Seminar *Rekonstruktion aus Bildern*

Dozent: *Detlev Droege*

WS 2003/04

Institut für Computervisualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
agas@uni-koblenz.de
http://www.uni-koblenz.de/∼agas

Inhaltsverzeichnis

Vortrag 1

Kameraaufbau & Geometrie bei PTV-Geräten

Christian Bauer

November 13th 2002

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
chrbauer@uni-koblenz.de
http://www.uni-koblenz.de/∼chrbauer

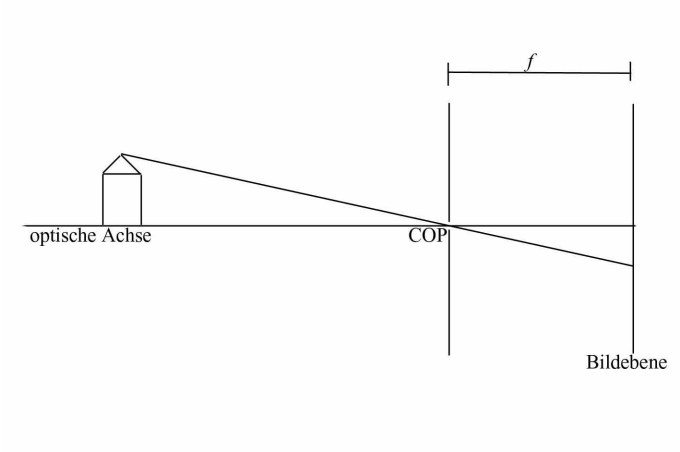


Abbildung 1.1: Skizze der Ideal Pinhole. Die Lichtstrahlen verlaufen vom Objekt zum Loch und kreuzen sich dort mit der Optischen Achse im COP. Der Abstand zwischen Bildebene links und und dem COP ist die Brennweite f.

1.1 Motivation

Für die Rekonstruktion aus Bildern werden verschiedene Vorkenntnisse benötigt. Diese sollen in den folgenden Seiten näher gebracht werden. Als erstes werden hierzu kurz verschiedene Kameramodelle vorgestellt. Außerdem wird genauer auf die unterschiedlichen Koordinatensysteme einer Szene eingegangen. Als drittes wird noch die Überführung der einzelnen Koordinatensysteme in ein anderes erklärt.

Somit dient dieses Kapitel als Einleitung und zum Einstieg in die späteren Kapitel, wo verschiedene Themen genauer betrachtet werden.

1.2 Kameramodelle

Ein Kameramodell beschreibt die Art und Weise, wie die Welt durch die Kamera auf dem Bild dargestellt wird; wie der Name schon sagt als Modell. Dies entspricht nicht unbedingt den realen physikalischen Vorgängen innerhalb einer Kamera. Es dient dazu, diese Vorgänge mehr oder weniger einfach zu berechnen.

Die drei hier vorgestellten Kameramodelle sind unterschiedlich in ihrer Realitätsnähe und dementsprechend verschieden komplex in ihrer Berechnung.

1.2.1 Ideal Pinhole

Das Modell des Ideal Pinhole, oder zu deutsch Lochkamera, ist das einfachste der drei Modelle und auch dementsprechend einfach in der Anwendung. Hier befindet sich das *COP* [1] exakt am Loch der Kamera. Die Lichtstrahlen verlaufen vom Objekt aus auf das COP zu, wo sie sich kreuzen. Die *Optische Achse* beschreibt eine Linie durch das Loch und liegt orthogonal auf der Bildebene. Die Distanz zwischen dem COP und der Bildebene ist die Brennweite f.

Das *COR* [2] sollte nach Möglichkeiten mit dem COP übereinstimmen. Dies ist aber, wie man in den späteren Beispielen sieht, nicht immer durchführbar. Zur Veranschaulichung des Aufbaus der Ideal Pinhole dient das Bild 1.1.

Auch wenn das Ideal Pinhole das einfachste Modell ist, genügt es meist den Anforderungen der Rekonstruktion aus Bildern und soll auch hier dafür verwendet werden. Das Bild entsteht hierbei durch *Perspektivische Projektion*, was später noch genauer erklärt wird.

[1] Center of Projection
[2] Center of Rotation: Der Punkt, um den die Drehung der Kamera stattfi nden.

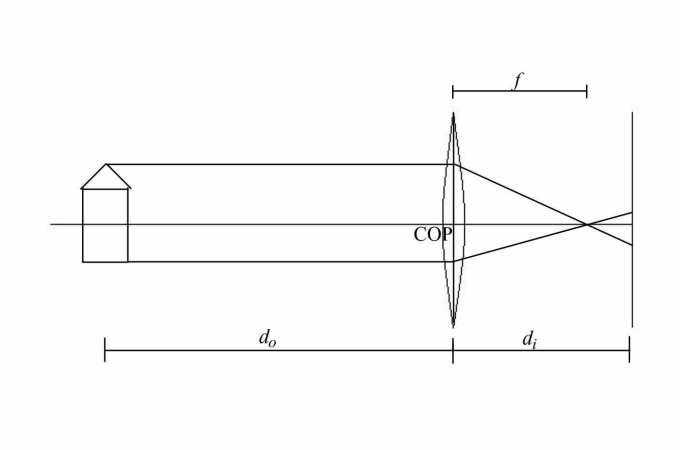


Abbildung 1.2: Skizze der Thin Lense. Die Lichtstrahlen verlaufen parallel der optischen Achse bis zur Linse und werden dort gebrochen. Die Distanz vom Linsenzentrum (COP) bis zum Schnittpunkt der Lichtstrahlen ist die Brennweite f. Die Distanz vom Objekt bis zum COP ist d_o und vom COP bis zur Bildebene links ist d_i.

1.2.2 Thin Lense

Beim Modell der Thin Lense wird angenommen, dass eine Linse vorhanden ist, in der sich die Lichtstrahlen brechen. Das COP ist in diesem Fall genau im Linsenzentrum. Hier durch verläuft auch die Optische Achse die wie bei der Ideal Pinhole ebenfalls orthogonal auf der Bildebene liegt. Die eintreffenden Lichtstrahlen verlaufen parallel zur Optischen Achse und werden in der Linse gebündelt, wobei der Strahl, der durch das COP verläuft, nicht gebrochen wird. Die Distanz vom COP bis zu dem Punkt, an dem sich die Lichtstrahlen kreuzen, ist die Brennweite f. Zur Veranschaulichung des Aufbaus der Thin Lens dient das Bild 1.2.

Ein scharfes Bild eines Objektes in unendlicher Entfernung erhält man genau auf der Brennweite f. Ist die Entfernung vom Objekt zum COP d_o geringer als unendlich, so muss die Entfernung zwischen COP und Bildebene d_i verstellt werden. Die Änderung von d_i wird mit der Formel

$$\frac{1}{f} = \frac{1}{d_i} + \frac{1}{d_o}$$

berechnet. Umso näher das Objekt der Kamera ist, desto größer wird d_i.

Das COR sollte beim Thin Lense Modell mit dem COP übereinstimmen, was aber genauso schwer zu realisieren ist, wie beim Ideal Pinhole.

1.2.3 Thick Lense

Das Modell der Thick Lense kommt der Realität einer echten Kamera mit mehreren Linsen am nächsten. Es soll hier aber nur kurz der Vollständigkeit halber erwähnt werden.

Bei der Thick Lense werden die Lichtstrahlen an mehreren *Optischen Ebenen* gebrochen. Mit 2 Ebenen ist eine ausreichend realistische Simulation einer echten Linse möglich. Genauere Informationen sind in [3] zu finden.

1.3 Koordinatensysteme und deren Überführung

Im Folgenden werden die vier verschiedenen Koordinatensysteme erläutert, mit denen eine Szene dargestellt wird. Sie repräsentieren die verschiedenen Stufen der Berechnung von einer Szenerie mit Objekt und Kamera bis hin zum Bild mit diskreten Pixelwerten.

Alle Koordinatensysteme sind *rechtshändige Koordinatensysteme*. Außerdem werden *homogene Koordinaten* [3] verwendet. Für weitere Details zu homogenen Koordinaten siehe [1]. Nachdem die erforderlichen Koordinatensysteme vorge-

[3] Projektiver Raum

stellt wurden, wird die Transformation von der einen zur anderen erklärt. Diese Transformation erfolgt in der Regel durch die Multiplikation der homogenen Koordinaten mit einer Matrix.

1.3.1 Welt-Koordinatensystem *WCS*

Im Welt-Koordinatensystem wird die gesamte Szenerie beschrieben. Die Lage sämtlicher Objekte in der Welt und auch der Kamera. In diesem Koordinatensystem wird auch die Bewegung einer Kamera beschrieben.

Der Ursprung und die Orientierung des WCS dürfen willkürlich gewählt werden. In der Regel werden sie möglichst praktisch gewählt, so dass eine Umrechnung vom Welt-Koordinatensystem ins nächste Koordinatensystem so einfach wie möglich ist.

Die Schreibweise für einen Punkt im WCS ist wie folgt: p^{W}.

1.3.2 Kamera-Koordinatensystem *CCS*

Das Kamera-Koordinatensystem beschreibt die Szene aus der Sicht der Kamera, ist also relativ zu deren Position.

Der Ursprung liegt im COP der Kamera. Die negative z-Achse ist identisch mit der Optischen Achse und entspricht damit der Blickrichtung der Kamera. Und die x- und y-Achsen liegen parallel zu den Bildreihen und Bildspalten.

Die Schreibweise für einen Punkt im CCS ist wie folgt: p^{C}.

1.3.3 Überführung WCS → CCS

Um jetzt einen Punkt p^{W} des WCS in einen Punkt p^{C} des CCS zu überführen, sind zwei Schritte erforderlich. Zum einen wird mittels einer Translation t_a der Ursprung des CCS in den Ursprung des WCS verschoben. t_a ist der Spaltenvektor $(t_x, t_y, t_z)^{\mathsf{T}}$. Als zweiter Schritt wird eine Rotation R benötigt. Mit ihr werden die Achsen des WCS auf die Achsen des CCS abgebildet, so dass diese colinear sind.

t_a und R werden dann zu der 4×4-Matrix D zusammengefasst.

$$D = \begin{pmatrix} R & t_a \\ 000 & 1 \end{pmatrix}$$

Für die 3×3-Rotationsmatrix R gibt es nun mehrere Möglichkeiten, von denen zwei im Folgenden vorgestellt werden.

Eulerwinkel

Bei der Rotation mit Eulerwinkeln werden genau genommen drei Rotationen durchgeführt. Eine um jede Achse des Koordinatensystems um die Winkel φ_x, φ_y und φ_z. Durch Multiplikation dieser drei Rotationen erhält man dann die komplette Rotation R.

$$R = R_x R_y R_z = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\varphi_x & -\sin\varphi_x \\ 0 & \sin\varphi_x & \cos\varphi_x \end{pmatrix} \begin{pmatrix} \cos\varphi_y & 0 & \sin\varphi_y \\ 0 & 1 & 0 \\ -\sin\varphi_y & 0 & \cos\varphi_y \end{pmatrix} \begin{pmatrix} \cos\varphi_z & -\sin\varphi_z & 0 \\ \sin\varphi_z & \cos\varphi_z & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

=

$$\begin{pmatrix} \cos\varphi_y\cos\varphi_z & -\cos\varphi_y\sin\varphi_z & \sin\varphi_y \\ \sin\varphi_x\sin\varphi_y\cos\varphi_z + \cos\varphi_x\sin\varphi_z & -\sin\varphi_x\sin\varphi_y\sin\varphi_z + \cos\varphi_x\cos\varphi_z & -\sin\varphi_x\cos\varphi_y \\ -\cos\varphi_x\sin\varphi_y\cos\varphi_z + \sin\varphi_x\sin\varphi_z & \cos\varphi_x\sin\varphi_y\sin\varphi_z + \sin\varphi_x\cos\varphi_z & \cos\varphi_x\cos\varphi_y \end{pmatrix}$$

Dies ist die bekannteste Methode um eine 3-D-Rotation in Matrix-Form zu beschreiben. Sehr wichtig hierbei ist die Einhaltung der Reihenfolge. Da die Matrixmultiplikation ja nicht kommutativ ist gilt fast immer

$$R_x R_y R_z \neq R_y R_x R_z$$

Rodrigues

Die andere hier vorgestellte Methode für eine 3-D-Rotation ist das Verfahren nach Rodrigues. Hier wird nur um einen Winkel θ rotiert und zwar um einen normalisierten Einheitsvektor $u = (u_x, u_y, u_z)^\mathsf{T}$ mit $\|u\| = 1$.

Mit diesem Winkel und Vektor kann dann die Rotationsmatrix R berechnet werden.

$$R = uu^\mathsf{T} + (Id_3 - uu^\mathsf{T})\cos\theta + [u]_\times \sin\theta$$

Wobei $[\quad]_\times$ die Kreuzproduktmatrix ist, für die gilt

$$n \times v \Longleftrightarrow [n]_\times v$$

was von folgender Matrix erfüllt wird

$$[n]_\times = \begin{pmatrix} 0 & -n_3 & n_2 \\ n_3 & 0 & -n_1 \\ -n_2 & n_1 & 0 \end{pmatrix}$$

Das Ergebnis der obigen Formel in Form einer 3×3-Matrix ist

$$R = \begin{pmatrix} u_x^2 + (1-u_x^2)\cos\theta & u_x u_y(1-\cos\theta) - u_z \sin\theta & u_x u_z(1-\cos\theta) + u_y \sin\theta \\ u_x u_y(1-\cos\theta) + u_z \sin\theta & u_y^2 + (1-u_y^2)\cos\theta & u_y u_z(1-\cos\theta) - u_x \sin\theta \\ u_x u_z(1-\cos\theta) - u_y \sin\theta & u_y u_z(1-\cos\theta) + u_x \sin\theta & u_z^2 + (1-u_z^2)\cos\theta \end{pmatrix}$$

Für genauere Erklärungen zur Rodrigues-Formel kann in [1] oder in [2] nachgeschlagen werden.

1.3.4 Bild-Koordinatensystem *ICS*

Das Bild-Koordinatensystem ist das zweidimensionale Koordinatensystem des Bildes, das auf der Bildebene in der Kamera dargestellt wird.

Der Ursprung des ICS liegt im *Optischen Zentrum* [4]. Die x- und y-Achsen liegen parallel zu den beiden x- und y-Achsen des CCS.

Die Schreibweise für einen Punkt im ICS ist wie folgt: p^I.

1.3.5 Überführung CCS → ICS

Um nun einen Punkt p^C des CCS in einen Punkt p^I des ICS zu überführen, wird die perspektivische Projektion verwendet, die wie bereits schon erwähnt dem Ideal Pinhhole entspricht. Die Berechnung der x- und y-Koordinaten im ICS wird mit folgenden Formeln durchgeführt:

$$x^\mathsf{I} = f\frac{x^\mathsf{C}}{z^\mathsf{C}} \quad y^\mathsf{I} = f\frac{y^\mathsf{C}}{z^\mathsf{C}}$$

Daraus ergibt sich die *Projektionsmatrix* P_{per}

$$P_{per} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

Bei der Multiplikation mit dieser 4×4-Matrix erhält man im Ergebnis sämtliche benötigten Informationen.

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = \begin{pmatrix} fx^\mathsf{C} \\ fy^\mathsf{C} \\ z^\mathsf{C} \end{pmatrix} = \begin{pmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \begin{pmatrix} x^\mathsf{C} \\ y^\mathsf{C} \\ z^\mathsf{C} \\ 1 \end{pmatrix}$$

[4] Schnittpunkt von der Optischer Achse mit der Bildebene

1.3.6 Pixel-Koordinatensystem *PCS*

Das Pixel-Koordinatensystem ist nun das zweidimensionale Koordinatensystem des *digitalen Bildes*. Hier wird das analoge Bild des ICS auf die Pixelwerte des PCS diskretisiert. Der Ursprung liegt im Gegensatz zum ICS nicht im Optischen Zentrum sondern in der linken oberen Ecke des Bildes. Die x- und y-Achsen sind parallel zu den Achsen des ICS.

Die Schreibweise für einen Punkt im PCS ist wie folgt: p^{S}.

1.3.7 Überführung ICS → PCS

Um diese Diskretisierung des analogen Bildes durchzuführen, werden Kenntnisse über den Aufbau des CCD-Chips in der Kamera benötigt.

H_x und H_y sind die Koordinaten des Ursprungs der Bildebene. s beschreibt den Skew [5] der Pixel. Die Werte k_x und k_y werden mit der Brennweite f und der Höhe d_x und der Breite d_y eines einzelnen Pixels berechnet.

$$k_x = \frac{f}{d_x} \quad k_y = \frac{f}{d_y}$$

Aus diesen Werten läßt sich die 3×3-*Kalibrierungsmatrix* K erstellen, mit der ein Bildpunkt p^{I} in einen Pixelpunkt p^{S} überführt wird.

$$p^{\mathsf{S}} = Kp^{\mathsf{I}}$$

$$K = \begin{pmatrix} k_x & s & H_x \\ 0 & k_y & H_y \\ 0 & 0 & 1 \end{pmatrix}$$

Genauere Informationen über die Kamerakalibrierung finden sich im dazugehörigen Kapitel. Für die Überführung der Koordinatesysteme sollen diese Informationen erstmal ausreichen.

