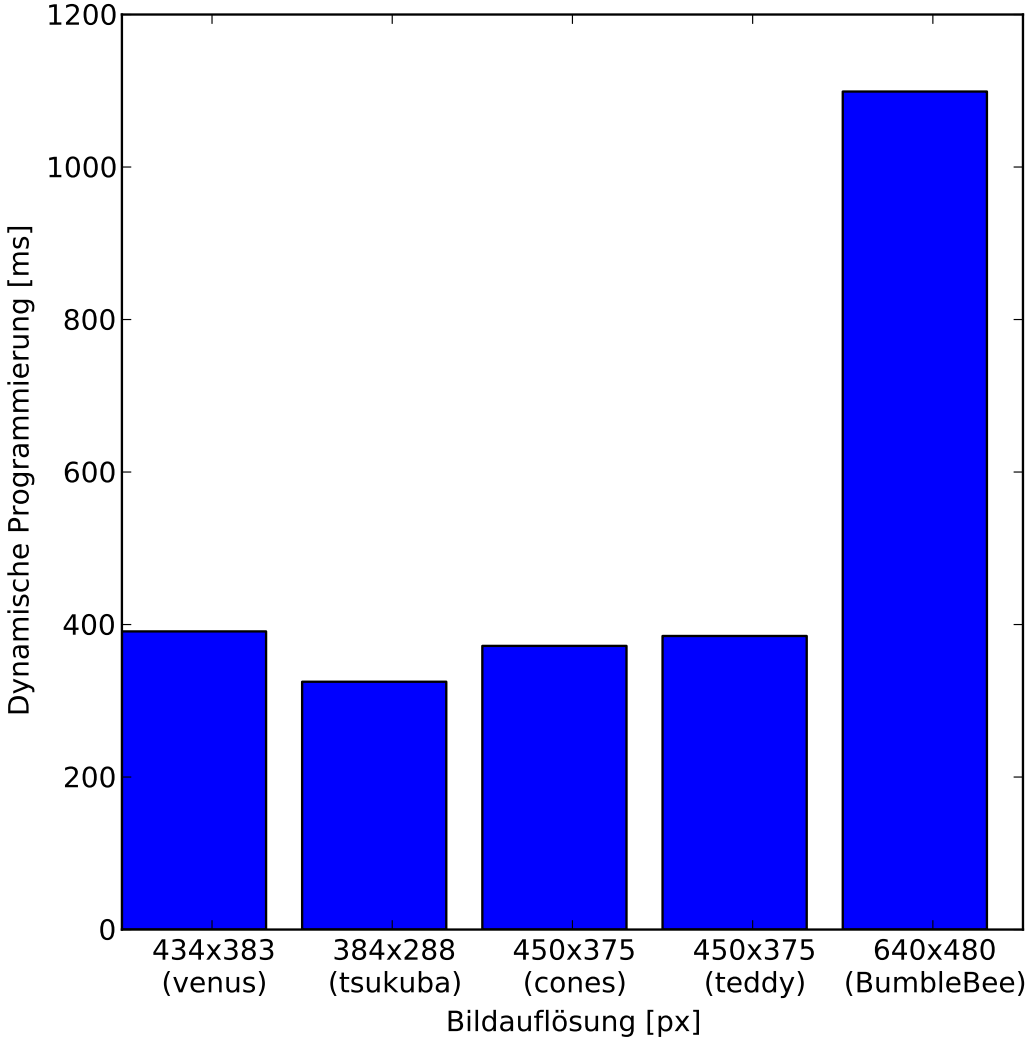


Abbildung 3.36.: Dynamische Programmierung im Verhältnis zur Bildauflösung

An diesem Diagramm erkennt man ganz deutlich, dass die Ausführungszeit der dynamischen Programmierung stark von der Bildgröße abhängig ist. Maßgebend ist hier die Höhe des Bildes, also die Anzahl der Zeilen, da für jedes Zeilenpaar eine Kostenmatrix berechnet werden muss.

4. Implementierung

Dieses Kapitel befasst sich mit der Implementierung des vorgestellten Verfahrens. Es werden kurz die verwendete Programmiersprache, die verwendete Bildverarbeitungsbibliothek und das eingesetzte Kamerasystem vorgestellt. Anschließend wird der Aufbau der Software beschrieben.

Die Software wurde unter dem Betriebssystem Linux entwickelt. Bei der Entwicklung und bei der Auswahl der einzelnen Softwarekomponenten wurde aber darauf geachtet, dass die Plattformunabhängigkeit der resultierenden Software sichergestellt ist.