1.3.8 Überführung WCS → PCS

Wendet man sämtliche Matrizen hintereinander auf einen Punkt p^{W} des WCS an, so überführt man ihn direkt in den Punkt p^{S} des PCS. Die drei Matrizen K, P_{per} und D lassen sich zu einer einzelnen Matrix P_t zusammenfassen.

$$P_t = KP_{per}D$$

$$\begin{pmatrix} u \\ v \\ w \end{pmatrix} = P_t \begin{pmatrix} x^{\mathsf{W}} \\ y^{\mathsf{W}} \\ z^{\mathsf{W}} \\ 1 \end{pmatrix} = KP_{per}D \begin{pmatrix} x^{\mathsf{W}} \\ y^{\mathsf{W}} \\ z^{\mathsf{W}} \\ 1 \end{pmatrix}$$

Auch hier ist, wie schon bei der Rotation mit Eulerwinkeln, die richtige Reihenfolge der Matrizen zu beachten.

1.4 Beispiele für den Verwendungszweck

In diesem Abschnitt werden jetzt kurz zwei Anwendungsmöglichkeiten für die Rekonstruktion aus Bildern gezeigt. Eine tiefergehende Behandlung dieser erfolgt nicht. Es soll nur veranschaulicht werden, in welchen Bereichen diese Verfahren beispielsweise verwendet werden können.

Das erste Beispiel ist die Webcam der Universität Koblenz (1.3). Bei der Konstruktion kann man erkennen, an welchen Stellen eine Rotation der Gelenke möglich ist. Demnach befindet sich das COR innerhalb des auf dem Bild oberen Kastens. Somit sind COR und das innerhalb der Kamera liegende COP nicht identisch. Etwas, das bei den Kameramodellen explizit als erwünscht erwähnt wurde. Dadurch entsteht durch Rotieren der Kamera eine zusätzliche Standortveränderung der Kamera. Dieser Umstand verursacht einen erhöhten Rechenaufwand.

[5] Schräglage der Pixel bedingt durch Produktion

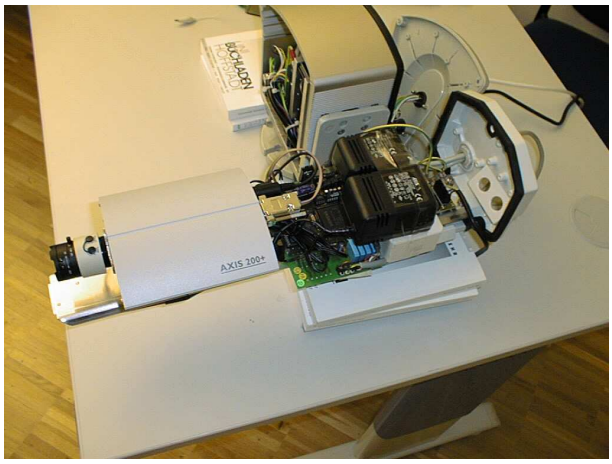


Abbildung 1.3: Webcam der Universität Koblenz

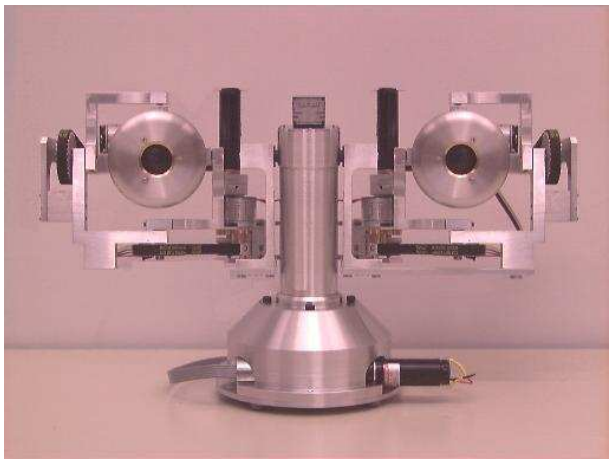


Abbildung 1.4: Roboterstereokopf mit zwei Kameras

Das zweite Beispiel sind Stereoköpfe von Robotern (1.4 und 1.5). Wie der Name schon vermuten läßt, besitzen diese Roboterköpfe zwei Kameras. Dadurch ist bei Stereoköpfen das Einhalten der Forderung, dass das COR gleich mit dem COP ist, sehr schwer. Mit zwei Kameras ist es für den Roboter möglich, Distanzen zu Gegenständen zu berechnen. Dadurch kann er zum Beispiel Hindernisse umgehen und Gegenstände ergreifen.

1.5 Schlussbemerkungen und Ausblick

Nachdem hier in knapper Form verschiedene Kameramodelle, Koordinatensysteme und deren Überführung vom einen ins andere vorgestellt wurden, ist noch ein interessanter Aspekt zu betrachten.

Es wird hier beschrieben, wie man aus einer Szenerie in der Welt das dazugehörige Bild in der Kamera mit Multiplikation verschiedener Matrizen berechnet. Bei der *Rekonstruktion aus Bildern* soll aber aus einem Bild (beziehungsweise mehreren Bildern) die ursprüngliche Szenerie berechnet werden. Folglich ist dieses Verfahren in umgekehrter Reihenfolge durchzuführen. Außerdem müssen davor noch zum Beispiel die Kalibrierungsmatrix K genau wie die Rotation R (siehe hierzu *8-Punkte-Algorithmus*) berechnet werden, damit die Matrixmultiplikation überhaupt vorgenommen werden kann. Somit sind die hier gezeigten Berechnungen nicht die ersten sondern die letzten, wenn man alles andere benötigte ermittelt hat. Wie man das macht und weitere interssante Themen werden in den folgenden Kapiteln behandelt.

Literaturverzeichnis

[1] Dietrich Paulus. Structure from motion. www.uni-koblenz.de/~paulus/idoc/sfmh.pdf, 7 2003. Skript zur Vorlesung im Sommersemester 2003.

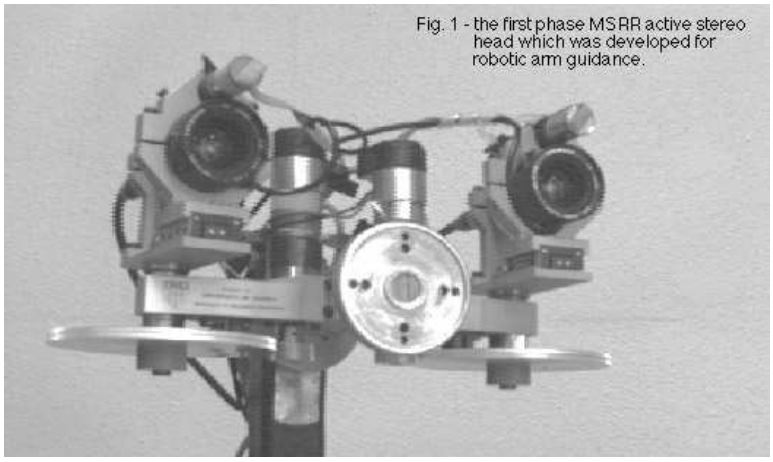


Abbildung 1.5: Roboterstereokopf mit zwei Kameras

[2] Dietrich Paulus. Medizinische bildverarbeitung. www.uni-koblenz.de/~paulus/idoc/medbvcompl.pdf, 2004. Skript zur Vorlesung im Wintersemester 2003/04.

[3] Sebastian Toelg. On the finite kinematics of visual fixation. Technical Report CS TR-3351 / CAR TR-736 / DACA 76-92-C-0009, University of Maryland, 9 1994.

Vortrag 2

Epipolargeometrie

Jochen Michel

27. November 2003

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
jmichel@uni-koblenz.de
http://www.uni-koblenz.de/~jmichel

2.1 Worum geht's?

"The epipolar geometry is the intrinsic projective geometry between two views. It is independent of scene structure, and only depends on the cameras' internal parameters and relative pose." [2]

Also:

- Geometrie, die dann zugrunde liegt, wenn man eine Szene von min. zwei Kameras von verschiedenen Positionen aus aufnimmt.
- Algebraisch wird dies in der E-Matrix (Essential Matrix) und der F-Matrix (Fundamental Matrix) modelliert.
- Die Zusammenhänge der Epipolargeometrie bilden die Grundlagen der 3D-Rekonstruktion von aufgenommenen Szenen.

2.1.1 Anordnung

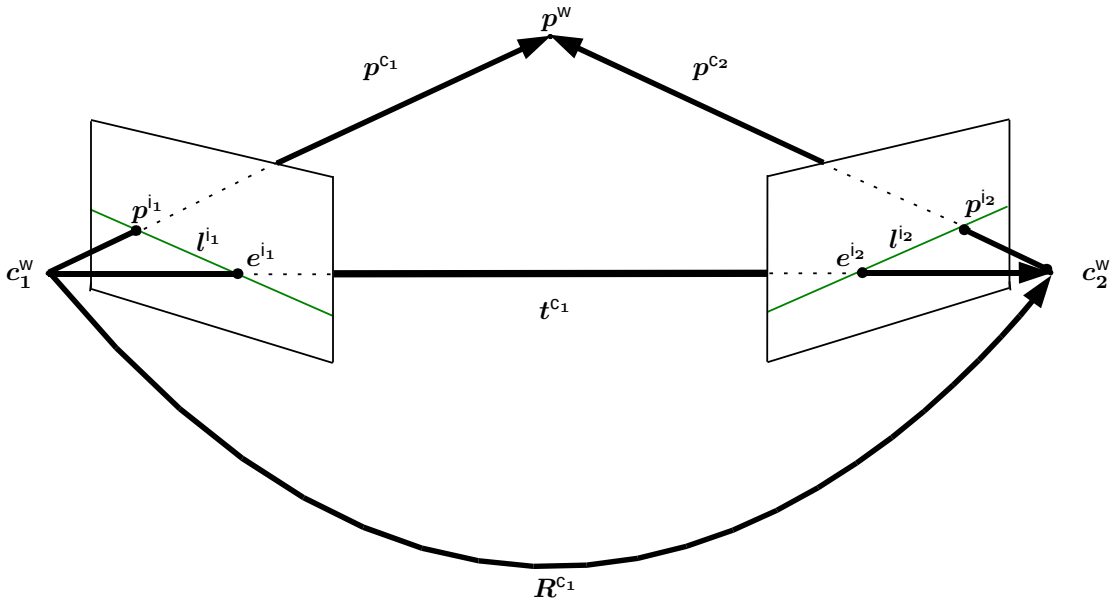


Abbildung 2.1: Geometrische Zusammenhänge der Epipolargeometrie

p^{w}	Punkt p im Weltkoordinatensystem,
c_1^{w}	Optisches Zentrum vom Kamera 1 im Weltkoordinatensystem,
c_2^{w}	Optisches Zentrum vom Kamera 1 im Weltkoordinatensystem,
p^{c_1}	Punkt p im Koordinatensystem von Kamera 1,
p^{c_2}	Punkt p im Koordinatensystem von Kamera 2,
p^{i_1}	Punkt p im Bildkoordinatensystem von Kamera 1,
p^{i_2}	Punkt p im Bildkoordinatensystem von Kamera 2,
l_1^{i}	Epipolarlinie in der Bildebene von Kamera 1,
l_2^{i}	Epipolarlinie in der Bildebene von Kamera 2,
e_1^{i}	Epipol in der Bildebene von Kamera 1 und
e_2^{i}	Epipol in der Bildebene von Kamera 2.

Ferner:

t^{c_1}	Translationsvektor um c_1^{w} in c_2^{w} zu überführen, wird im folgenden auch Basislinie genannt.
R^{c_1}	Rotationsmatrix um Vektor p^{c_1} auf p^{c_2} zu drehen.

Der hochgestellte Index ^{w} soll andeuten, dass es der zugehörige Vektor im Weltkoordinatensystem ($\mathbb{R}^3$) gegeben ist. Die Vektoren in den Kamerakoordinatensystemen mit hochgestelltem Index ^{c} sind ebenfalls aus $\mathbb{R}^3$. Der hochgestellte Index ^{i} soll schließlich zeigen, dass es sich um einen Vektor im Bildkoordinatensystem im projektiven (homogenen)

Raum ($\mathbb{P}^2$) handelt. Die einzelnen Koordinatensysteme werden im vorigen Kapitel von Christian Bauer ausführlich behandelt.

Wie in Abbildung 2.1 zu sehen ist, wird im Folgenden der Punkt $\boldsymbol{p}^{\mathsf{W}}$ betrachtet. Dieser ist genauso wie die beiden optischen Zentren der Kameras in Weltkoordinaten gegeben, also im $\mathbb{R}^3$. Der Ursprung des Weltkoordinatensystems kann hier frei gewählt werden, um die Sache zu vereinfachen, wird er gemeinhin in $\boldsymbol{c}_1^{\mathsf{W}}$ angenommen.

Der „Sehstrahl“ $\boldsymbol{p}^{\mathsf{c}_1}$, der vom optischen Zentrum der ersten Kamera ($\boldsymbol{c}_1^{\mathsf{W}}$) ausgeht und den Weltpunkt $\boldsymbol{p}^{\mathsf{W}}$ trifft, schneidet die Bildebene der ersten Kamera im Punkt $\boldsymbol{p}^{\mathsf{i}_1}$. Dieser Punkt wird, wie es auch sein hochgestellter Index zeigen soll, im Bildkoordinatensystem der ersten Kamera angegeben. Dieses Koordinatensystem hat seinen Ursprung in der Mitte der Bildebene und es werden hier homogene Koordinaten ($\mathbb{P}^2$) verwendet. Dieses Koordinatensystem ist nicht mit dem Pixelkoordinatensystem, das seinen Ursprung an der linken oberen Ecke der jeweiligen Bildebene hat und auf das später noch eingegangen wird zu verwechseln.
Die Epipole sind die Schnittpunkte der Basislinie mit den Bildebenen und werden ebenfalls als homogene Koordinaten angegeben.

Für die Epipolarlinien wird ebenfalls eine homogene Darstellung gewählt. Eine Linie lässt sich allgemein in der Form $ax + by + c = 0$ darstellen. In diesem Kontext ist eine Linie $\boldsymbol{l}$ ein Zeilenvektor der die Parameter a, b und c enthält: $\boldsymbol{l} = (a\ b\ c)$.

Wenn man nun den Punkt $\boldsymbol{c}_1^{\mathsf{W}}$ auf $\boldsymbol{c}_2^{\mathsf{W}}$ abbilden möchte, sind dafür im Allgemeinen zwei geometrische Operationen nötig: Eine Translation und eine Rotation.

Sowohl die Translation als auch die Rotation werden hier relativ zum Kamerakoordinatensystem der ersten Kamera ausgedrückt.
Der Translationsvektor $\boldsymbol{t}^{\mathsf{c}_1}$ verschiebt das optische Zentrum der ersten Kamera $\boldsymbol{c}_1^{\mathsf{W}}$ auf das optische Zentrum der zweiten Kamera $\boldsymbol{c}_2^{\mathsf{W}}$. Dieser Vektor bildet auch die sogenannte Basislinie, also die Verbindungslinie der beiden optischen Zentren.
Die Rotation wird in der Rotationsmatrix $\boldsymbol{R}^{\mathsf{c}_1}$ modelliert und ist so beschaffen, dass sie den Vektor $\boldsymbol{p}^{\mathsf{c}_1}$ auf $\boldsymbol{p}^{\mathsf{c}_2}$ dreht.

2.1.2 Mathematische Zusammenhänge

Wie bereits erwähnt kann man $\boldsymbol{p}^{\mathsf{c}_1}$ mit einer Translation $\boldsymbol{t}^{\mathsf{c}_1}$ und einer Rotation $\boldsymbol{R}^{\mathsf{c}_1}$ in $\boldsymbol{p}^{\mathsf{c}_2}$ überführen. Dies lässt sich algebraisch wie folgt formulieren:

$$\boldsymbol{p}^{\mathsf{c}_2} = \boldsymbol{R}^{\mathsf{c}_1}(\boldsymbol{p}^{\mathsf{c}_1} - \boldsymbol{t}^{\mathsf{c}_1}) \tag{2.1}$$

$$\Longrightarrow {\boldsymbol{R}^{\mathsf{c}_1}}^{\mathrm{T}} \boldsymbol{p}^{\mathsf{c}_2} = \boldsymbol{p}^{\mathsf{c}_1} - \boldsymbol{t}^{\mathsf{c}_1} \tag{2.2}$$

Außerdem brauchen wir noch algebraisch die Tatsache, dass $\boldsymbol{p}^{\mathsf{c}_1}$, $\boldsymbol{t}^{\mathsf{c}_1}$ und $\boldsymbol{p}^{\mathsf{c}_1} - \boldsymbol{t}^{\mathsf{c}_1}$ auf einer gemeinsamen Ebene, der Epipolarebene liegen. Es gilt also folgende Gleichung:

$$(\boldsymbol{p}^{\mathsf{c}_1} - \boldsymbol{t}^{\mathsf{c}_1})^{\mathrm{T}}(\boldsymbol{t}^{\mathsf{c}_1} \times \boldsymbol{p}^{\mathsf{c}_1}) = 0 \tag{2.3}$$

Nun können wir die Essential Matrix herleiten.

2.2 Die Essential Matrix

Die Essential Matrix, oder auch kurz $\boldsymbol{E}$-Matrix, fasst die zuvor erläuterten geometrischen Zusammenhänge in einer 3×3-Matrix zusammen. In sie gehen keine intrinsischen Kameraparameter ein sondern lediglich die Translation $\boldsymbol{t}^{\mathsf{c}_1}$ und die Rotation $\boldsymbol{R}^{\mathsf{c}_1}$. Zuerst wird die Essential Matrix hergeleitet, danach machen wir den Sprung vom Kamerakoordinatensystem zum Bildkoordinatensystem um schließlich die Epipolarbedingung angeben zu können.