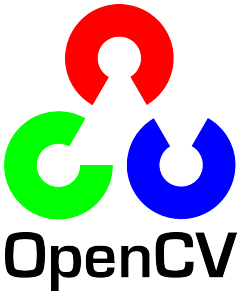
4.1. Wahl der Programmiersprache

Als Programmiersprache für die Implementierung des Verfahrens wurde Python[1] ausgewählt. Python ist eine universelle, portable, interpretative, objektorientierte, höhere Programmiersprache. Sie kann sowohl gut für Rapid Development als auch für große komplexe Projekte eingesetzt werden. Die Sprache ist kostenlos und ist für viele Betriebssysteme vorhanden. Zu ihren Eigenschaften gehören zum Beispiel Vererbung, Exceptions, parallele Programmierung und Schnittstellen zu allen gängigen GUI-Bibliotheken. Python verfügt über eine mächtige Standardbibliothek, die durch eigene Module leicht ergänzt werden kann. Die Module können auch in anderen Sprachen geschrieben werden, zum Beispiel in C. Die Syntax von Python ist sehr klar und ausdrucksstark, sie ermöglicht sehr kompakte Programmtexte, sie wurde auf das Wesentliche reduziert und auf Übersichtlichkeit und Lesbarkeit optimiert. Die Sprache umfasst einen hohen Funktionsumfang in nur wenigen Schlüsselwörtern. Neben der übersichtlichen Syntax, macht das die Sprache einfach erlernbar und anwendbar. Python umfasst mehrere Programmierparadigmen. So kann in Python beispielsweise rein prozedural oder objektorientiert programmiert werden.

[1] http://www.python.org

Durch eine Vielzahl von wissenschaftlichen Modulen und Bibliotheken ist Python in diesem Bereich sehr beliebt und wird von vielen Wissenschaftlern in der Forschung eingesetzt. NumPy[2] ist zum Beispiel eine sehr umfangreiche und leistungsfähige Bibliothek für die Arbeit mit mehrdimensionalen Arrays. Die Bibliothek SciPy[3] stellt unter anderem Module für Statistik, Optimierung, numerische Integration, lineare Algebra, Fourier-Transformation, Signal- und Bildverarbeitung zur Verfügung. SymPy[4] ist ein Modul für symbolische Kalkulationen, Matplotlib[5] und Mayavi2[6] sind Module für professionelle 2D- und 3D-Visualisierung wissenschaftlicher Daten und mit iPython[7] enthält Python eine interaktive Kommandozeile (Shell).

4.2. Verwendete Bildverarbeitungsbibliothek



Mit NumPy, SciPy und PIL[8] verfügt Python bereits über viele nützliche Bildverarbeitungsfunktionen. Um eine einheitliche Schnittstelle zu vielen aktuellen Verfahren für die Extraktion von Bildmerkmalen bereitzustellen, wurde die Bildverarbeitungsbibliothek OpenCV[9] verwendet.

OpenCV ist eine quelloffene Bibliothek für digitale Bildverarbeitung und maschinelles Sehen. Sie ist auf vielen Plattformen vorhanden und wird aktiv weiterentwickelt. Ihre Vorteile liegen in der Geschwindigkeit und vor allem in der großen Menge der implementierten Algorithmen. Die neusten Forschungsergebnisse im Bereich der Bildverarbeitung werden sehr schnell in die Bibliothek aufgenommen. Die Bibliothek umfasst unter anderem Algorithmen für Gesichtsdetektion, 3D-Funktionalität, Haar-Klassifikatoren, verschiedene Filter (Sobel, Canny, Gauß) und Funktionen für die Kamerakalibrierung. OpenCV ist in der Programmiersprache C geschrieben, enthält aber eine Python-Schnittstelle, die Python den kompletten Funktionsumfang von OpenCV zur Verfügung stellt. OpenCV enthält auch Optimierungen für Intel-Prozessoren, Grafikprozessoren und für die parallele Verarbeitung von Bilddaten.

[2] http://numpy.scipy.org
[3] http://www.scipy.org
[4] http://code.google.com/p/sympy
[5] http://matplotlib.sourceforge.net
[6] http://code.enthought.com/projects/mayavi
[7] http://ipython.org
[8] http://www.pythonware.com/products/pil
[9] http://code.opencv.org

4.3. Verwendetes Kamerasystem

Abbildung 4.1.: Bumblebee Stereo-Kamera

Bei der Entwicklung wurde das Stereokamerasystem *Bumblebee* der kanadischen Firma Point Grey Research[10] verwendet. Dieses Kamerasystem besteht intern aus zwei Kameras (2x Sony ICX084 Farb- oder Schwarzweiß-CCDs 640x480 Pixel), der Objektivabstand beträgt 12cm. Die Kamera wurde speziell für solche Aufgaben wie Personenverfolgung und Gestenerkennung für mobile Roboter entwickelt. Das Stereokamerasystem wurde vom Hersteller bereits vorkalibriert und enthält im Lieferumfang Software für die Erstellung von Tiefenbildern.

4.4. Aufbau der Software

Die Software besteht aus mehreren Modulen. Jedes Modul implementiert einen bestimmten Schritt des Verfahrens. Auf diese Weise können die einzelnen Verfahrensschritte durch neue, schnellere oder genauere Implementierungen ersetzt werden. Im Folgenden werden die einzelnen Module genauer beschrieben.

4.4.1. test.py

Dieses Modul beinhaltet sozusagen das Hauptprogramm. Hier werden andere Module initialisiert und ausgeführt.

4.4.2. features.py

Das ist das einzige Modul, das die OpenCV-Bibliothek benutzt. Es ist zuständig für die Kamerakalibrierung, für die Rektifizierung der Bilder und für das Auffinden und Filtern von robusten Bildmerkmalen.

Für die Kamerakalibrierung wird eine interaktive Menüführung bereitgestellt. Die Kalibrierung erfolgt automatisch mit Hilfe eines Schachbrettmusters. Die Kalibrierungsinformationen werden für jede Kamera separat gespeichert. Beim Programmstart wird eine angeschlossene

[10] http://www.ptgrey.com

Kamera anhand ihrer Seriennummer identifiziert und die zugehörigen Kalibrierungsinformationen werden automatisch geladen. Sollten keine Kalibrierungsinformationen vorliegen, wird der Benutzer zur Kalibrierung aufgefordert.

Das verwendete Merkmalsextraktionsverfahren kann einfach per Parameter bestimmt werden. Alle in OpenCV enthaltenen Verfahren werden unterstützt (in der aktuellen Version (2.4.1) sind es 10 unterschiedliche Verfahren).

4.4.3. stereo_match.py und homography.py

Das Modul *stereo_match.py* ist zuständig für die Delaunay-Triangulation robuster Punktkorrespondenzen, für die Validierung der Dreieckssegmente des Delaunay-Netzes und für die projektive Transformation der Segmente. Für die projektive Transformation wird das Modul *homography.py* verwendet.

4.4.4. cdp.pyx

Dieses Modul implementiert den Algorithmus der dynamischen Programmierung. Bei der dynamischen Programmierung wird für jedes Zeilenpaar einer Stereoaufnahme eine Kostenmatrix erstellt. Die Kosten dieser Kostenmatrix werden aggregiert und anschließend wird der beste Pfad durch die Matrix gesucht. Dadurch entstehen, insbesondere bei großen Bildern, sehr viele Schleifeniterationen. Schleifen können von kompilierbaren Programmiersprachen (wie zum Beispiel C oder C++) in der Regel sehr viel schneller ausgeführt werden. Aus diesem Grund wurde dieses Modul in der Programmiersprache Cython[11] implementiert. Cython ermöglicht auf eine einfache Art und Weise C-Module für Python zu entwickeln. Die Syntax von Cython ist weitgehend kompatibel zu Python. Ein Cython-Modul wird beim Laden in die Programmiersprache C übersetzt und anschließend automatisch kompiliert. Durch die Verwendung von Cython konnte die dynamische Programmierung um etwa das 500-fache beschleunigt werden.

4.4.5. bumblebee.py und filereader.py

Da es bei der Installation des proprietären Treibers für die Bumblebee-Stereokamera auf 64Bit-Linuxsystemen immer wieder Probleme aufgetreten sind, wurde im Rahmen dieser Arbeit ein leichtgewichtiger Treiber für die Bumblebee entwickelt. Der Treiber wurde in der Programmiersprache Python implementiert und ist im Modul *bumblebee.py* enthalten. Er

[11] http://www.cython.org

kann ohne großen Installationsaufwand auf verschiedenen Betriebssystemen benutzt werden.

Für einen einheitlichen Zugriff auf Bilddaten wurde eine einfache Programmierschnittstelle (API) definiert. Dadurch werden einzelne gespeicherte Bildpaare auf die gleiche Weise gelesen wie Live-Bildsequenzen einer Kamera.

Die API wird von *bumblebee.py* und von *filereader.py* implementiert. Das Modul *filereader.py* ist für das Lesen von Bildpaaren zuständig, die im lokalen Dateisystem gespeichert sind.

5. Zusammenfassung und Ausblick

Die vorliegende Arbeit stellt ein Verfahren zur robusten Tiefenbildgewinnung aus Stereobildern vor. Das Verfahren basiert auf dem Verfahren der dynamischen Programmierung in Kombination mit hochrobusten Punktkorrespondenzen (GCP), der Delaunay-Triangulation und der projektiven Transformation.

Die Implementierung des Verfahrens erfolgt in der Programmiersprache Python unter Verwendung der Bildverarbeitungsbibliothek OpenCV. Die Implementierung ist plattformunabhängig und hat einen modularen Aufbau. Bei der Entwicklung kommt das Stereokamerasystem Bumblebee zum Einsatz.

Die Arbeit ist in drei Hauptkapitel unterteilt. Nach einer kurzen Einführung in das Thema, werden im Kapitel 2 die Grundlagen der Bildaufnahme und der Stereoskopie vorgestellt. Im Kapitel 3 wird ausführlich, Schritt für Schritt das entwickelte Verfahren beschrieben. Am Ende des Kapitels werden die Ergebnisse des Verfahrens zusammengefasst. Das letzte Kapitel widmet sich schließlich der Implementierung des Verfahrens. Hier werden die verwendeten Softwarekomponenten und der Aufbau der Software kurz beschrieben. Im Anhang werden das Prinzip der dynamischen Programmierung und die Singulärwertzerlegung einer Matrix vorgestellt.