2.2.1 Herleitung

Wenn man Gleichung 2.2 in in 2.3 einsetzt folgt unmittelbar:

$$(\boldsymbol{R}^{c_1\,\mathrm{T}}\boldsymbol{p}^{c_2})^{\mathrm{T}}(\boldsymbol{t}^{c_1}\times\boldsymbol{p}^{c_1}) = \tag{2.4}$$

$$\Longrightarrow \boldsymbol{p}^{c_2\,\mathrm{T}}\boldsymbol{R}^{c_1}(\boldsymbol{t}^{c_1}\times\boldsymbol{p}^{c_1}) = \tag{2.5}$$

$$\Longrightarrow \boldsymbol{p}^{c_2\,\mathrm{T}}\underbrace{\boldsymbol{R}^{c_1}[\boldsymbol{t}^{c_1}]_\times}_{\boldsymbol{E}}\boldsymbol{p}^{c_1} = 0 \tag{2.6}$$

Wobei $\boldsymbol{E}$ die sog. Essential Matrix ist und obiger Zusammenhang der Grundstein für die 3D-Rekonstruktion bei einem Stereokamerasystem darstellt.

Hinweis: $[\boldsymbol{t}^{c_1}]_\times$ ist eine sogenannte Kreuzproduktmatrix, eine spezielle Schreibweise für einen Vektor, damit man sich das Kreuzprodukt sparen kann und statt dessen eine elegantere Matrixmultiplikation berechnen kann.

Es sei $\boldsymbol{x} = \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix}$ und $\boldsymbol{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$.

Es gilt $\boldsymbol{x}\times\boldsymbol{y} = [\boldsymbol{x}]_\times\boldsymbol{y}$ mit $[\boldsymbol{x}]_\times = \begin{pmatrix} 0 & -x_3 & x_2 \\ x_3 & 0 & -x_1 \\ -x_2 & x_1 & 0 \end{pmatrix}$.

Um nun noch die Epipolarbedingung (epipolar constraint) angeben zu können, muss nun jedoch noch der Wechsel vom Kamerakoordinatensystem in das Bildkoordinatensystem vollzogen werden.

2.2.2 Die Epipolarbedingung

Das Bildkoordinatensystem hat den Ursprung in der Mitte der Bildebene. Wie bereits eingangs erwähnt, benutzt man hier meist homogene Koordinaten aus $\mathbb{P}^2$. Die Bildebene liegt vom optischen Zentrum der Kamera genau so weit entfernt, wie es die Brennweite F angibt.

Um also $\boldsymbol{p}^{c_1}$ auf die Bildebene zu projizieren, muss man diesen Vektor mit der Projektzionsmatrix $\boldsymbol{P}$ multipizieren:

$$\boldsymbol{p}^{i_1} = \boldsymbol{P}_{per}\,\boldsymbol{q}^{c_1} \tag{2.7}$$

wobei $\boldsymbol{P}_{per}$ bei der perspektivischen Projektion (Lochkamera-Modell) allgemein die Form

$$\boldsymbol{P}_{per} = \begin{pmatrix} F & 0 & 0 \\ 0 & F & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.8}$$

hat. Der Einfachheit nimmt man eine Brennweite von F= 1 an. Somit gilt $\boldsymbol{P}_{per} = Id_3$. Man kann also vom Kamerakoordinatensystem wie folgt ins Bildkoordinatensystem umformen:

$$\boldsymbol{p}^{c_2\,\mathrm{T}}\,\boldsymbol{R}^{c_1}[\boldsymbol{t}^{c_1}]_\times\,\boldsymbol{p}^{c_1} = 0 \tag{2.9}$$

$$\Longrightarrow (\boldsymbol{P_{per}}\boldsymbol{p}^{c_2})^{\mathrm{T}}\cdot\boldsymbol{E}\cdot\boldsymbol{P}_{per}\boldsymbol{p}^{c_1} = 0 \tag{2.10}$$

$$\Longrightarrow \boldsymbol{p}^{i_2\,\mathrm{T}}\cdot\boldsymbol{E}\cdot\boldsymbol{p}^{i_1} = 0 \tag{2.11}$$

Gleichung 2.11 heißt Epipolarbedingung oder epipolar constraint.
Wenn nun ein Punkt in der Welt auf die Bildebene der ersten Kamera abgebildet wird, kann man eine Linie im Bild der zeiten Kamera angeben, auf der dieser Punkt aus Sicht dieser zweiten Kamera liegt. Es gelten also weiterhin folgende Gleichungen:

$$\boldsymbol{l}^{i_2}\,\boldsymbol{E}\,\boldsymbol{p}^{i_1} = 0 \tag{2.12}$$

$$\boldsymbol{p}^{i_2\,\mathrm{T}}\,\boldsymbol{E}\,\boldsymbol{l}^{i_1\,\mathrm{T}} = 0 \tag{2.13}$$

Die Linien werden hier in einem 1×3-Zeilenvektor in der Form $\boldsymbol{l} = (a \quad b \quad c)$ notiert. Diese Notation passt zur homogenen 2D-Linengleichung $ax + by + c = 0$ wo man eine Linie eindeutig durch die Paramter a, b und c festlegen kann.

Wenn man nun also Punktzuordnungen in zwei Bildern, die von einem Stereokamerasystem aufgenommen wurden finden möchte, hat man es dank diesen Zusammenhängen sehr viel leichter: Anstatt zu einem Punkt in der ersten Bildebene alle Punkte in der zweiten Bildebene auf Zugehörigkeit zu testen, braucht man nur die Punkte auf der zugehörigen Linie in der zweiten Bildebene testen. Es gibt also zu jedem Punkt in der einen Bildebene eine Linie in der anderen, auf dem der korrespondierende Punkt zu suchen ist.
In der Praxis würde man hier anstatt einer Linie eine Nachbarschaft untersuchen, da es ja durch die Diskretisierung im Pixelkoordinatensystem (das gilt aber erst für die F-Matrix) Verzerrungen auftreten.

2.2.3 Eigenschaften der Essential Matrix

Weitere Eigenschaften der $\boldsymbol{E}$-Matrix sind:

- $\boldsymbol{E}$ ist eine 3×3 Matrix.
- Wenn man $\boldsymbol{E}$ mit einem skalaren Wert multipliziert, ist die Epipolarbedingung weiterhin gültig.
- Der Rang von $\boldsymbol{E}$ ist 2.
- $\boldsymbol{E}$ hat 5 Freiheitsgrade.
- Alle Epipolarlinien schneiden den jeweiligen Epipol.

2.3 Die Fundamental Matrix

Die Fundamental-Matrix oder kurz $\boldsymbol{F}$-Matrix bezieht sich auf das Pixelkoordinatensystem, also auf die Bilder, die man später tatsächlich vorliegen hat. Es gehen also sowohl die extrinsischen Parameter der Essential-Matrix als auch die intrinsischen der beiden Kameras in sie ein. Die intrinsischen Kameraparameter sind in die 3×3-Matrizen $\boldsymbol{K}_1$ für Kamera 1 und in $\boldsymbol{K}_2$ für Kamera 2 gefasst. Auch die Pixelkoordinaten werden homogen in $\mathbb{P}^2$ angegeben.

2.3.1 Herleitung

Von den Bildkoordinaten gelangt man zu dem Pixelkoordinatensystem durch folgende Zusammenhänge:

$$\boldsymbol{p}^{\mathrm{p}_1} = \boldsymbol{K}_1 \boldsymbol{p}^{\mathrm{i}_1} \iff \boldsymbol{p}^{\mathrm{i}_1} = {\boldsymbol{K}_1}^{-1} \boldsymbol{p}^{\mathrm{p}_1} \tag{2.14}$$

$$\boldsymbol{p}^{\mathrm{p}_2} = \boldsymbol{K}_2 \boldsymbol{p}^{\mathrm{i}_2} \iff \boldsymbol{p}^{\mathrm{i}_2} = {\boldsymbol{K}_2}^{-1} \boldsymbol{p}^{\mathrm{p}_2} \tag{2.15}$$

Wenn man nun Gleichung 2.14 und 2.15 in die Epipolarbedingung (s. Gleichung 2.11) einsetzt, erhält man folgendes:

$$\left({\boldsymbol{K}_2}^{-1} \boldsymbol{p}^{\mathrm{p}_2}\right)^{\mathrm{T}} \cdot \boldsymbol{E} \cdot \left({\boldsymbol{K}_1}^{-1} \boldsymbol{p}^{\mathrm{p}_1}\right) = 0 \tag{2.16}$$

$$\iff {\boldsymbol{p}^{\mathrm{p}_2}}^{\mathrm{T}} \cdot \underbrace{\left({\boldsymbol{K}_2}^{-1}\right)^{\mathrm{T}} \cdot \boldsymbol{E} \cdot {\boldsymbol{K}_1}^{-1}}_{\boldsymbol{F}} \cdot \boldsymbol{p}^{\mathrm{p}_1} = 0 \tag{2.17}$$

Es gilt also

$$\boldsymbol{F} = \left({\boldsymbol{K}_2}^{-1}\right)^{\mathrm{T}} \cdot \boldsymbol{E} \cdot {\boldsymbol{K}_1}^{-1} \tag{2.18}$$

2.3.2 Eigenschaften der Fundamental Matrix

Eigenschaften von $\boldsymbol{F}$:

- $\boldsymbol{F}$ ist auch eine 3×3 Matrix.
- ${\boldsymbol{p}^{\mathrm{p}_2}}^{\mathrm{T}} \cdot \rho \cdot \boldsymbol{F} \cdot \boldsymbol{p}^{\mathrm{p}_1} = 0$ ist bis auf einen skalaren Faktor ρ eindeutig
- Der Rang von $\boldsymbol{F}$ ist 2.

- $\boldsymbol{F}$ hat 7 Freiheitsgrade.
- Jedem Punkt im Bild kann seine Epipolarlinie mit $\boldsymbol{l}^{\mathrm{T}} = \boldsymbol{F} \cdot \boldsymbol{p}^{\mathrm{p}}$ in Pixelkoordinaten zugeordnet werden.
- Alle Epipolarlinien in beiden Bildebenen schneiden den jeweiligen Epipol.

2.4 Beispielbilder

Die beiden folgenden Bilder wurden aus [2] entnommen.

Abbildung 2.2: Linkes Kamerabild eines Stereokamerasystems mit ausgewählten Punktkorrespondenzen und zugehörigen Epipolarlinien im rechten Bild.

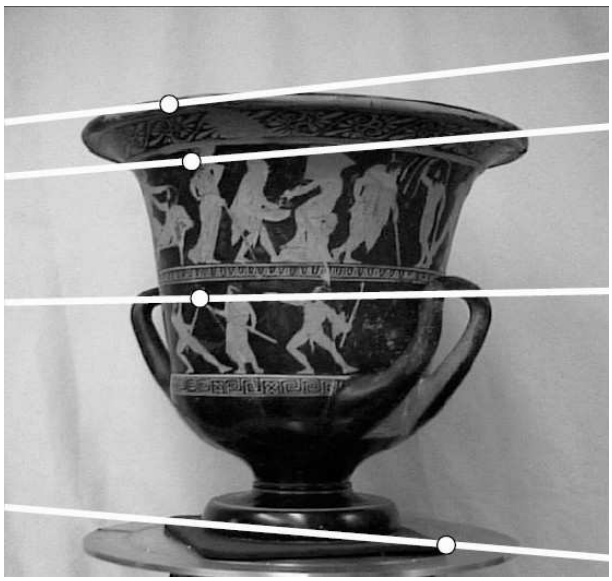


Abbildung 2.3: Rechtes Kamerabild eines Stereokamerasystems mit ausgewählten Punktkorrespondenzen und zugehörigen Epipolarlinien im linken Bild.

Die geometrische Anordnung aus Abbildung 2.1 kann man als Java-Applet bei [1] finden. Hier kann man die Anordnung auch dreidimensional drehen und sich so den Aufbau besser verdeutlichen.

Bei [3] kann man selber ein linkes und ein rechtes Stereokamerabild hochladen und dort automatisch Punktkorrespondezen finden lassen. Danach wird die $\boldsymbol{F}$-Matrix geschätzt und man bekommt beide Bilder im Webbrowser angezeigt. Nun kann man in einem Bild einen beliebigen Bildpunkt anklicken und im anderen Bild wird die zugehörige Epipolarlinie gezeichnet.

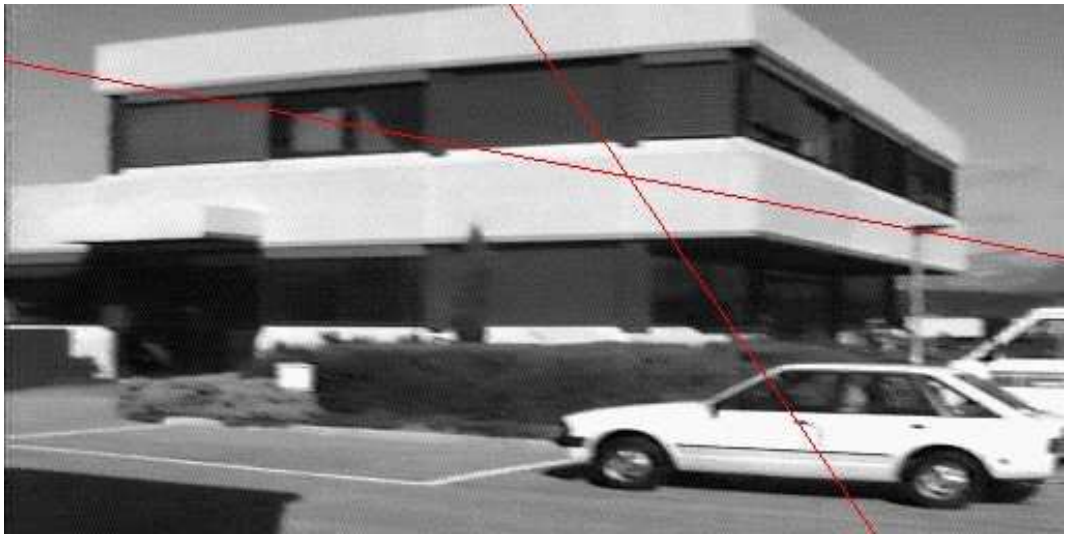


Abbildung 2.4: Linkes Kamerabild mit Epipolarlinien

Abbildung 2.5: Rechtes Kamerabild mit 2 ausgewählten Punkten (vordere Kante Fahrertür am Auto und linke obere Fensterecke am Gebäude

Literaturverzeichnis

[1] Sylvain Bougnoux. Learning Epipolar Geometry
http://www-sop.inria.fr/robotvis/personnel/sbougnou/Meta3DViewer/EpipolarGeo.html.

[2] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. 2000. Uni Koblenz Bib Signatur: INF 2002/7531 + INF 2002/8013.

[3] France INRIA, Robotvis. Computation of the epipolar geometry
http://www-sop.inria.fr/robotvis/demo/f-http/html.

[4] Dietrich Paulus. Skript zur Vorlesung „Structure from Motion“, SS03
http://www.uni-koblenz.de/~paulus/idoc/sfmh.pdf
Exercises and Tutorials by S. Bouattour and D. Droege.

Vortrag 3

Stereo-Sehen

Tina Walber

December 4th 2003

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
walber@uni-koblenz.de
http://www.uni-koblenz.de/~walber

3.1 Was ist Stereosehen

Mit Hilfe des Stereosehens können Informationen über die Tiefenwerte der Objekte in einem Bild gewonnen werden. Diese dreidimensionalen Informationen helfen z.B. dem Menschen sich im Raum zu bewegen und steuern die Hand-Augenkoordination. Der Mensch beherrscht diese Technik intuitiv. Die Bilder die durch die Netzhäute beider Augen geliefert werden, werden im Gehirn zu einem Gesamteindruck der betrachteten Szene „umgerechnet“.

Bei der Abbildung von Szenen durch Kameras gehen die dreidimensionalen Informationen verloren, da sämtliche Informationen in Richtung des Sehstrahls auf einen Punkt abgebildet werden. Soll das Stereosehen genutzt werden, um einem Bild Tiefeninformationen zu entnehmen, sind komplexe Mechanismen nötig, die im Folgenden vorgestellt werden sollen.

In dieser Arbeit werden zuerst die prinzipiellen Schritte eines Stereoverfahrens, sowie die besonderen Schwierigkeiten vorgestellt. Im Anschluss werden verschiedene Schritte detailliert behandelt: die Gewinnung von Bildern anhand von Stereokameras und die Kalibrierung dieser Kameras, die Zuordnung von korrespondierenden Bildelementen in den zwei Stereobildern und die geometrischen Grundlagen der Tiefenbestimmung.

3.2 Gewinnung von Tiefeninformationen

Zur Tiefenwertgewinnung wurden zum Teil recht unterschiedliche Ansätze entwickelt, über die hier eine kurz Übersicht gegeben werden soll. Es existieren passive und aktive Verfahren [3].