In den durchgeführten Tests zeigt das entwickelte Verfahren qualitativ gute Ergebnisse in einer guten Ausführungszeit. Im Abschnitt 3.6 werden einige Möglichkeiten vorgeschlagen, die zukünftig zur Verbesserung der Qualität und der Geschwindigkeit des Verfahrens beitragen können. Vor allem die Parallelisierung der dynamischen Programmierung und möglicherweise die Wahl eines schnelleren Merkmalsextraktionsverfahrens (wie zum Beispiel ORB [Rublee u. a. (2011)]), sind notwendig, um eine Ausführung in Echtzeit zu erreichen.

Literaturverzeichnis

[Bay u. a. 2006] BAY, Herbert ; TUYTELAARS, Tinne ; VAN GOOL, Luc: SURF: speeded up robust features. In: *Proceedings of the 9th European conference on Computer Vision - Volume Part I*. Berlin, Heidelberg : Springer-Verlag, 2006 (ECCV'06), S. 404–417. – URL http://dx.doi.org/10.1007/11744023_32. – ISBN 3-540-33832-2, 978-3-540-33832-1

[Beis und Lowe 1997] BEIS, Jeffrey S. ; LOWE, David G.: Shape Indexing Using Approximate Nearest-Neighbour Search in High-Dimensional Spaces. In: *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*. Washington, DC, USA : IEEE Computer Society, 1997 (CVPR '97), S. 1000–. – URL http://dl.acm.org/citation.cfm?id=794189.794431. – ISBN 0-8186-7822-4

[Bentley 1975] BENTLEY, Jon L.: Multidimensional binary search trees used for associative searching. In: *Commun. ACM* 18 (1975), September, S. 509–517. – URL http://doi.acm.org/10.1145/361002.361007. – ISSN 0001-0782

[Birchfield und Tomasi 1999] BIRCHFIELD, Stan ; TOMASI, Carlo: Depth Discontinuities by Pixel-to-Pixel Stereo. In: *Int. J. Comput. Vision* 35 (1999), December, S. 269–293. – URL http://dl.acm.org/citation.cfm?id=340180.340201. – ISSN 0920-5691

[Bobick und Intille 1999] BOBICK, Aaron F. ; INTILLE, Stephen S.: Large Occlusion Stereo. In: *Int. J. Comput. Vision* 33 (1999), September, S. 181–200. – URL http://dl.acm.org/citation.cfm?id=335178.335180. – ISSN 0920-5691

[Bradski und Kaehler 2008] BRADSKI, Dr. Gary R. ; KAEHLER, Adrian: *Learning opencv, 1st edition*. First. O'Reilly Media, Inc., 2008. – ISBN 9780596516130

[Chen u. a. 2008] CHEN, Chao-I ; SARGENT, Dusty ; TSAI, Chang-Ming ; WANG, Yuan-Fang ; KOPPEL, Dan: Stabilizing Stereo Correspondence Computation Using Delaunay Triangulation and Planar Homography. In: *Proceedings of the 4th International Symposium on Advances in Visual Computing*. Berlin, Heidelberg : Springer-Verlag, 2008 (ISVC '08), S. 836–845. – URL http://dx.doi.org/10.1007/978-3-540-89639-5_80. – ISBN 978-3-540-89638-8

[Fischler und Bolles 1981] FISCHLER, Martin A. ; BOLLES, Robert C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. In: *Communications of the ACM* 24 (1981), Nr. 6, S. 381–395. – URL http://portal.acm.org/citation.cfm?id=358669.358692

[Forstmann u. a. 2004] FORSTMANN, Sven ; KANOU, Yutaka ; OHYA, Jun ; THUERING, Sven ; SCHMITT, Alfred: Real-Time Stereo by using Dynamic Programming. In: *Proceedings of the 2004 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'04) Volume 3 - Volume 03*. Washington, DC, USA : IEEE Computer Society, 2004, S. 29–. – URL http://dl.acm.org/citation.cfm?id=1032634.1032920. – ISBN 0-7695-2158-4

[Geiger u. a. 2011] GEIGER, Andreas ; ROSER, Martin ; URTASUN, Raquel: Efficient large-scale stereo matching. In: *Proceedings of the 10th Asian conference on Computer vision - Volume Part I*. Berlin, Heidelberg : Springer-Verlag, 2011 (ACCV'10), S. 25–38. – URL http://dl.acm.org/citation.cfm?id=1964320.1964325. – ISBN 978-3-642-19314-9

[Gong und Yang 2003] GONG, Minglun ; YANG, Yee-Hong: Fast Stereo Matching Using Reliability-Based Dynamic Programming and Consistency Constraints. In: *Proceedings of the Ninth IEEE International Conference on Computer Vision - Volume 2*. Washington, DC, USA : IEEE Computer Society, 2003 (ICCV '03), S. 610–. – URL http://dl.acm.org/citation.cfm?id=946247.946596. – ISBN 0-7695-1950-4

[González u. a. 2007] GONZÁLEZ, J. I. ; GÁMEZ, J. C. ; SOSA, J. D. H. ; BRITO, A. C. D.: Comparing self-calibration methods for static cameras. In: *Proceedings of the 11th international conference on Computer aided systems theory*. Berlin, Heidelberg : Springer-Verlag, 2007 (EUROCAST'07), S. 660–667. – URL http://dl.acm.org/citation.cfm?id=1783034.1783125. – ISBN 3-540-75866-6, 978-3-540-75866-2

[Gonzalez u. a. 1999] GONZALEZ, R. C. ; CANCELAS, J. A. ; ALVAREZ, J. C. ; FERNANDEZ, J. A. ; ALVAREZ, I.: Dynamic programming stereo vision algorithm for robotic applications. In: *Vision Interface* (1999), S. 117–124

[Hartley und Zisserman 2003] HARTLEY, Richard ; ZISSERMAN, Andrew: *Multiple View Geometry in Computer Vision*. 2. New York, NY, USA : Cambridge University Press, 2003. – ISBN 0521540518

[Hermann und Klette] HERMANN, Simon ; KLETTE, Reinhard: The Naked Truth about Cost Functions for Stereo Matching / The University of Auckland. – Forschungsbericht

[Hirschmüller und Scharstein 2007] HIRSCHMÜLLER, H. ; SCHARSTEIN, D.: Evaluation of cost functions for stereo matching. In: *In IEEE Computer Society Conference on*

Computer Vision and Pattern Recognition (CVPR 2007) (2007), Juni. – URL http://bj.middlebury.edu/~schar/papers/evalCosts_cvpr07.pdf

[Holzhäuser 2008] HOLZHÄUSER, Dennis: *Medizinische Bildverarbeitung: Eine Zusammenfassung der wichtigsten Vorlesungsthemen*. 2008

[Hough 1962] HOUGH, Paul: *Method and Means for Recognizing Complex Patterns*. U.S. Patent 3.069.654. Dezember 1962

[Jähne 2002] JÄHNE, B.: *Digitale Bildverarbeitung*. 5. Springer, 2002. – URL http://books.google.de/books?id=jCk2AAAACAAJ. – ISBN 9783540412601

[Juan und Gwon 2009] JUAN, Luo ; GWON, Oubong: A Comparison of SIFT, PCA-SIFT and SURF. In: *International Journal of Image Processing (IJIP)* 3 (2009), Nr. 4, S. 143–152. – URL http://www.cscjournals.org/csc/manuscript/Journals/IJIP/volume3/Issue4/IJIP-51.pdf

[Kalarot und Morris 2010] KALAROT, R. ; MORRIS, J.: Comparison of FPGA and GPU implementations of real-time stereo vision. In: *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, URL http://dx.doi.org/10.1109/CVPRW.2010.5543743, Juni 2010, S. 9–15

[Kim u. a. 2005] KIM, Jae C. ; LEE, Kyoung M. ; CHOI, Byoung T. ; LEE, Sang U.: A Dense Stereo Matching Using Two-Pass Dynamic Programming with Generalized Ground Control Points. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*. Washington, DC, USA : IEEE Computer Society, 2005 (CVPR '05), S. 1075–1082. – URL http://dx.doi.org/10.1109/CVPR.2005.22. – ISBN 0-7695-2372-2

[Lowe 2004] LOWE, David G.: Distinctive Image Features from Scale-Invariant Keypoints. In: *Int. J. Comput. Vision* 60 (2004), November, S. 91–110. – URL http://dl.acm.org/citation.cfm?id=993451.996342. – ISSN 0920-5691

[Mattoccia 2011] MATTOCCIA, Stefano: *Stereo Vision: Algorithms and Applications*. 2011

[Mikolajczyk und Schmid 2005] MIKOLAJCZYK, Krystian ; SCHMID, Cordelia: A Performance Evaluation of Local Descriptors. In: *IEEE Trans. Pattern Anal. Mach. Intell.* 27 (2005), October, S. 1615–1630. – URL http://dl.acm.org/citation.cfm?id=1083822.1083989. – ISSN 0162-8828

[Passig 2006] PASSIG, Georg: *Umgebungsmodellierung auf der Basis von Stereo-Kamerabildern für eine Telepräsenzanwendung*, Lehrstuhl für Realzeit–Computersysteme, Technische Universität München, Dissertation, 2006. – URL http://mediatum.ub.tum.de/mediatum/servlets/MCRQueryServlet?mode=ObjectMetadata&status=2&type=alldocs&hosts=local&lang=

de&query=%2Fmycoreobject[%40ID%3D'mediaTUM_disshab_000000000004518']

[Rublee u. a. 2011] RUBLEE, Ethan ; RABAUD, Vincent ; KONOLIGE, Kurt ; BRADSKI, Gary: ORB: An Efficient Alternative to SIFT or SURF. In: *International Conference on Computer Vision*. Barcelona, 11/2011 2011

[Sadeghi u. a. 2008] SADEGHI, Hajar ; MOALLEM, Payman ; MONADJEMI, S. A.: Feature Based Dense Stereo Matching using Dynamic Programming and Color. In: *International Journal of Information and Mathematical Sciences* (2008)