3.2.1 Aktive Verfahren

Als aktiv werden die Verfahren bezeichnet, die die betrachtete Szene durch eine spezielle Beleuchtung beeinflussen. Es kann z.B. strukturiertes Licht in Form eines Rasters auf eine Szene geworfen werden und anhand der Reflektionen dieses Musters Aussagen über die Räumlichkeit dieser Szene bewonnen werden (mit Hilfe der Trigonometrie 3.5). Eine andere Methode ist das Abstasten der Szene mit Hilfe eines Laserstrahls oder Ultraschalls, diese Verfahren werden Laufzeitverfahren genannt. Durch die Phasenverschiebung der, von der Szene reflektierten, Strahlen, können die Tiefenwerte rekonstruiert werden. Der größte Nachteil dieser aktiven Verfahren ist, dass sie durch bereits vorhandene Beleuchtung einer Szene beeinflusst werden können und somit nicht in jeder Situation anwendbar sind. Unter den entsprechenden Rahmenbedingungen kann allerdings eine hohe Genauigkeit erreicht werden, das kann heißen, dass die Umgebung abgedunkelt oder die Strahlung erhöht werden muss, was aber für den Menschen nicht immer unbedenklich ist.

3.2.2 Passive Verfahren

Passive Verfahren arbeiten mit Aufnahmen einer Szene, ohne diese zu beeinflussen. Zur Gewinnung der Tiefeninformation sind verschiedene Aufnahmen der gleichen Szene nötig, die zueinander in Relation gesetzt werden. Das können zum einen Stereobilder sein, d.h. Bilder, die gleichzeitig aber mit einer räumlichen Verschiebung aufgenommen werden oder Bildfolgen einer einzigen Kamera, die mit kurzem zeitlichen Abstand mehrere Bilder liefert. Es existieren weitere Verfahren, die anhand von Objekten im Bild oder Merkmalen wie Texturen, Konturen, Schatten usw. die gewünschten Informationen rekonstruieren.

Im Folgenden werden die Stereoverfahren, d.h. die Verfahren die zwei gleichzeitig aufgenommene Bilder zueinander in Bezug setzten, näher betrachtet.

3.2.3 Schritte eines Stereoverfahrens

Das prinzipielle Vorgehen bei der Tiefengewinnung mit Hilfe eines Stereoverfahrens beinhaltet die folgenden Schritte [3]:

1. Aufnahme des Stereobildes

   Am Anfang jedes Verfahrens steht die Gewinnung des Bildmaterials. Sie erfolg in der Regel mit Hilfe von Stereokameras 3.3, welche zwei Bilder der gleichen Szenen, aber mit leicht veränderten Blickwinkel aufnehmen. Die

Kalibrierung der Stereokameras ist notwendig, um den genauen geometrischen Aufbau der Kameras selbst und zueinander für die Tiefengewinnung zur Verfügung zu stellen.

2. Wahl und Detektion der Primitiva

   Die zwei Bilder müssen zueinander in Bezug gesetzt werden, d.h. in den Szenen müssen Elemente erkannt werden, die im rechten und linken Kamerabild abgebildet sind. Die Primitiva sind die Elemente im Bild, die als Erkennungsmerkmale dienen. Mögliche Primitiva sind einfache Bildpunkte, markante Stellen im Bild wie Eckpunkte oder Kantenelemente, prägnante Regionen im Bild oder auch komplexe Objekte. Es können auch verschiedene Primitiva kombiniert werden. Nach der Entscheidung für eine oder mehrere Arten von Primitiva müssen diese in beiden Bildern detektiert werden.

3. Korrespondenzproblem

   Die große Schwierigkeit und der entscheidende Schritt des Stereosehens ist die Zuordnung der korrespondierenden Primitiva in beiden Bildern. Zwei Primitiva korrespondieren, wenn sie Abbildungen des gleichen Weltpunktes sind. Die Schwierigkeit ist hierbei, dass die Zuordnung nicht immer eindeutig ist, da unterschiedliche Weltpunkte identisch abgebildet oder ein Weltpunkt aus zwei Blickwinkeln nicht identisch abgebildet werden kann. Deshalb werden zur Lösung des Korrespondenzproblems verschiedene Informationen herangezogen, welche in 3.4.3 vorgestellt werden.

4. Tiefenbestimmung

   Sind zwei Punkte detektiert kann, bei bekannter Kamerageometrie, der Tiefenwert eines solchen Punktes rekonstruiert werden.

5. Interpolation

   Um eine vollständige Tiefenkarte zu erhalten, müssen die gewonnenen Werte noch interpoliert werden.

3.3 Stereokameras

Eine Stereokamera nimmt zwei Bilder der gleichen Szene mit zwei nebeneinanderliegenden Kameras auf, die Aufnahmewinkel der beiden Bilder sind leicht unterschiedlich.

Die Eigenschaften einer Stereokamera hängen von den internen und externen Abbildungseigenschaften ab. Die internen Parameter beschreiben die Abbildungseigenschaften jeder einzelnen Kamera, wie die Brennweite, die Linsenverzerrung, usw. Mit den externen Parametern werden Eigenschaften wie die Lage und Ausrichtung der zwei Kameras zueinander beschrieben. Während die internen Parameter der Kameras konstant bleiben, solange nicht in deren Innenleben eingegriffen wird, kann die Position der Kameras zueinander von Aufnahme zu Aufnahme leicht variieren. Deshalb müssen die externen Parameter immer wieder neu bestimmt werden.

Diese Eigenschaften sind entscheidend für die Analyse einer Szene anhand der Epipolargeometrie, eine der wichtigsten Verfahren bei der Suche von korrespondierenden Primitiva. Die internen und externen Parameter werden bei der Kamerakalibierung bestimmt (vgl.3.3.2). Die gewonnen Stereobilder werden in der Regel vor der Weiterverarbeitung mit Hilfe dieser Parameter transformiert.

3.3.1 Stereokameramodelle

Es gibt zwei Stereokameramodelle, die sich in der Lage der beiden Kameras zueinander unterscheiden[3].

Bei der ersten Gruppe der Stereokameramodelle sind die beiden Kameras so zueinander ausgerichtet, dass sich ihre optischen Achsen in einem Punkt, dem Fixationspunkt, schneiden, wie in 3.1 dargestellt. Diese Modelle entsprechen vom Aufbau her dem menschlichen Sehen, da die Sehstrahlen der Augen sich ebenfalls in einem Punkt schneiden: dem Punkt des scharfen Sehens.

In der zweiten Gruppe von Stereokameramodellen werden die zusammengefasst, deren Fixationspunkt im Unendlichen liegt, d.h. bei denen die Sehstrahlen der beiden Kameras parallel verlaufen (vgl.3.2). Der Vorteil dieses Aufbau ist es, dass die Geometrie erheblich vereinfacht wird und damit die Rekonstruktion der Tiefeninformationen schneller und unkomplizierter realisiert werden kann. Gleichzeitig ist allerdings diese ideale Ausrichtung der Kameras nahezu unmöglich, da die exakt parallele Positionierung der Kameras zueinander kaum realisierbar ist und kleine produktionsbedingte Ungenauigkeiten zu weiteren Verzerrungen und Verschiebungen führen.

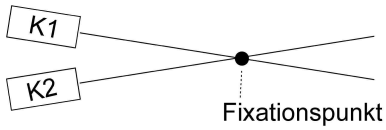


Abbildung 3.1: Stereokameramodell mit Fixationspunkt

In der Praxis wird deshalb davon ausgegangen, dass das zweite, ideale Kameramodell verwendet wird, um die anschließenden Stereoverfahren zu vereinfachen. Die Bilder die geliefert werden stammen aber normalerweise von einem Kameramodell ersten Typs. Sie müssen mit Hilfe der Bildvorverarbeitung so vorbehandelt werden, dass sie einer Aufnahme der idealen Kamera entsprechen, hierzu ist die Kalibrierung der Kameras nötig.

Abbildung 3.2: Stereokameramodell mit parallelen optischen Achsen

3.3.2 Kalibrierung

Mit der Kalibrierung der Stereokamera werden die internen und externen Parameter einer Kamera bestimmt. Dazu wird ein Testbild aufgenommen, dessen Position in der Welt bekannt ist und in dem leicht korrespondierende Punkte detektiert werden können, wie z.B. in den zwei Kalibrierungsbildern in 3.3.

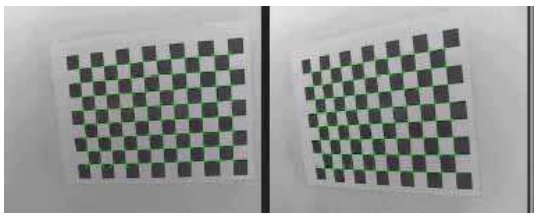


Abbildung 3.3: Aufnahmen eines Kalibrierungsmusters einer Stereokamera

Die Kalibrierung ist wichtig um später, mit verschiedenen geometrischen Verfahren, die Tiefeninformationen zu rekonstruieren.

Die Wahl eines Kalibrierungsverfahrens ist abhängig vom Kameramodell, der Anzahl und der Lage der Testpunkte, sowie der mathematischen Formulierung der Abbildung. Es existieren Verfahren, die nur lineare Beziehungen erlauben und andere, welche dann aber auch komplizierter sind, die auch nichtlinereare Beziehungen bestimmen können[3].

Das prinzipielle Vorgehen ist es, aus der Projektionsgleichung ein lineares Gleichungssystem zu erzeugen und mit Hilfe von mindestens 6 Passpunkten (die jeweils in beiden Bildern detektiert sein müssen) die Rotationsmatrix und den Translationsvektor, der das Bild der einen Kamera auf das der anderen abbildet, zu bestimmen.

Zu Lösen ist also die Modellgleichung, die den Zusammenhang von Welt- und Kamerakoordinaten eines Objektpunktes und seines Bildpunktes beschreibt. Die Gleichungen für die rechte Kamera lautet z.B. [3]:

$$\frac{d_x(X_p - C_x)}{1 + k_1 r^2 + k_2 r^4} = f\frac{r_{11}x_w + r_{12}y_w + r_{13}z_w + T_y}{r_{31}x_w + r_{32}y_w + r_{33}z_w + T_z} \tag{3.1}$$

$$\frac{d_y(Y_p - C_y)}{1 + k_1 r^2 + k_2 r^4} = f\frac{r_{21}x_w + r_{22}y_w + r_{23}z_w + T_y}{r_{31}x_w + r_{32}y_w + r_{33}z_w + T_z} \tag{3.2}$$

Für die linke Kamera gibt es eine entsprechende Formel.

d_x ist der Skalierungsfaktor, er kann den Herstellerangaben entnommen werden. Die Rechnerkoordinaten C_x werden im Bild bestimmt, die unverzerrten Kamerakoordinaten X_d stehen ebenfalls zur Verfügung, da d_x bekannt ist. Gesucht werden die Elemente r_ij der Rotationsmatrix sowie die drei Komponenten der Tranlationsvektors T.

3.3.3 Transformation der Stereobilder

Um die geometrischen Verfahren zur Berechnung der Tiefeninformation so einfach und damit so schnell wie möglich zu halten, wird davon ausgegangen, dass die Bilder mit einer idealen Kamera aufgenommen wurden (vgl.3.3.1). Vor der weiteren Verarbeitung müssen die Rohstereobilder in der Regel kalibriert werden, da ein ideales Kameramodell normalerweise nicht vorliegt.

Die Parameter hierfür wurden bei der Kamerakalibrierung bestimmt. Es müssen die internen Parameter berücksichtigt werden, um Linsenverzerrungen u.ä. auszugleichen und die externen Paramter, um die Bilder so zu tranformieren, als wären sie von zwei Kameras mit parallelen optischen Achsen aufgenommen worden. Dabei müssen nichtlineare Verzerrungen berücksichtigt werden, da z.B. bei der Verzerrung der Linse die Abweichungen nach außen hin zunehmen, wie in Abbildung 3.4 schematisch dargestellt.

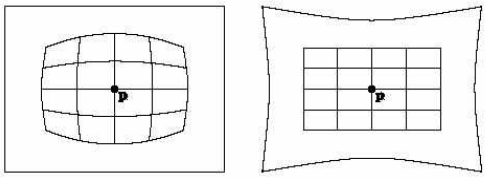


Abbildung 3.4: Verzerrte und kalibrierte Bildaufnahme

Die Ziele der Transformation sind die Korrektur der Linsenverzerrung, die Anpassung der Bildweiten der beiden Kameras und der Ausgleich der Rotation der beiden Kamerakoordinatsysteme.

Die Transformation kann entweder mit dem gesamten Bild geschehen, bevor die Primitiva darin bestimmt werden oder die bereits im Bild detektierten Segmente werden transformiert. Hier ist es allerdings von Vorteil, das gesamte Bild zu transformieren, da so bei der Detektion der Primitiva bereits geometrische Verfahren angewendet werden können (vgl. Epipolargemoetrie 3.4.3).

Die Rotation wird in zwei Schritten realisiert: Zuerst müssen die Parameter bestimmt werden. Danach wird das rotierte Bild berechnet, dabei muss interpoliert werden, da die neuen Werte zwischen den ursprünglichen Pixeln zu liegen kommen können.

In [3] wird folgende Vorgehensweise beschrieben:

Bestimmung der Rotationsparameter

Im einem ersten Schritt werden die Kamerakoordinatensysteme mit einer Rotationsmatrix R zuerst um die z-Achse und anschließend um die y-Achse gedreht, so dass das fast ideale Kamerakoordinatensystem entsteht, d.h. die x-Achsen der beiden Kamerakoordinatensysteme fallen mit der Stereobasis, also der Verbindung zwischen den beiden optischen Zentren der Kameras, zusammen. Im zweiten Schritt wird um die x-Achse gedreht, so daß y- und z-Achse parallel zu liegen kommen. Damit erhält man das ideale Kamerakoordinatensystem und die Rotationsparameter.

Interpolation der Grauwerte

Nachdem die Rotationsparameter im ersten Schritt bestimmt wurden, muss für jeden Bildpunkt der neue Grauwert bestimmt werden. Um dies zu realisieren wird im Rohstereobild die Koordinate bestimmt, in welcher der Abbildungsstrahl für den jeweils betrachteten Punkt die Bildebene schneidet. Da der neue Punkt in der Regel keine ganzzahlingen Koordinaten hat, also nicht genau auf einem Bildpunkt im Rohstereobild zu liegen kommt, muss der Grauwert aus der Nachbarschaft der neuen Koordinate bestimmt werden. Dies wird realisiert indem die Grauwerte der vier Nachbarn eine Ebene approximieren, die den gesuchten Grauwert liefert.

3.4 Korrespondenzanalyse

Mit Hilfe der Korrespondenzanalyse werden die zwei Bilder, die eine Stereokamera liefert, zueinander in Bezug gesetzt. Dies geschieht durch die Detektion von Abbildungen des gleichen Weltpunktes oder Objektes in beiden Bildern. Im einem der beiden Bilder werden die Primitiva ausgesucht, deren Pendant dann im anderen Stereobild bestimmt wird.

3.4.1 Stereoalgorithmen

Die verschiedenen Ansätze zum Lösen des Korrespondenzproblems unterscheiden sich hauptsächlich in der Wahl der Primitiva, also der Merkmale, die detektiert werden sollen. [3] Die erste Gruppe von Stereoalgorithmen sind merkmalsbasierte Verfahren. Die Primitiva, also Segmentierungsobjekte, werden aufgrund ihrer lokalen Eigenschaften gewählt und detektiert. Bei diesen Verfahren ist also sowohl die Wahl, als auch die Detektion der Primitiva abhängig vom Bildinhalt. Eine weitere Gruppe ist die der Differentiellen Verfahren. Als Primitiva werden die Bildpunkte selbst gewählt, der korrespondierende Punkt im anderen Bild wird mit Hilfe der lokalen Umgebung bestimmt. Die dritte Gruppe ist die der Blockmatchingverfahren. Hierbei handelt es sich um ein Korrelationsverfahren. Es werden regelmäßige Bildausschnitte gewählt, die Wahl ist also unabhängig vom Bildinhalt. Im korrespondieren Bild werden dann ähnliche Ausschnitte gesucht.

3.4.2 Korrespondenzproblem

Das größte Problem bei sämtlichen Verfahren ist die Mehrdeutigkeit dieser Zuordnungen. Zum einen ist die Korrespondenzanalyse bereits durch Abbildungsprobleme auf das diskrete Pixelsystem (Chip der Kamera) nicht eindeutig, hinzu kommen Mehrdeutigkeiten wegen ähnlicher Eigenschaften der Primitiva und besondere Eigenschaften der 3-D-Abbildung.

Bei der Abbildung eines Punktes auf das Pixelsystem des Chips einer Kamera, kann dieser Punkt so auf das diskrete Pixelsystem projiziert werden, dass er z.B. genau auf einem Pixel abgebildet wird, damit auch im Bild genau einem Punkt entspricht. Er kann aber auch so zu liegen kommen, dass er genau zwischen zwei Sensoren, also späteren Pixeln abgebildet wird und damit, je nach Größe, garnicht in der Abbildung auftaucht oder nicht weiter ein Punkt ist, sondern als Fläche, also auf mehreren nebeneinaderliegenden Pixeln abgebildet wird. Problematisch wird das, da ein Punkt in den beiden Aufnahmen einer Stereokamera unterschiedlich abgebildet werden kann, wie in Abbildung 3.5 schematisch dargestellt. Das bedeutet, dass ein einzelner Punkt im einem Bild einer Gruppe von Bildpunkten im anderen zugeordnet werden muss.

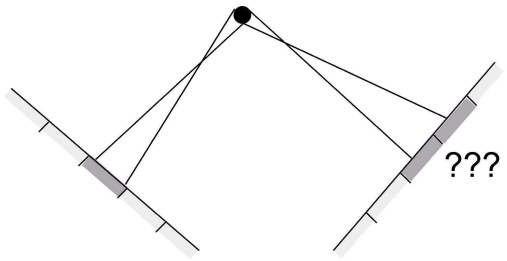


Abbildung 3.5: Abbildung eines Punktes auf verschiedenen Kameras

Hinzu kommen spezielle Eigenschaften von Abbildungen dreidimensionaler Szenen. Zum Beispiel die Verdeckung. Durch den leicht unterschiedlichen Blickwinkel der beiden Kameras gibt es Punkte, die auf einem Bild zu sehen sind, aber auf dem anderen durch einen andern Körper verdeckt werden (3.6, linke Darstellung). Ist dies der Fall, kann ein solcher Punkt im ersten Bild nicht als Primitivum gewählt werden. Es kann vorkommen, dass Körper in den beiden Stereoaufnahmen sehr unterschiedlich abgebildet werden. Das kann, wie bereits angesprochen mit den Abbildungseigenschaften auf ein Pixelsystem, oder auch mit der Form eines Körpers zusammenhängen. Wie in Bild 3.6 rechts kann ein Körper aus einer Sichtrichtung als Fläche abgebildet werden, aus einer anderen allerdings als Punkt. Durch den unterschiedlichen Winkel der Abbildung, können sich auch Farben und Helligkeiten verändern.

Im Folgenden werden verschiedene Ansätze zum Lösen des Korrespondenzproblems vorgestellt. In der Praxis werden, je nach Anwendung, meist verschiedene Ansätze kombiniert um ein möglichst genaues Ergebnis zu erziehlen.

3.4.3 Einschränkung der Mehrdeutigkeit

Es gibt viele verschiedene Ansätze zur Einschränkung der Mehrdeutigkeit bei der Detektion korrespondierender Element [3]. Die Möglichkeiten reichen von geometrischen Ansätzen bis hin zu Einschränkungen, die sich an der menschlichen Wahrnehmung orientieren.

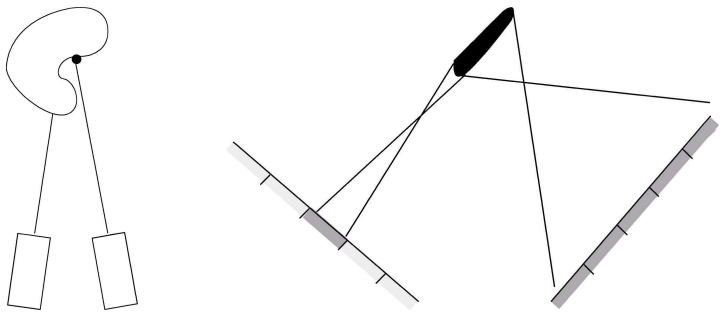


Abbildung 3.6: Probleme bei der Abbildung von 3D-Szenen: Verdeckung und Verzerrung

Eindeutigkeit

Jedes Primitivum entspricht genau einem Primitivum im korrespondierenden Bild. Das Problem ist, dass Abbildungsungenauigkeiten, Veränderungen der Szene durch Blickwinkelwechsel und ähnliche Probleme, wie in 3.4.2 beschrieben, diese Einschränkung beeinflussen.

Epipolargeometrie

Die Epipolargeometrie ist ein geometrisches Verfahren, um die Suche nach korrespondierenden Punkten einzuschränken. Wenn die Kamerageometrie bekannt ist, kann der Suchbereich für einen Punkt auf eine Gerade im anderen Bild beschränkt werden. Wird mit einem ideales Stereokameramodell gearbeitet, d.h. die optischen Achsen der beiden Kameras sind parallel, verlaufen diese epipolaren Linien horizontal, was eine weitere Vereinfachung darstellt.

Lokale Eigenschaften der Primitiva

Die Zuordnung der korrespondierenden Primitiva beruht natürlich auch auf deren lokalen Eigenschaften, wie z.B. Farbe oder Kontrast. Das Problem ist, dass diese Eigenschaften normalerweise nicht exakt übereinstimmen, da sie von verschiedenen Faktoren (wie Lichteinfall oder Blickrichtung) abhängig sind, die je nach Kameraposition variieren können. Deshalb müssen Werte gefunden werden, die einerseits gleiche Objekte erkennen und andereseits auch Variationsmöglichkeiten zulassen. Die Wahl der Primitiva fällt zwischen einfachen oder komplexen Segmentierungobjekten. Bei einfachen Objekten gibt es nur wenige lokale Eigenschaften, die Prüfung auf Übereinstimmung ist also leichter durchzuführen. Einfache Objekte treten aber auch häufiger im Bild auf, es konnt also häufiger zu Mehrdeutigkeiten. Eher komplexe Segmentierungsobjekte haben den Vorteil, dass die Zuordnung von Objekten aufgrund der vielen zur Verfügung stehenden Eigenschaften relativ eindeutig ist, die Detektion allerdings sehr aufwendig.

Kanten-Kontinuität

Um zu entscheiden, ob ein Punkt im Bild zu einem Segmentierungselement gehört, ist es eine Hilfe die Eigenschaften der auf dem Bild sichtbaren Objekte zu nutzen. Wenn Kanten detektiert wurden, kann davon ausgegangen werden, dass benachbarte Kantenelemente die gleiche Tiefe besitzen, also dem gleichen Primitvum zuzuordnen sind.

Form des Tiefenfeldes

Auch bei der Zuhilfenahme des Tiefenfeldes werden die Eigenschaften der 3D-Objekte im Bild genutzt. Ein Objekt wird bei der Projektion auf eine Fläche abgebildet, diese repräsentiert in der Abbildung eine Region mit ähnlichen Eigenschaften. Bildpunkte innerhalb dieser Region besitzen eine ähnliche Tiefe und sind damit den gleichen Primitiva zuzuordnen. Problematisch ist, dass das Tiefenfeld einer Szenen normalerweise nicht als Information vorliegt und das die Detektion ebendieser das Ziel des Verfahrens ist. Deshalb werden eher allgemeine Restriktionen für die Detektion angenommen: Z.B. dass es nur zu geringen Variationen in der Tiefe benachbarter Bildbereiche kommt.

Einschränkung aus einer Auflösungshierarchie

Die Detektion der Primitiva kann durch Einschränkungen aus einer Auflösungshierachie unterstützt werden: Aufnahmen der Szene liegen in verschiedenen Auflösungen vor und werden zueinander in Bezug gesetzt. Dabei wird meist der „grob-nach-fein“-Strategie gefolgt, d.h. zuerst werden in einer groben Auflösung Primitva detektiert, danach wird die Auflösung immer feiner und das Verfahren wiederholt. Bei grober Auflösung werden am wenigsten aber auch die größten Primitiva detektiert, je feiner die Auflösung desto mehr Details können in die Detektion einbezogen werden.

Ordnungs-Einschränkung

Diese Methode funktioniert nur bei Aufnahmen von Stereokameras mit parallelen optischen Achsen, d.h. im Bild kann mit Hilfe der Epipolargeometrie eine horizontale Linie konstruiert werden auf der korrespondierende Punkt zu finden ist. Die Zuordnung innerhalb einer solchen Bildzeile kann eingschränkt werden, wenn bereits ein Paar von korresponierenden Punkten bestimmt wurde. Liegt der zu suchenden Punkt z.B. im linken Stereokamerabild rechts eines Punktes der bereits detektiert wurde, so ist der Punkt im rechten Kamerabild ebenfalls rechts des korrespondierenden Punktes zu suchen. In Bild 3.7 entsprechen a und b dem bereits detektierten Primitivapärchen, die Korrespondenz zu Punkt x ist gesucht und muss sich rechts von Punkt b auf der epipolaren Linie befinden.

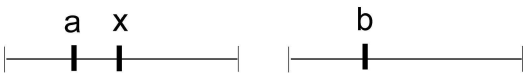


Abbildung 3.7: Beispiel Ordnungseinschränkung

Menschliche Wahrnehmenung

Der Mensch ist in der Lage durch die Benutzung verschiedener Informationsquellen die Tiefeninformation seiner Umgebung zu gewinnen. Das Stereosehen ist dabei die wichtigste und intuitivste Methode, dennoch genügen Informationen innerhalb des Bildes um räumliches Sehen zu ermöglichen, so kann auch ein Mensch der ein Auge verloren hat nach einiger Zeit räumliche Eindrücke gewinnen. Einige dieser Wahrnehmungsgesetzte können zur Einschränkung der Mehrdeutigkeit herangezogen werden. Hier einige Beispiele:

- Texturgradient

  Bei zunehmener Entfernung wird der Texturgradient kleiner, d.h. Muster und Strukturen im Bild werden kleiner.

- Farbintensität

  Bei Fotos nimmt die Farbintensität bei weiter Entfernung ab, da Partikel in der Luft die Farben abschwächen.

- Größe bekannter Objekte

  Gelingt es Objekte im Bild zu detektieren, deren Größe bekannt ist, kann festgestellt werden ob sich ein Objekt nah an der Kamera oder entfernt im Raum befindet. Dies ist z.B. möglich wenn das gleiche Objekt mehrere Male im Bild zu sehen ist.

- Verdeckung

  Objekte die gestaffelt hintereinander liegen verdecken das jeweils hintere Objekt.

3.5 Tiefenbestimmung

Nachdem korrespondierende Punktpaare detektiert wurden, kann die Tiefenbestimmung mit Hilfe der Trigonometrie stattfinden. Es gibt verschiedene Triangulierungsmethoden um die Tiefeninformation zu gewinnen. Sie haben gemeinsam, dass ein Objekt von verschiedenen Postionen aus betrachtet wird und durch die Unterschiede in den Bildern die Information gewonnen werden kann. Beispiele sind die Stereoskopie, die aktive Triangulierung, die Tiefenmessung durch Fokussierung und die konfokale Mikroskopie.[4]

Hier vorgestellt wird die Stereoskopie, die auch von vielen Lebewesen (auch dem Menschen) genutzt wird. Vorraussetzung sind zwei Kameras in Standardgeometrie, d.h. deren Achsen parallel angeordnet und Brennweiten- und ebenen

identisch sind. Durch Kalibierung und Transformation ist dieser Fall in unserem Ablauf gegeben. Der Aufbau wird in 3.8 dargestellt[2].

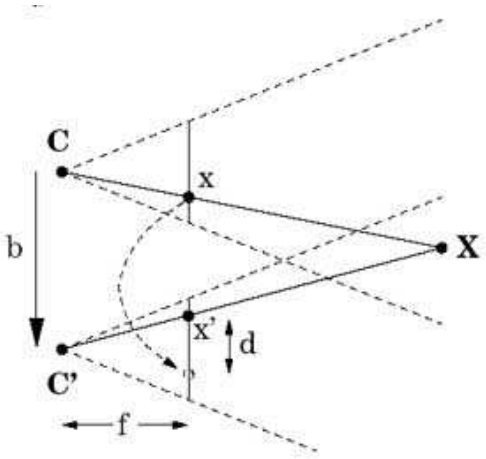


Abbildung 3.8: Triangulierung: Disparität

f bezeichnet die Brennweite der beiden Kameras, der Abstandsvektor b der beiden Achsen wird die stereoskopische Basis genannt. Die beiden Kameras unterscheiden sich nur um die Translation $\begin{pmatrix} b & 0 & 0 \end{pmatrix}^T$, die Abbildungen des 3D-Punktes X unterscheiden sich in den beiden Bildern nur in der x-Koordinate. d wird Disparität genannt und gibt den Abstand des 3D-Punktes zu den Kameras an.

Für die horizontalen Bildkoordinaten eines Punktes $X = (X, Y, Z)^T$ gilt:

$$x = f\frac{X}{Z} \tag{3.3}$$

und

$$x' = f\frac{X-b}{Z} = x - f\frac{b}{Z} = x - d \tag{3.4}$$

Die Disparität wird berechnet durch:

$$d = x - x' \tag{3.5}$$

Desto näher der Punkt an die Kamerapositionen rückt, desto größer wird die Disparität, bei großem Abstand wird sie klein.

Der Z-Wert des gesuchten Punktes wird berechnet durch:

$$Z = f\frac{b}{d} \tag{3.6}$$

3.6 Tiefenkarte

Die gewonnen Ergebnisse werden in einer Tiefenkarte gespeichert, d.h. für jeden Pixel wird der berechnete X-Wert in die Karte eingetragen. Zur Bestimmung der Tiefeninformation konnten nur markante Punkte im Bild ausgewählt werden um den korrespondierenden Bildpunkt im anderen Kamerabild zuzuordnen. Deshalb gibt es im entstehenden 3D-Bild nicht für jeden Punkt eine Tiefeninformation. Um eine geschlossenen Tiefenkarte zu erhalten und um kleine Fehler auszugleichen muss also interpoliert werden.

3.7 Anwendungsbeispiele

Stereoverfahren die räumliche Informationen aus Aufnahmen von zwei Kameras rekonstruieren können werden in vielen Bereichen eingesetzt. Solche Verfahren sind z.B. sehr hilfreich in der wissensbasierten Bildanalyse oder bei der Navigation und Orientierung im Raum, z.B. von Robotern. Auch die 3-D-Szenenrekonstruktion, wie sie für die Schaffung von virtuellen Welten von Nutzen ist, kann durch das Stereosehen automatisiert und damit vereinfacht werden. Zur Verdeutlichung der Bedeutung von Stereosystemen werden im Folgenden zwei Anwendungsbeispiele vorgestellt.

3.7.1 Flußvermessung mit Stereo-Kameramodellsystem

Ein Beispiel für den Einsatz von Stereosehen wird in der Studienarbeit von Roland Barthel vorgestellt[1]. Hier wird ein System zur Flussvermessung mit einem Stereokamerastystem vorgestellt. Das Ziel des Projektes ist die Automatische Schiffsführung auf Binnengewässern um das Unfallrisiko zu verringern. Das Stereokamerasystem soll dabei die Führung des Schiffs in Verbingund mit anderen Orientierungssystemen, wie Radar oder GPS, übernehmen.

Die Stereokameras sind auch die beiden Uferlinien ausgerichtet und sollen dort die Uferlinie sowie Landmarken oder Markierungsobjekte wie z.B. Schilder oder Radartonnen erkennen. Durch die räumliche Information der Kameras wird diese Erkennung ermöglicht.

Dieses Beispiel verdeutlicht, welche goßen Vorteile ein Stereosystem bei der Orientierung im Raum bieten kann.

3.7.2 Marssonde Mars Express

Am 2.Juni 2003 startete die Marssonde Mars Express um am 25.Dezember 2003 ihre polare Umlaufbahn um den Mars zu erreichen. Betrieben wird sie von der Europäischen Weltraumorganisation ESA, sie befindet sich auf der Suche nach Wasser und Leben auf dem Mars. Eines von sieben wissenschaftlichen Instrumenten an Bord ist eine Hochleistungsstereokamera mit dem Namen HRSC (High Resolution Stereo Colour Imager) die vom DLR (Deutsches Zentrum für Luft- und Raumfahrt) Institut in Berlin entwickelt wurde.

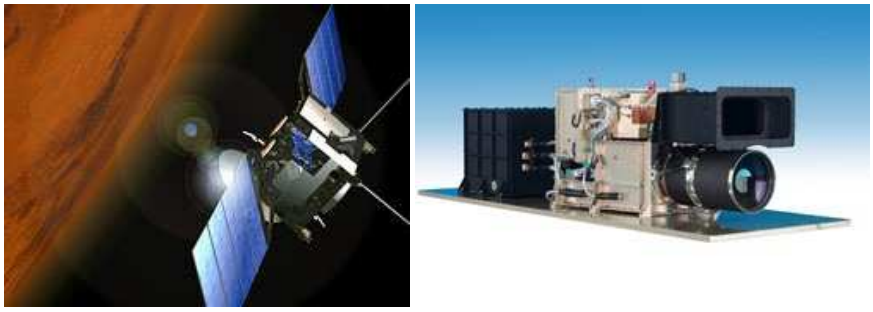


Abbildung 3.9: Marssonde Mars Express und HRSC-Kamera

Die Kamera ist 20 Kilo schwer und besteht aus 9 CCD-Zeilensensoren, von denen jeder aus mehr als 5000 CCD-Pixeln besteht. Die Zeilensensoren nehmen die Oberfläche des Mars aus jeweils leicht unterschiedliechen Postitionen auf, so dass aus den Einzelnbildern mit Hilfe von Stereoverfahren topographische Informationen sowie das Relief der Oberfläche berechnen lassen. Um die 3D-Landkarte des Mars zu erstellen wird die Vorwärtsbewegung des Raumschiffs ausgenutzts: Nach dem Scannerprinzip wird jeder Zeilensensor an der Oberfläche des Mars vorbeigeführt.

Nach Aufnahme der Bilder wurden diese zur Erde geschickt um dort möglichst schnell verarbeitet und den Wissenschaftlern zur Verfügung gestellt werden. Dazu mussten die Daten zuerst entpackt, dann dekomprimiert, radiometrisch kalibriert und geometrisch entzerrt werden. Diese aufwendige Rechen- und Prozessierungsarbeit nimmt einige Stunden in Anspruch.

Das Besondere an der Weltraummission ist, dass das erste mal ein Planet in hoher Auflösung, Farbe und 3D abgebildet wurde.

Die Informationen und Fotos zu diesem Projekt wurden der Internetseite des DRL entnommen: http://www.dlr.de/mars-express/

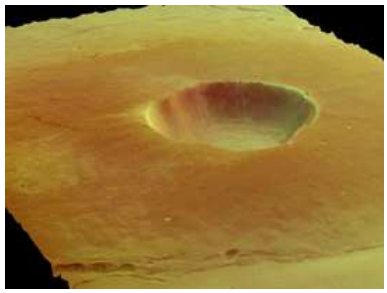


Abbildung 3.10: Hochaufgelöste 3D-Abbildung des Mars in Farbe

3.8 Fazit

Bei der Konstruktion eines Stereomodells müssen wichtige Entscheidungen in Abhängigkeit von der späteren Aufgabe des Systems getroffen werden. Als erstes muss die Entscheidung für ein Stereokameramodell fallen, dementsprechend müssen die aufgenommenen Bilder vorverarbeitet werden. Danach muss die Entscheidung für die Art der Segmentierungsobjekte fallen und die Möglichkeiten der Einschränkung der Mehrdeutigkeit durchdacht werden.