[Salmen u. a. 2009] SALMEN, Jan ; SCHLIPSING, Marc ; EDELBRUNNER, Johann ; HEGEMANN, Stefan ; LÜKE, Stefan: Real-Time Stereo Vision: Making More Out of Dynamic Programming. In: *Proceedings of the 13th International Conference on Computer Analysis of Images and Patterns*. Berlin, Heidelberg : Springer-Verlag, 2009 (CAIP '09), S. 1096–1103. – URL http://dx.doi.org/10.1007/978-3-642-03767-2_133. – ISBN 978-3-642-03766-5

[Scharstein und Pal 2007] SCHARSTEIN, D. ; PAL, C.: Learning conditional random fields for stereo. In: *In IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2007)* (2007), Juni. – URL http://bj.middlebury.edu/~schar/papers/LearnCRFstereo_cvpr07.pdf

[Scharstein und Szeliski] SCHARSTEIN, D. ; SZELISKI, R.: *http://vision.middlebury.edu/stereo/eval/*

[Scharstein und Szeliski 2002] SCHARSTEIN, Daniel ; SZELISKI, Richard: A Taxonomy and Evaluation of Dense Two-Frame Stereo Correspondence Algorithms. In: *Int. J. Comput. Vision* 47 (2002), April, S. 7–42. – URL http://dl.acm.org/citation.cfm?id=598429.598475. – ISSN 0920-5691

[Scharstein und Szeliski 2003] SCHARSTEIN, Daniel ; SZELISKI, Richard: High-accuracy stereo depth maps using structured light. In: *Proceedings of the 2003 IEEE computer society conference on Computer vision and pattern recognition*. Washington, DC, USA : IEEE Computer Society, 2003 (CVPR'03), S. 195–202. – URL http://dl.acm.org/citation.cfm?id=1965841.1965865. – ISBN 0-7695-1900-8, 978-0-7695-1900-5

[Su und Drysdale 1995] SU, Peter ; DRYSDALE, Robert L. S.: A comparison of sequential Delaunay triangulation algorithms. In: *Proceedings of the eleventh annual symposium on Computational geometry*. New York, NY, USA : ACM, 1995 (SCG '95), S. 61–70. – URL http://doi.acm.org/10.1145/220279.220286. – ISBN 0-89791-724-3

[Sun 2002] SUN, Changming: Fast Stereo Matching Using Rectangular Subregioning and 3D Maximum-Surface Techniques. In: *Int. J. Comput. Vision* 47 (2002), April, S. 99–117. – URL http://dl.acm.org/citation.cfm?id=598429.598481. – ISSN 0920-5691

[Tsai 1992] TSAI, Roger Y.: Radiometry. USA : Jones and Bartlett Publishers, Inc., 1992, Kap. A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off-the-shelf TV cameras and lenses, S. 221–244. – URL http://dl.acm.org/citation.cfm?id=136913.136938. – ISBN 0-86720-294-7

[Veksler 2005] VEKSLER, Olga: Stereo Correspondence by Dynamic Programming on a Tree. In: *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 2 - Volume 02*. Washington, DC, USA : IEEE Computer Society, 2005 (CVPR '05), S. 384–390. – URL http://dx.doi.org/10.1109/CVPR.2005.334. – ISBN 0-7695-2372-2

[Viola und Jones # 2002] VIOLA, Paul ; JONES, Michael: Robust Real-time Object Detection. In: *International Journal of Computer Vision - to appear* (# 2002). – URL http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.23.2751

[Wang u. a. 2006] WANG, Liang ; LIAO, Miao ; GONG, Minglun ; YANG, Ruigang ; NISTER, David: High-Quality Real-Time Stereo Using Adaptive Cost Aggregation and Dynamic Programming. In: *Proceedings of the Third International Symposium on 3D Data Processing, Visualization, and Transmission (3DPVT'06)*. Washington, DC, USA : IEEE Computer Society, 2006 (3DPVT '06), S. 798–805. – URL http://dx.doi.org/10.1109/3DPVT.2006.75. – ISBN 0-7695-2825-2

Anhang

A. Dynamische Programmierung

Die dynamische Programmierung (DP) [Richard Bellman, 1953] ist ein Verfahren, mit dem eine Vielzahl von Optimierungsproblemen gelöst werden kann.

Das Prinzip des Verfahrens ist die Aufteilung eines zu lösenden Problems in geeignete Teilprobleme. Die Gesamtlösung setzt sich dann aus den Lösungen der Teilprobleme zusammen.

Der große Unterschied zu rekursiven Verfahren ist, dass bei der dynamischen Programmierung die Zwischenergebnisse, also die Lösungen der Teilprobleme, in einer Tabelle gespeichert werden. Bei rekursiven Ansätzen werden viele Teilprobleme mehrmals berechnet. Bei der dynamische Programmierung, kann dagegen auf ein bereits berechnetes und gespeichertes Ergebnis immer wieder zugegriffen werden. Das führt zu einer enormen Verkürzung der Rechenzeit.

Abbildung A.1 zeigt einen Graph mit gewichteten Kanten. Die Aufgabe ist es, ausgehend von dem Startknoten oben links (rot umrandet), den Pfad zu dem Endknoten unten links (blau umrandet), auf dem die Summe der Kantengewichte am geringsten ist, zu ermitteln.