Es bleiben also viele Faktoren wählbar, so dass ein System entworfen werden kann, dass den speziefischen Anforderungen am Besten entspricht.

Die Bedeutung des Stereosehens wird aller Vorraussicht nach in den nächsten Jahren an Bedeutung gewinnen, da einerseits mehr vollautomatische Systeme entwickelt werden, die sich im Raum orientieren können müssen (wie z.B. Roboter oder Autopiloten), andererseits die Virtuelle Realität weiterentwickelt und die Möglichkeiten des 3D-Scans ausgebaut werden. Auch andere Anwendungsgebiete z.B. in der Medizin befinden sich in der Entwicklung.

Bei vielen dieser Anwendungen muss die dreidimensionale Information in Echtzeit geliefert werden, was heute noch sehr schwierig, bei komplexen Systemen nicht möglich ist. Doch mit der Steigerung der Leistungsfähigkeit von Computersystemen wird auch dies in der Zukunft möglich sein.

Literaturverzeichnis

[1] Roland Barthel. *Flußvermessung mit einem Stereo-Kamerasystem*. Studienarbeit, http://elib.uni-stuttgart.de/opus/volltexte/1999/318/, Stuttgart, 1998.

[2] Karsten Mühlmann. *Design und Implementierung eines Systems zur schnellen Rekonstruktion dreidimensionaler Modelle aus Stereobildern*. Dissertation, http://bibserv7.bib.uni-mannheim.de/madoc/volltexte/2002/56/, Mannheim, 2002.

[3] Stefan Posch. *Automatische Tiefenbestimmung aus Grauwert-Stereobildern*. Dissertation, Deutscher Universitäts Verlag, Wiesbaden, 1990.

[4] Andreas Pott. *Dynamisches Greifen von Gegenständen durch 3D-Formerfassung unter Verwendung eines echtzeitfähigen Sensorsystems*. Projektarbeit, http://www.mechatronik.uni-duisburg.de/pott/projektarbeit.pdf, Duisburg, 2002.

Vortrag 4

Kamerakalibrierung

Caroline Brunn

December 11th 2003

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
cmlbrunn@uni-koblenz.de
http://www.uni-koblenz.de/~cmlbrunn

4.1 Zieldefinition Kamerakalibrierung

Bei der Aufnahme einer 3D- Szene mit einer CCD- Kamera stellt sich das Problem, dass ihre 2D- Photographie auf dem Rechner verzerrt dargestellt wird. Um aber die Abbildung der 3D- Weltkoordinaten auf 2D- Rechner- Koordinaten korrekt zu vollziehen, ist die Kenntnis um die extrinsischen und intrinsischen Kameraparameter vonnöten. Mit diesen Werten lassen sich die Verzerrungen der aufgenommenen Szene zurückrechnen.

4.2 Extrinsische Parameter

Bei den extrinsischen Parametern handelt es sich um eine Rotationsmatrix und einen Translationsvektor, die eindeutig die Position der Kamera in der Welt bestimmen. Sie vollziehen die Transformation des szenenorientierten Welt- Koordinatensystems in das beobachterzentrierte Kamera- Koordinatensystem, will heißen: sie dienen dazu, die Sichtweise der Kamera einzunehmen. Die extrinsischen Kameraparameter müssen bei jeder Veränderung der Position der Kamera neu ermittelt werden.

4.3 Intrinsische Kameraparameter

Bei den intrinsischen Parametern handelt es sich um die individuellen Eigenheiten der CCD- Kamera, welche bekannt sein müssen, um die Verzerrung innerhalb der Kamera nachvollziehen zu können und somit den Wissensstand zu haben, diese Verzerrungen rückgängig zu machen. Die intrinsischen Kameraparameter sind unabhängig von der Position der Kamera und müssen für jede Kamera einmal und nie wieder ermittelt werden, wobei der Abstand zwischen den Sensoren und die Anzahl der Sensoren in horizontaler Richtung aus den Herstellerangaben der Kamera ersichtlich sind.

4.4 Funktionsweise der CCD- Kamera

Das Prinzip der CCD- Kamera besteht darin, dass das Licht auf die lichtempfindlichen Sensorelemente trifft, welche gleichmässig auf dem CCD- Chip im so genannten CCD- Array verteilt sind. Bei dieser Abbildung kommt es zu folgenden Ungenauigkeiten

- **Skalierungsfaktor** s_x

  Dieser Wert resultiert aus Ungenauigkeiten des CCD- Arrays, welche durch inkorrekte Abtastfrequenzen zwischen Bildspeicher und Kamera- Hardware bei der horizontalen Abtastung auftreten können.

- **radiale Verzerrung**

  Die radiale Verzerrung wird durch die Linse der Kamera verursacht. Dank ihrer linearen Eigenschaften, lässt sich die radiale Verzerrung relativ problemlos entfernen, wie im Punkt 4.7 erläutert wird.

4.4.1 Radiale Verzerrung

Bei der radialen Verzerrung handelt es sich um eine Verzerrung, die sich proportional zum Bildradius verhält. Mittels dieser direkten Abhängigkeit lässt sich die radiale Verzerrung durch folgende Formel modellieren und somit im Umkehrschluss auch entfernen.

$$(\kappa_1 r^2 + \kappa_2 r^4 + \kappa_3 r^6 + ...)$$

Theoretisch sind dabei unendlich viele Verzerrungsparameter zu betrachten, in der Praxis jedoch hat sich gezeigt, dass ein einziger Term für eine verwertbare Modellierung ausreichend ist.

$$\kappa_1 r^2$$

mit
r: Radius bzw. Abstand vom Hauptpunkt zum Punkt im Pixel-Koordinatensystem

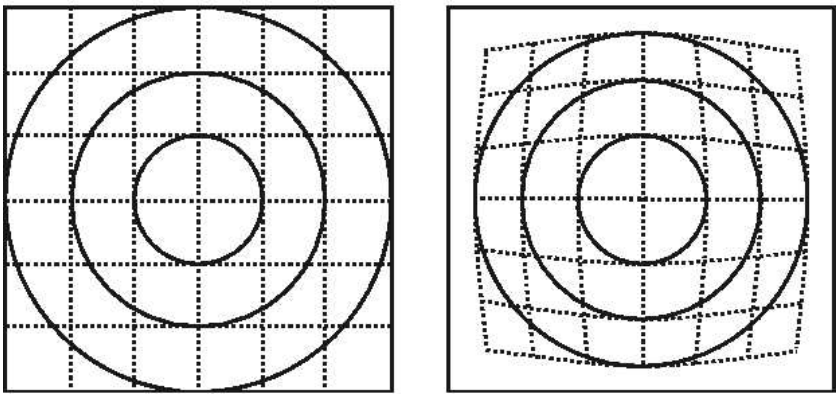


Abbildung 4.1:

4.5 Einigung auf Bezeichner

Hier sind die verschiedenen Koordinatensysteme und deren Koordinaten zusammengefasst, die uns die Arbeit mit den Daten der aufzunehmenden Szene aus unterschiedlichen Sichtweisen ermöglichen.

4.5.1 Koordinatensysteme

Koordinatensysteme	**Koordinaten**	**Ursprung**
Welt-Koordinatensystem	$(x_w, y_w, z_w)^T$	Ursprung o_w
Kamera-Koordinatensystem	$(x, y, z)^T$	Ursprung im optischen Zentrum o
Bild-Koordinatensystem *verzerrte Bild-Koordinaten* *unverzerrte Bild-Koordinaten*	 $(X_d, Y_d)^T$ $(X_u, Y_u)^T$	Ursprung (C_x, C_y) im Bild-Mittelpunkt
Rechner-Koordinatensystem *verzerrte Rechner-Koordinaten* *unverzerrte Rechner-Koordinaten*	 $(\hat{X}_P, \hat{Y}_P)^T$ $(X_P, Y_P)^T$	Ursprung in der linken oberen Ecke

4.6 Parameter- Bestandsaufnahme

Um die Schritte von Welt- Koordinaten auf ideale Rechner- Koordinaten nachvollziehen zu können, benötigen wir sämtliche extrinsische und intrinsische Kameraparameter. Einige davon stehen uns durch Herstellerdaten zur Verfügung, andere müssen noch berechnet bzw. ermittelt werden.
Die folgende Tabelle führt alle Parameter auf, die wir zur Kamerakalibrierung benötigen und beschreibt unseren derzeitigen Wissensstand.

4.6.1 Übersicht Parameter

extrinsische Parameter		
gesucht:		
Rotationsmatrix	$R = \begin{pmatrix} r_{11}r_{12}r_{13} \\ r_{21}r_{22}r_{23} \\ r_{31}r_{32}r_{33} \end{pmatrix}$	Welt- in Kamera-Koordinatensystem
Translationsvektor	$T = \begin{pmatrix} t_x \\ t_y \\ t_z \end{pmatrix}$	Welt- in Kamera- Koordinatensystem
intrinsische Parameter		
gesucht:		
	κ_1	Verzerrungskoeffizient durch Linse
Einheiten des Pixel-Koordinatensystem aus Sicht des Bild-Koordinatensystems	k_x, k_y mit $k_x = \frac{F}{d_x}$	Höhe und Breite eines Pixels
Brennweite	F	Abstand optisches Zentrum - Bildebene
gegeben bzw. berechenbar:		
Abstand zwischen den Sensoren	(d_x, d_y)	horizontaler und vertikaler Abstand im CCD-Array
Skalierungsfaktor	s_x	Abtastungenauigkeit des CCD-Chips
Anzahl Sensoren in horizontaler Richtung	N_{fx}	im CCD-Array
Herstellerangaben des Computers:		
Anzahl von Pixeln in einer Zeile	N_{cx}	

4.6.2 Berechnung des Skalierungsfaktors s_x

Den Skalierungsfaktor sehen wir als gegeben an, weil er sich mit folgender Formel unmittelbar aus den Herstellerangaben ableiten lässt:

$$d_x = \frac{d_x^{'}}{s_x}$$

mit:

$$d_x^{'} = d_x \frac{N_{cx}}{N_{fx}}$$

4.7 Radiale Entzerrung

Unser Ziel ist es nun, aus den verzerrten Bild- Koordinaten die unverzerrten zu berechnen.

4.7.1 Vorgehen

Für die radiale Entzerrung des Bildes ist das Wissen um die Koordinaten des Hauptpunktes Voraussetzung, welcher das Zentrum des Bildes im Frame- Buffer bezeichnet. Für unsere Berechnungen werden wir anstelle des Hauptpunktes (H_x, H_y) der Einfachheit halber den Bildebenen- Mittelpunkt (C_x, C_y) verwenden. Versuche in der Praxis haben nämlich gezeigt, dass Differenzen von bis zu 10 Pixel zum tatsächlichen Mittelpunkt keine signifikanten Auswirkungen auf das Ergebnis haben.

4.7.2 Rechenweg

Zwischen idealen und verzerrten Bild- Koordinaten herrscht folgender Zusammenhang:

- Zusammenhang ideale und verzerrte Pixel-Koordinaten:

$$\begin{pmatrix} X_P \\ Y_P \end{pmatrix} = (1 + \kappa_1 r^2) \begin{pmatrix} \hat{X}_P - H_x \\ \hat{Y}_P - H_y \end{pmatrix} + \begin{pmatrix} H_x \\ H_y \end{pmatrix}$$

- bzw. Zusammenhang verzerrte und unverzerrte Bild-Koordinaten:

$$\begin{pmatrix} X_d \\ Y_d \end{pmatrix} = (1 + \kappa_1 r^2) \begin{pmatrix} X_u \\ Y_u \end{pmatrix}$$

Löst man dieses nun jeweils nach der X- und nach der Y- Komponenten auf, erhält man die unverzerrten Bild- Koordinaten. Beispielhaft für die X- Komponente, ergibt sich das Ergebnis aus folgender Formel:

$$X_u = \frac{X_d}{(1 + \kappa_1 r^2)}$$

mit:

$$r = \sqrt{X_u^2 + Y_u^2}$$

oder

$$r = \sqrt{(X_P - H_x)^2 (Y_P - H_y)^2}$$

$$(AbstandPunkt - Hauptpunkt)$$

4.8 Kamerakalibrierung nach Tsai

In meiner Seminar-Arbeit habe ich mich für das Verfahren zur Kamerakalibrierung nach Roger Y. Tsai entschieden, weil es allgemein als schnellstes, flexibelstes und genauestes Verfahren anerkannt ist. Meine Arbeit behandelt die theoretische Durchführung seines Algorithmus zur Kamerakalibrierung, wobei bemerkt werden muss, dass in der Praxis leider weitere Schwierigkeiten auftreten, wie z.B. Vorzeichenberechnungen für numerische Stabiltät, auf die ich jedoch nicht weiter eingehen werde.

4.8.1 Ziel der Kamerakalibrierung

Wir befinden uns auf dem Stand der Dinge, dass die extrinsischen Parameter, die Brennweite, sowie der Koeffizient der Linsenverzerrung noch zu ermitteln sind, wobei uns aber die Herstellerangaben der Kamera und des Rechners zur Verfügung stehen. Weitere wertvolle Information wird uns die Testpunktkalibrierung liefern, die uns den Zusammenhang zwischen Welt- Koordinaten und verzerrten Rechner- Koordinaten ermittelt.

4.8.2 Das Vorgehen zur Kamerakalibrierung nach Tsai

Bevor wir uns an die Testpunktkalibrierung machen, werden die vier Abbildungsschritte von Welt- Koordinaten hin zu unverzerrten Rechner- Koordinaten zu den so genannten Modellgleichungen zusammengefasst. Danach stehen uns genügend Daten zur Verfügung, um in zwei Schritten die restlichen Kameraparameter zu berechnen.

- Modellgleichungen aufstellen
- Testpunktkalibrierung
- Berechnung der Kameraparameter in 2 Schritten
  1. R, t_x, t_y ermitteln
  2. F, t_z, κ_1 ermitteln

4.9 Abbildung Welt- Koordinaten auf Pixel- Koordinaten

Die vollständige Abbildung der 3D- Welt- Koordinaten auf 2D- Rechner- Koordinaten wird in vier Koordinatensystem-Transformationen vollzogen:

1. **Welt- Koordinaten (3D) in Kamera- Koordinaten (3D)**
   Mittels Rotation und Translation lässt sich das Welt- Koordinatensystem in das Kamera- Koordinatensystem transformieren, um den Blick der Kamera auf die aufzunehmende Szene nachzumodellieren.
2. **Kamera- Koordinaten (3D) in unverzerrte Bild- Koordinaten (2D)**
   Projiziert man nun perspektivisch die Szene aus der Sicht des Kamera- Koordinatensystems auf eine 2D- Ebene, so erhält man die idealen Bild- Koordinaten.
3. **unverzerrte Bild- Koordinaten (2D) in verzerrte Bild- Koordinaten (2D)**
   In der Realität jedoch führen spezifische Eigenheiten der Kamera dazu, dass die 3D- Szene verzerrt auf eine 2D- Ebene abgebildet wird, wobei nicht nur die radiale Verzerrung gemeint ist, welch sich relativ leicht entfernen lässt. In diesem Schritt vollziehen wir also die gesamte Verzerrung innerhalb der Kamera nach, um eine Entzerrung möglich zu machen.
4. **Bild-Koordinaten (2D) in Rechner- Koordinaten (2D)**
   Um eine korrekte Darstellung der Aufnahme am Rechner zu realisieren, müssen die Bild- Koordinaten durch Diskretisierung auf Rechner- Koordinaten abgebildet werden.

4.10 Modellgleichungen aufstellen

Die Abbildung von Welt- Koordinaten auf Rechner- Koordinaten lässt sich mathematisch in vier Schritten nachvollziehen:

1. Welt-Koordinatensystem → Kamera-Koordinatensystem

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix} = R \begin{pmatrix} x_w \\ y_w \\ z_w \end{pmatrix} + T$$

2. Kamera-Koordinaten → unverzerrte Bild-Koordinaten
   (perspektivische Projektion)

$$\begin{pmatrix} X_u \\ Y_u \end{pmatrix} = \frac{f}{z} \begin{pmatrix} x \\ y \end{pmatrix}$$

3. unverzerrte Bild-Koordinaten → verzerrte Bild-Koordinaten

$$\begin{pmatrix} X_d \\ Y_d \end{pmatrix} = (1 + \kappa_1 r^2) \begin{pmatrix} X_u \\ Y_u \end{pmatrix}$$

4. Bild-Koordinaten → Pixel-Koordinaten

$$\begin{pmatrix} X_p \\ Y_p \end{pmatrix} = \begin{pmatrix} \frac{X_d}{d_x} + C_x \\ \frac{Y_d}{d_y} + C_y \end{pmatrix}$$

Integriert man diese vier Gleichungen, erhalten wir je eine Modellgleichung für die X- und für die Y- Komponente.

1.Modellgleichung:

$$X_u = \frac{d_x(X_p - C_x)}{1 + \kappa_1 r^2} = f\frac{r_{11}x_w + r_{12}y_w + r_{13}z_w + t_x}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_z}$$

2.Modellgleichung:

$$Y_u = \frac{d_y(Y_p - C_y)}{1 + \kappa_1 r^2} = f\frac{r_{21}x_w + r_{22}y_w + r_{23}z_w + t_y}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_z}$$

4.10.1 Integration

Die folgenden Umformungen zeigen beispielhaft für die X- Koordinate auf, wie wir von den Abbildungsschritten zu der Modellgleichung gelangen.
Als Ausgangsgleichung nehmen wir die zweite Gleichung aus Punkt 4.9, welche die Abbildung von Kamera-Koordinaten auf unverzerrte Bild-Koordinaten bezeichnet:

$$\begin{pmatrix} X_u \\ Y_u \end{pmatrix} = \frac{f}{z}\begin{pmatrix} x \\ y \end{pmatrix}$$

Betrachtung der X-Koordinate, anhand derer wir beispielhaft die Integration der Abbildungsgleichung zur Modellgleichung durchführen möchten:

$$X_u = \frac{F}{z}x$$

Die erste Modellgleichung ergibt sich durch folgende Umformungen:

- Wir ersetzen X_u, indem wir die dritte Abbildungsgleichung nach X_u auflösen $X_u = \frac{X_d}{(1+\kappa_1 r^2)}$ und in die Ausgangsgleichung einsetzen

$$\frac{X_d}{1 + \kappa_1 r^2} = \frac{f}{z}x$$

- wir ersetzen x und z, indem wir die erste Abbildungsgleichung jeweils nach x und nach z auflösen $x = r_{11}x_w + r_{12}y_w + r_{13}z_w + t_x$
  $z = r_{31}x_w + r_{32}y_w + r_{33}z_w + t_z$ und in die aus dem vorherigen Schritt gewonnene Gleichung einsetzen

$$\frac{X_d}{1 + \kappa_1 r^2} = f\frac{r_{11}x_w + r_{12}y_w + r_{13}z_w + t_x}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_z}$$

- wir ersetzen X_d, indem wir die vierte Abbildungsgleichung nach X_d auflösen $X_d = d_x(X_p - Cx)$ und in die aus dem vorherigen Schritt gewonnene Gleichung einsetzen

$$\frac{d_x(X_p - C_x)}{1 + \kappa_1 r^2} = f\frac{r_{11}x_w + r_{12}y_w + r_{13}z_w + t_x}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_z}$$

  Voilà: die gewünschte Modellgleichung!

4.11 Testpunktkalibrierung

Bei der Testpunktkalibrierung handelt es sich um die Aufnahme von koplanar angeordneten Testpunkten, deren Welt-Koordinaten bekannt sind. Aufgrund der koplanaren Anordnung der Testpunkte, darf die z- Koordinate wegfallen, was die Rechnung deutlich vereinfacht. Die Testpunktkalibrierung liefert uns auf diese Weise Korrespondenzen zwischen Welt- Koordinaten und verzerrten Rechner- Koordinaten, von denen wir auf die verzerrten Bild- Koordinaten schließen können.
Auflösen der Abbildungsgleichung Bild-Koordinaten → Pixel-Koordinaten

$$\begin{pmatrix} X_p \\ Y_p \end{pmatrix} = \begin{pmatrix} \frac{X_d}{d_x} + C_x \\ \frac{Y_d}{d_y} + C_y \end{pmatrix}$$

nach X_d bzw. Y_d ergibt:

$$X_d = d_x(X_p - C_x)$$
$$Y_d = d_y(Y_p - C_y)$$

Aus den verzerrten Bild- Koordinaten wiederum lassen sich mittels radialer Entzerrung die unverzerrten Bild- Koordinaten berechnen, die dann mittels Diskretisierung auf die idealen Rechner- Koordinaten abgebildet werden können.
Diese aufgestellten Korrespondenzen aus der Testpunktkalibrierung liefern uns nunmehr genügend Information, um die noch fehlenden Kameraparameter zu berechnen.

4.12 Vorgehen Schritt 1

- gesucht sind folgende Kameraparameter:R, t_x, t_y

1. Integration der Modellgleichungen
   - Aufstellen eines linearen Gleichungssystems durch Division der Modellgleichungen durcheinander
   - Auflösen nach den Unbekannten
   - Lösen mit der Pseudo-Inversen-Methode

$$Ax = y$$
$$\Leftrightarrow (A^T A)^{-1} A^T y = x$$

2. Extrahierung von t_y
3. Berechnung der restlichen Koeffizienten der Rotationsmatrix

4.12.1 Durchführung Schritt 1

Im ersten Schritt der Kamerakalibrierung nach Tsai suchen wir folgende Kameraparameter:

- Die Rotationsmatrix R
- Die Komponenten t_x, t_ydes Translationsvektors T

Rechnung Teil 1

1. Aufstellen eines linearen Gleichungssystems
   - Integration der Modellgleichungen
     Die beiden Modellgleichungen werden miteinander integriert, indem man sie durcheinander dividiert. Die Division der Modellgleichung 1 durch die Modellgleichung 2 ergibt folgende Formel:

$$\frac{X_d}{Y_d} = \frac{r_{11}x_w + r_{12}y_w + t_x}{r_{21}x_w + r_{22}y_w + t_y}$$

- Ausmultiplizieren und Umformen:
  Um die gesuchten Parameter von den gegebenen zu trennen, wird die integrierte Modellgleichung zu folgender Vektormultiplikation umgeformt:

$$(Y_d x_w, Y_d y_w, Y_d, -X d x_w, -X d y_w) \begin{pmatrix} \frac{r_{11}}{t_y} \\ \frac{r_{12}}{t_y} \\ \frac{t_x}{t_y} \\ \frac{r_{21}}{t_y} \\ \frac{r_{22}}{t_y} \end{pmatrix} = X_d$$

  mit den Unbekannten:

$$(\frac{r_{11}}{t_y}, \frac{r_{12}}{t_y}, \frac{t_x}{t_y}, \frac{r_{21}}{t_y}, \frac{r_{22}}{t_y})$$

- Lösen mit der Pseudo-Inversen-Methode

2. t_y extrahieren
   Bis hierher steht uns der Wert der Y- Komponente des Translationsvektors lediglich als Kehrwert im Skalierungsfaktor $\frac{1}{t_y}$ zur Verfügung, lässt sich aber mit Hilfe der Untermatrix extrahieren.

   - Untermatrix aus bereits bekannten Rotations-Koeffizienten aufstellen

$$C = \begin{pmatrix} r'_{11} & r'_{12} \\ r'_{21} & r'_{22} \end{pmatrix} = \begin{pmatrix} \frac{r_{11}}{t_y} & \frac{r_{12}}{t_y} \\ \frac{r_{21}}{t_y} & \frac{r_{22}}{t_y} \end{pmatrix}$$

   - Falls die Untermatrix C Nullzeilen besitzt, lässt sich t_y einfach berechnen. Ansonsten gilt folgende Formel:

$$t_y^2 = \frac{S_r - \sqrt{S_r^2 - 4(r_{11}r_{22} - r_{21}r_{12})^2}}{2(r_{11}r_{22} - r_{21}r_{12})^2}$$

     mit

$$S_r = r'^2_{11} + r'^2_{12} + r'^2_{21} + r'^2_{22}$$

3. Restliche Koeffizienten der Rotationsmatrix ermitteln

$$R = \begin{pmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{pmatrix}$$

Gefunden haben wir bisher lediglich vier der insgesamt neun Komponenten der Rotationsmatrix, nämlich: $r_{11}, r_{12}, r_{21}, r_{22}$
Die Rotationsmatrix zeichnet sich dadurch aus, dass sie sowohl rechtshändig ist, was bedeutet, dass ihre Determinante den Wert 1 besitzt, als auch orthonormal, was bedeutet, dass die Norm ihrer Zeilenvektoren 1 beträgt und diese Vektoren eine orthogonale Basis bilden. Unter Berücksichtigung dieser Eigenschaften lassen sich die restlichen Komponenten problemlos berechnen, wie der folgende Rechenweg zeigt:

Rechnung Teil 2

Berechnung der noch fehlenden Komponenten der Rotationsmatrix R

- gegeben: $r_{11}, r_{12}, r_{21}, r_{22}$
- gesucht: $r_{13}, r_{23}, r_{31}, r_{32}, r_{33}$

1. Berechnung von r_{13} und r_{23}
   Aus der Bedingung, dass die Länge der Zeilenvektoren 1 beträgt, lassen sich unter der Verwendung der bereits ermittelten Komponenten von R sowohl r_{13} als auch r_{23} berechnen:

$$\sqrt{r_{i1}^2 + r_{i2}^2 + r_{i3}^2} = 1 = r_{i1}^2 + r_{i2}^2 + r_{i3}^2$$

$\Rightarrow$

$$r_{13} = \sqrt{(1 - r_{11}^2 + r_{12}^2)}$$
$$r_{23} = \sqrt{(1 - r_{21}^2 + r_{22}^2)}$$

2. Berechnung der restlichen Komponenten r_{31}, r_{32}, r_{33}
   Bei den noch fehlenden Komponenten handelt es sich um den dritten Zeilenvektor der Rotationsmatrix. Da die Zeilenvektoren eine orthogonale Basis bilden, ergibt sich dieser aus dem Kreuzprodukt der beiden anderen, weil das Kreuzprodukt zweier Vektoren ja den Vektor ergibt, der senkrecht auf diesen beiden steht. Somit können wir die Rotationsmatrix als berechnet ansehen.

4.13 Vorgehen Schritt 2

- gesucht sind folgende Kameraparameter: F, t_z, κ_1

1. Reduktion der Modellgleichungen
   - Gefundene Parameter aus Schritt 1 einsetzen
   - Lineares Gleichungssystem aufstellen
   - Lösen mit minimalem quadratischen Fehler

4.13.1 Durchführung Schritt 2

Im zweiten Schritt der Kamerakalibrierung nach Tsai suchen wir folgende Kameraparameter:

- Die Brennweite F
- Die letzte Komponente $t_z$des Translationsvektors T
- Den Verzerrungskoeffizienten κ_1 der Linse

Rechenweg

Uns stehen nunmehr genügend Werte zur Verfügung, um die restlichen Kameraparameter mit Hilfe der 2. Modellgleichung zu berechnen. Das Einsetzen von konkreten Werten für

$$y = x_w r_{21} + y_w r_{22} + z_w r_{23} + t_y$$

und

$$z = x_w r_{31} + y_w r_{32} + z_w r_{33}$$

in die 2. Modellgleichung

$$Y_u = \frac{d_y(Y_p - C_y)}{1 + \kappa_1 r^2} = f\frac{r_{21}x_w + r_{22}y_w + r_{23}z_w + t_y}{r_{31}x_w + r_{32}y_w + r_{33}z_w + t_z}$$

führt zu folgender Reduktion:

$$\frac{d_y(Y_p - Cy)}{1 + \kappa_1 r^2} = F\frac{y}{z + t_z}$$
$$= \frac{Y_d}{1 + \kappa_1 r^2} = F\frac{y}{z + t_z}$$

Stellt man daraus ein Gleichungssystem der folgenden Art auf, lassen sich die verbleibenden unbekannten Kameraparameter mit minimalem quadratischen Fehler berechnen:

$$yF - Y_d t_z + yr^2 F\kappa_1 = zY_d$$

Literaturverzeichnis

[1] Stefan Posch. *Automatische Tiefenbestimmung aus Grauwert-Stereobildern*. Dissertation, Deutscher Universitäts Verlag, Wiesbaden, 1990.

[2] Roger Y. Tsai. *A Versatile Camera Calibration Technique for High- Accuracy 3D Machine Vision Metrology Usíng Off-The-Shelf TV Cameras and Lenses*. IEEE Journal of Robotics and Automation, August 1987.

Vortrag 5

Image Warping

Christina Lacalli

18. Dezember 2003

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
clacalli@uni-koblenz.de
http://www.uni-koblenz.de/∼clacalli

5.1 Einführung

Image Warping ist ein Teilgebiet der Bildverarbeitung, das sich mit geometrischen Transformationstechniken befasst. Ein *Warp* ist dabei gleichzusetzen mit einer geometrischen Transformation, also einer Operation, die die räumliche Beziehung zwischen den Bildpunkten eines Bildes neu definiert. Dabei kann es sich um einfache Rotationen, Skalierungen und Translationen handeln, beinhaltet aber auch aufwendigere Transformationen, die Grund für die häufige Auffassung sind, Image Warping befasse sich ausschliesslich mit stark verzerrten Bildern.

5.1.1 Anwendungsgebiete

Image Warping findet seit mehr als 20 Jahren vor allem in der Fernerkundung, der medizinischen Bildverarbeitung, der Bilderkennung und in der Computergraphik Anwendung. In der Fernerkundung hat man es beispielsweise oft mit einer Menge von Bildaufnahmen (z.B. Luftbildaufnahmen) der gleichen Geländeformation/des gleichen Gebiets zu tun. Jedes einzelne Bild dieser Menge gilt es auf jedes der anderen Bilder auszurichten, so dass alle korrespondierenden Punkte übereinstimmen. Diesen Prozess nennt man auch *Bildregistrierung*. Dabei kann es zu Ausrichtungsfehlern kommen, beispielsweise wenn die Einzelbilder zur gleichen Zeit, aber von unterschiedlichen Kameras aufgenommen werden, wobei jede Kamera spezifische Verzerrungseigenschaften hat, oder wenn die Bilder zwar von der gleichen Kamera, aber zu unterschiedlichen Zeitpunkten aufgenommen werden (unterschiedliche Blickwinkel). Image Warping dient dazu, diese Verzerrungen zu korrigieren und eine akkurate Bestimmung der räumlichen Beziehungen und des Maßstabs zu ermöglichen/gewährleisten. In Abbildung 5.1 sind typische Bildverzerrungen zu sehen, die sich in interne, bezogen auf die spezifischen Verzerrungseigenschaften des Bildsensors und externe Verzerrungen, bezogen auf die spezifischen Begebenheiten der aufgenommenen Szene, unterteilen lassen.

In der medizinischen Bildverarbeitung spielen geometrische Transformationen u.a. in der digitalen Radiologie eine wichtige Rolle. Die digitale Substraktions-Angiographie (DSA) ermöglicht beispielsweise die Darstellung isolierter Gefäßbilder: Ein Röntgenbild der Körperregion wird während einer Kontrastmittelinjektion aufgenommen. Ein Maskenbild, das vor der Injektion aufgenommen wurde, wird anschließend davon subtrahiert, um die Gefäße zu isolieren. Bewegt sich der Patient während der Aufnahmen, kommt es zu Ausrichtungsfehlern, die korrigiert werden müssen.

Im Gegensatz zu den oben aufgeführten Beispielen ist das Ziel in der Computergraphik oftmals nicht geometrische Korrektheit, vielmehr sind geometrische Verzerrungen beispielsweise beim *Texture Mapping* erwünscht. Hierbei werden 2-D Texturen auf 3-D Oberflächen abgebildet und anschließend auf einen 2-D Bildschirm zurückprojiziert.

Image Warping findet nicht zuletzt im Graphikdesign Anwendung, um z.B. interessante visuelle Effekte zu erzielen.

5.1.2 Prinzip des Image Warpings

Das Prinzip des Image Warpings wird im Folgenden anhand eines einfachen Beispiels (vgl. Abbildung 5.2) erläutert. Gegeben ist ein verzerrtes diskretes *Inputbild*, das auf einem gleichmäßigen Raster, dem *input sampling grid*, definiert ist. Gesucht ist das entzerrte diskrete *Outputbild*, das ebenfalls auf einem gleichmäßigen Raster, dem *output sampling grid* definiert ist. Die Mappingfunktion ist durch einen Kalibrierungsschritt gegeben. Das Outputbild kann entweder durch *Forward Mapping* oder durch *Backward Mapping* generiert werden, wobei letzteres zu bevorzugen ist. Dabei wird die *inverse* Mappingfunktion auf das output sampling grid angewendet. Das daraus resultierende *resampling grid* wird

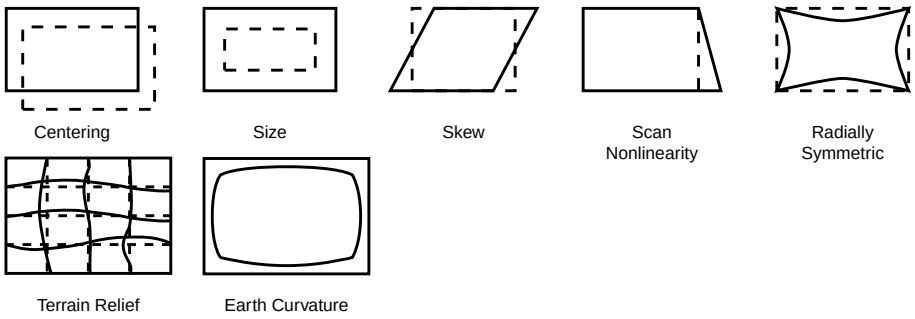


Abbildung 5.1: Interne (oben) und externe (unten) Bildverzerrungen.