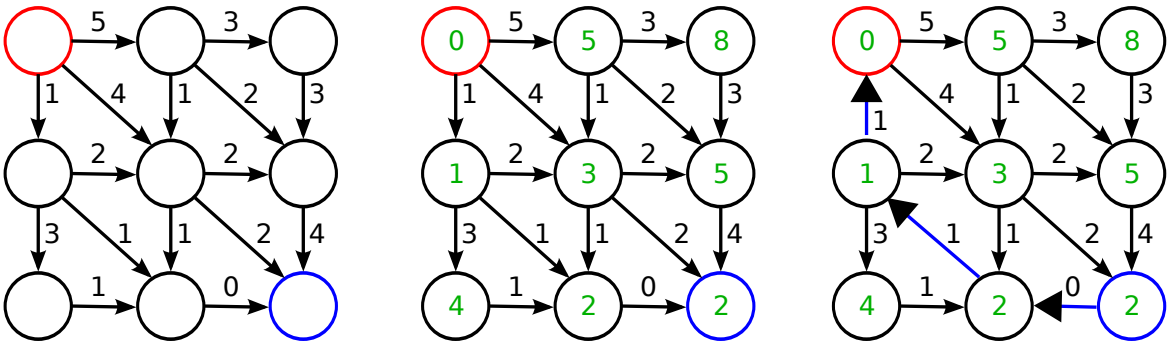


Abbildung A.1.: Dynamische Programmierung: Suche nach dem Pfad mit den geringsten Gewichten in einem Graph mit gewichteten Kanten.

Die Pfadsuche mit Hilfe der dynamischen Programmierung erfolgt nun in zwei Schritten:

1. Im ersten Schritt werden Kantengewichte des gesamten Graphen aufsummiert. Dazu geht man vom Startknoten aus. Bei jedem Nachfolgeknoten wird die Summe aus dem Knotenwert und dem dazugehörigen Kantengewicht des besten Vorgängers gespeichert. Der beste Vorgänger ist der mit den, in der Summe, kleinsten Gewichten.
2. Der zweite Schritt, bei dem schließlich der beste Pfad mit Hilfe der aufsummierten Gewichte ermittelt wird, wird als *Backtracking* bezeichnet. Dazu fängt man rechts unten an und folgt dem Pfad entlang der geringsten Knotenwerte.

Ein weiteres sehr anschauliches Beispiel für den Vergleich einer rekursiven Lösung und einer Lösung mit Hilfe der dynamischen Programmierung, ist die Berechnung von Fibonacci-Zahlen. Das Bildungsgesetz der Fibonacci-Folge ist wie folgt rekursiv definiert:

$$fib(n) = \begin{cases} 0 & \text{für } n = 0 \\ 1 & \text{für } n = 1 \\ fib(n) = fib(n-1) + fib(n-2) & \text{für } n \geq 2 \end{cases} \tag{A.1}$$

Das Listing A.1 zeigt die rekursive Implementierung der Fibonacci-Funktion in der Programmiersprache Python. Der Quellcode ist selbsterklärend.

Listing A.1: Fibonacci (rekursiv)

```
def fib(n):

    if n == 0:
        return 0

    if n == 1:
        return 1

    return fib(n - 1) + fib(n - 2)
```

Abbildung A.2 zeigt einen Baum mit Funktionsaufrufen (Teillösungen), den die rekursive Implementierung für $n = 5$ erzeugt.

Man sieht, dass die Teillösungen im grau unterlegten Bereich, mehrmals berechnet werden. Man erkennt auch, dass für die Gesamtlösung nur die Teillösungen `fib(4)` und `fib(3)`, also nur `fib(n - 1)` und `fib(n - 2)`, benötigt werden.

Um einen DP-Algorithmus herzuleiten, geht man folgendermaßen vor:

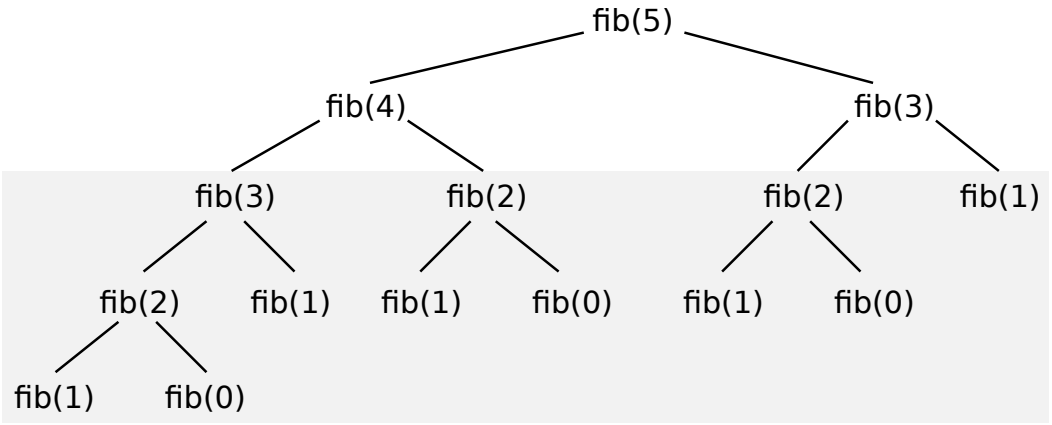


Abbildung A.2.: Funktionsaufrufgraph für die rekursive Implementierung der Fibonacci-Funktion (A.1) mit $n = 5$.

1. Aufteilung des Gesamtproblems in geeignete, kleinere Teilprobleme.
2. Herleitung einer rekursiven Beschreibung des Problems.
3. Iterative Berechnung und Speicherung von Teillösungen, solange bis das Gesamtproblem gelöst ist.

Die drei genannten Schritte lassen sich auch auf das Fibonacci-Problem übertragen:

1. $fib(0), \ldots, fib(n-1)$
2. Die rekursive Problembeschreibung ist durch die rekursive Definition der Fibonacci-Reihe gegeben.
3. Iterative Berechnung aller Teillösungen mit Speicherung der Zwischenergebnisse (siehe Listing A.2).

Listing A.2: Fibonacci (iterativ)

```
def fib_dp(n):

    n1 = 1
    n2 = 0

    for i in range(n - 2):
        n2 = n1
        n1 = n1 + n2

    return n1 + n2
```

Mit Hilfe der dynamischen Programmierung lassen sich viele bekannte Probleme der Informatik, wie zum Beispiel das *Rucksackproblem* oder das *Handelsreisendenproblem*, lösen. Die Anwendung der dynamischen Programmierung zur Lösung des Stereokorrespondenzproblems wird im Abschnitt 3.5 beschrieben.

B. Singulärwertzerlegung (SVD)

Jede beliebige Matrix $A \in \mathbb{R}^{m \times n}$ mit dem Rang p kann als Produkt der drei Matrizen $U \in \mathbb{R}^{m \times m}$, $D \in \mathbb{R}^{m \times n}$ und $V \in \mathbb{R}^{n \times n}$ dargestellt werden:

$$A = UDV^T = \begin{pmatrix} u_1 & u_2 & \cdots & u_m \end{pmatrix} \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \sigma_p \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \tag{B.2}$$

Die Elemente σ_i mit $i = 1, \ldots, p$ und $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_p \geq 0$ nennt man die *Singulärwerte* von A. u_i und v_i sind die linken und rechten *Singulärvektoren*.

Der Rang einer Matrix wird durch die Anzahl der linear unabhängigen Vektoren in dieser Matrix bestimmt. Diese Anzahl entspricht der Anzahl der Singulärwerte.

Mit Hilfe der SVD lässt sich der Rang einer Matrix erzwingen, indem, beginnend beim kleinsten, die gewünschte Anzahl der Singulärwerte in D auf 0 gesetzt wird.

Die Singulärwertzerlegung kann herangezogen werden, um die Lösung mit einem minimalen quadratischen Fehler für überbestimmte lineare Gleichungssysteme der Art $A \cdot x = a$ zu bestimmen. Für die Bestimmung des Lösungsvektors $x = A^+ \cdot a$ wird dabei die Pseudoinverse A^+ der Matrix A benötigt. A^+ lässt sich berechnen, indem alle Singulärwerte in D durch ihren Kehrwert ersetzt werden.

$$A^+ = UD^{-1}V^T = \begin{pmatrix} u_1 & u_2 & \cdots & u_m \end{pmatrix} \begin{pmatrix} \frac{1}{\sigma_1} & 0 & \cdots & 0 \\ 0 & \frac{1}{\sigma_2} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & \frac{1}{\sigma_p} \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \tag{B.3}$$

Glossar

3D-Objekt siehe Objekt

3D-Punkt siehe Raumpunkt

Bildpunkt Pixel eines digitalen Bildes. Ein Bildpunkt liegt im zweidimensionalen Raum und besitzt die Koordinaten x und y.

Merkmalspunkt SIFT/SURF-Merkmale sind subpixelgenau. Das heißt, ein Merkmalspunkt kann sich auch zwischen zwei Pixel befinden. In diesem Fall wird das Merkmal pixelgenau interpoliert.

Objekt Gegenstand im dreidimensionalen Raum.

Objektpunkt Ein Raumpunkt, der sich auf einem Objekt befindet.

Punkt In der Regel ist damit ein Bildpunkt gemeint.

Raumpunkt Ein Punkt im dreidimensionalen Raum mit den drei Koordinaten x, y und z.

Region Ein Ausschnitt eines digitalen Bildes, welcher mehrere Pixel beinhaltet.

Weltpunkt siehe Raumpunkt

Versicherung der Selbstständigkeit

Hiermit versichere ich, dass ich die vorliegende Arbeit im Sinne der Prüfungsordnung nach §24(5) ohne fremde Hilfe selbstständig verfasst und nur die angegebenen Hilfsmittel benutzt habe.

Hamburg, 29. Juni 2012	
Ort, Datum	Unterschrift