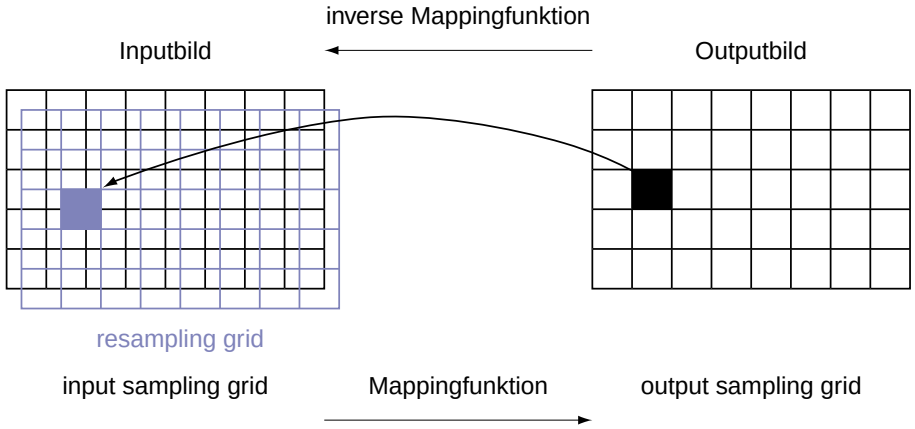


Abbildung 5.2: Image Warping.

auf das input sampling grid projiziert und spezifiziert für den jeweiligen Rasterpunkt im Outputbild die Position des zugehörigen Wertes im Inputbild. Der Intensitätswert an dieser Position wird dann in das Outputbild übertragen.

Folgendes Problem tritt nun auf: Wie in Abbildung 5.2 zu sehen, stimmen resampling grid und input sampling grid im Allgemeinen nicht überein. Dies ist darauf zurückzuführen, dass der Wertebereich der inversen Mappingfunktion reellwertig ist, der Definitionsbereich des Inputbildes dagegen auf ganzen Zahlen definiert ist. Definitionsbereich des Inputbildes und Wertebereich der inversen Mappingfunktion müssen also folglich übereinstimmen. Das diskrete Inputbild muss dafür in ein kontinuierliches Bildsignal rekonstruiert werden (*Image Reconstruction*), um es an jeder beliebigen Stelle erneut abtasten zu können. Diesen zweistufigen Prozess, Image Reconstruction - oft auch als *Interpolation* bezeichnet - gefolgt von Sampling, nennt man *Image Resampling*. Er ist in Abbildung 5.3 schematisch dargestellt. Wie der Name schon vermuten lässt, sind Grundlagen aus der Sampling Theorie und des Digitalen Filterings erforderlich, die im nächsten Abschnitt behandelt werden. Darauf aufbauend wird der ideale Image Resampling Prozess noch einmal detailliert beschrieben. Die wichtigsten Interpolationsverfahren werden dann in Abschnitt 5.4 vorgestellt.

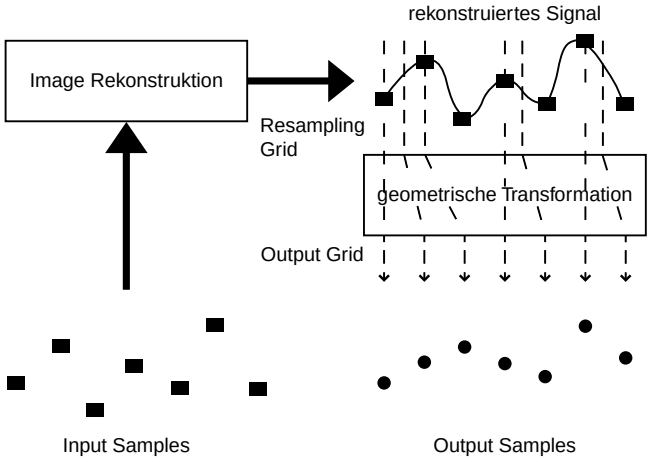


Abbildung 5.3: Image Resampling.

5.2 Grundlagen

5.2.1 Signalrepräsentation

Im Folgenden werden der Einfachheit halber nur eindimensionale Signale betrachtet. Dies entspricht beispielsweise einer einzelnen Bildzeile.
Im *Ortsbereich* werden diese Signale repräsentiert, indem man ihre Amplituden in Abhängigkeit von ihrer räumlichen Position graphisch darstellt. Im *Frequenzbereich* können Signale abgebildet werden, indem man sie in eine Summe

von Sinuskurven unterschiedlicher Frequenzen zerlegt, wobei jede Frequenz durch eine bestimmte Amplitude und Phasenverschiebung charakterisiert ist. Die Umrechnung zwischen Orts- und Frequenzbereich erfolgt durch die *Fourier-Transformation* (Ortsbereich nach Frequenzbereich) bzw. durch die *inverse Fourier-Transformation* (Frequenzbereich nach Ortsbereich).

5.2.2 Sampling

Ausgangspunkt ist ein kontinuierliches Inputsignal. Aufgrund der *Point-Spread-Funktion* des Bilderfassungsgeräts ist das degradierte Outputsignal $g(x)$ bandbegrenzt. Der Frequenzgehalt von $g(x)$ ist durch sein Spektrum $G(f)$ gegeben und lässt sich mithilfe der Fourier-Transformation berechnen:

$$G(f) = \int_{-\infty}^{\infty} g(x) e^{-i2\pi fx}\, dx \tag{5.1}$$

x kennzeichnet dabei die Position im Ortsbereich und f steht für Frequenz. In Abbildung 5.4 ist das Amplitudenspektrum eines Signals dargestellt. Es zeigt eine hohe Konzentration in den niedrigen Frequenzen, die zu den hohen Frequenzen langsam ausläuft. Das Signal ist durch die Frequenzen $-f_{max}$ und f_{max} bandbegrenzt.

Im nächsten Schritt wird das kontinuierliche bandbegrenzte Outputsignal $g(x)$ durch einen idealen Impuls Sampler $s(x)$ digitalsiert, um das diskrete Signal $g_s(x)$ zu erhalten:

$$g_s(x) = g(x)\, s(x) \tag{5.2}$$

Der ideale Impuls Sampler ist für den 1-D Fall gegeben durch

$$s(x) = \sum_{n=-\infty}^{\infty} \delta(x - nT_s) \tag{5.3}$$

wobei δ die Impulsfunktion und T_s die Abtastperiode ist.

Die Fourier-Transformation des diskreten Signals $g_s(x)$ ergibt

$$\begin{aligned} G_s(f) &= G(f) * S(f) && (5.4)\\ &= G(f) * \left[\sum_{n=-\infty}^{n=\infty} f_s \delta(f - nf_s) \right] && (5.5)\\ &= f_s \sum_{n=-\infty}^{\infty} G(f - nf_s) && (5.6) \end{aligned}$$

Die Gleichungen (5.4), (5.5) und (5.6) ergeben sich aus den Eigenschaften der Fourier-Transformation:

- Multiplikation im Ortsbereich entspricht einer Faltung im Frequenzbereich (Gleichung (5.2) und Gleichung (5.4)).
- Die Fourier-Transformation einer Impulsfolge ist wiederum eine Impulsfolge (Gleichung (5.5)).
- Das Spektrum eines Signals, das mit Frequenz f_s ($T_s = 1/f_s$) abgetastet wurde, ergibt das Originalspektrum wiederholt im Frequenzbereich mit Periode f_s (Gleichung (5.6)).

In Abbildung 5.5 ist das Spektrum $G_s(f)$ zu sehen.

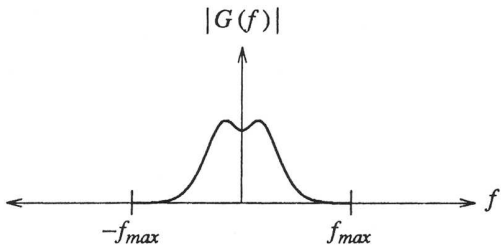


Abbildung 5.4: Spektrum $G(f)$.

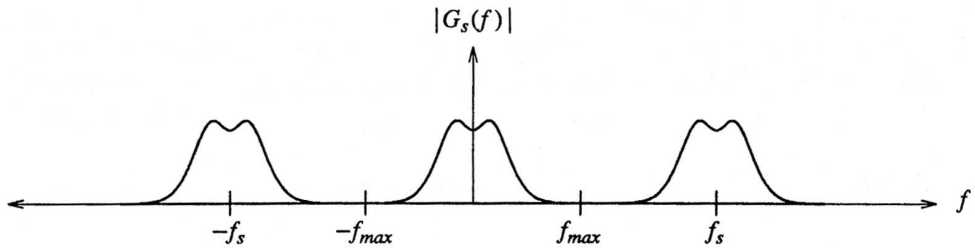


Abbildung 5.5: Spektrum $G_s(f)$.

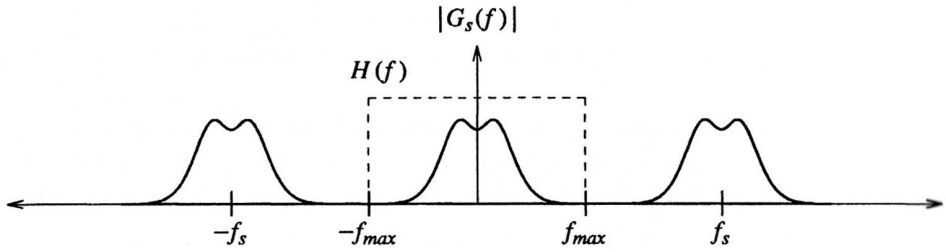


Abbildung 5.6: Ideales Low-Pass Filter $H(f)$ für den Frequenzbereich.

5.2.3 Signalrekonstruktion

Im vorherigen Abschnitt ist deutlich geworden, dass die Samplingoperation das Originalspektrum $G(f)$ intakt lässt - es wird lediglich periodisch im Frequenzbereich mit einem Abstand von f_s wiederholt. $G_s(f)$ kann daher auch als Summe niedriger und hoher Frequenzkomponenten geschrieben werden, wobei die niedrigen Frequenzen $G_s(f)$ entsprechen und die hohen Frequenzkomponenten aus den wiederholten Versionen von $G_s(f)$ bestehen:

$$G_s(f) = G(f) + G_{high}(f) \tag{5.7}$$

Um ein Signal exakt zu rekonstruieren, müssen die hohen Frequenzkomponenten $G_{high}(f)$ unterdrückt werden, um das Originalspektrum $G(f)$ zu isolieren.
Zwei Bedingungen müssen dafür erfüllt sein:

1. Das Signal muss bandbegrenzt sein.
2. Die Abtastfrequenz f_s muss größer sein, als zweimal die Grenzfrequenz f_{max}.

Die minimale Abtastfrequenz ist auch als *Nyquist Rate* bekannt, so dass

$$f_s > f_{Nyquist}, \quad \text{wobei } f_{Nyquist} = 2 \cdot f_{max} \tag{5.8}$$

Ideales Low-Pass-Filter

$G_{high}(f)$ wird unterdrückt, indem $G_s(f)$ mit dem idealen Low-Pass-Filter $H(f)$ multipliziert wird,

$$G(f) = G_s(f)\, H(f) \tag{5.9}$$

wobei

$$H(f) = \begin{cases} 1 & |f| < f_{max} \\ 0 & |f| \geq f_{max} \end{cases} \tag{5.10}$$

In Abbildung 5.6 ist das ideale Low-Pass-Filter $H(f)$ für den Frequenzbereich zu sehen. Es ist die Rechteckfunktion, die alle Frequenzen oberhalb von f_{max} unterdrückt. Diesen Bereich bezeichnet man auch als *stopband*. Das *passband* besteht dagegen aus den Frequenzen, die unterhalb von f_{max} liegen und somit nicht unterdrückt werden.

Sinc-Funktion

Um das ideale Low-Pass-Filter für den Ortsbereich zu erhalten, wird die inverse Fourier-Transformation der Rechteckfunktion $H(f)$ berechnet. Dies ergibt die *sinc*-Funktion (Abbildung 5.7), definiert als

$$sinc(x) = \frac{sin(\pi x)}{\pi x} \tag{5.11}$$

Da eine Multiplikation im Frequenzbereich (vgl. Gleichung (5.9)) einer Faltung im Ortsbereich entspricht, repräsentiert $sinc(x)$ den Faltungskern, der verwendet wird, um jeden Punkt x auf der kontinuierlichen Inputkurve g zu berechnen, wenn nur die Abtastwerte g_s gegeben sind:

$$g(x) = g_s(x) * sinc(x) \tag{5.12}$$

$$= \int_{-\infty}^{\infty} sinc(\lambda) g_s(x - \lambda)\, d\lambda \tag{5.13}$$

In Gleichung (5.13) ist der praktische Nachteil der *sinc*-Funktion zu sehen: Die exakte Berechnung der Punkte des Outputsignals erfordert eine unendliche Anzahl von Nachbarwerten. Dieser Forderung kann in der Praxis jedoch nicht nachgekommen werden, da nur eine begrenzte Anzahl von Abtastwerten zur Verfügung steht.

Das Kürzen der *sinc*-Funktion für näherungsweise Lösungen führt zu sogenanntem *Klingeln*. Abbildung 5.8 zeigt diesen unerwünschten Effekt.

In der Praxis sind eine Vielzahl von Algorithmen entwickelt worden, die lokale Lösungen bieten, d.h., deren Faltungskern nur über eine kleine Nachbarschaft definiert ist. Diese und andere wichtige Interpolationsmethoden werden in Abschnitt 5.4 vorgestellt.

5.3 Ideales Image Resampling

Der ideale Image Resampling Prozess besteht aus den vier Basiselementen *Rekonstruktion*, *Warping*, *Prefiltern* und *Sampling* (vgl. Abbildung 5.9).
Ausgangspunkt ist das diskrete Inputsignal $f(u)$, das über ganzzahlige Werte von u definiert ist:

$$f(u), u \in \mathbb{Z}. \tag{5.14}$$

Faltung mit $r(u)$ ergibt das rekonstruierte kontinuierliche Inputsignal $f_c(u)$

$$f_c(u) = f(u) * r(u) = \sum_{k \in \mathbb{Z}} f(k) r(u - k) \tag{5.15}$$

wobei das ideale Rekonstruktionsfilter $r(u)$ die *sinc*-Funktion (vgl. Abschnitt 5.2.3) ist.
Das kontinuierliche Inputsignal $f_c(u)$ wird dann entsprechend der Mappingfunktion m transformiert. Das Mapping kann sowohl als *Forward Mapping*, wobei $x = m(u)$, als auch als *Backward Mapping*, wobei $u = m^{-1}(x)$, formuliert werden. In diesem Fall ist das Mapping als Backward Mapping definiert:

$$g_c(x) = f_c(m^{-1}(x)) \tag{5.16}$$

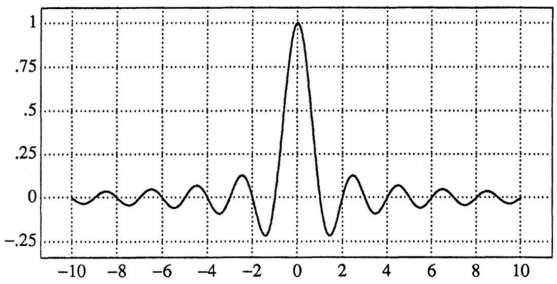


Abbildung 5.7: Die *sinc*-Funktion.

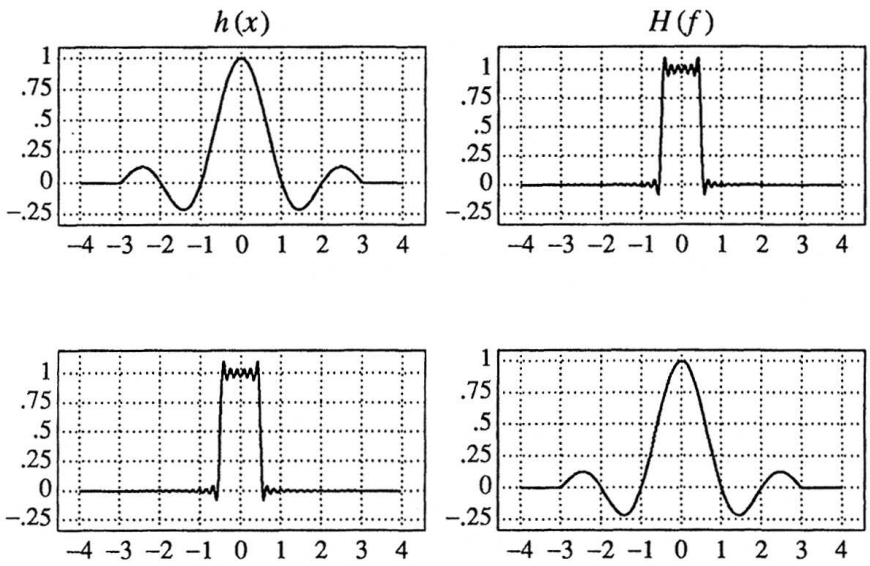


Abbildung 5.8: Das Kürzen in einem Bereich, führt zum *Klingeln* im anderen Bereich.

Das kontinuierliche transformierte Outputsignal wird durch Faltung mit $h(x)$ bandbegrenzt, um der *Nyquist Rate* des Outputs zu genügen. Das bandbegrenzte Ergebnis ist $g_c'(x)$:

$$g_c'(x) = g_c(x) * h(x) = \int g_c(t)h(x-t)\,dt \tag{5.17}$$

$g_c'(x)$ wird im letzten Schritt durch $s(x)$, dem *output sampling grid*, abgetastet, um das diskrete Outputsignal $g(x)$ zu erzeugen:

$$g(x) = g_c'(x)\,s(x) \tag{5.18}$$

Das Rekonstruktionsfilter $r(u)$ und das bandbegrenzende Prefilter $h(x)$ können in einem Filter zusammengefasst werden:

$$\begin{aligned} g(x) &= g_c'(x) \qquad \text{für } x \in \mathbb{Z} && (5.19)\\ &= \int f_c(m^{-1}(t))h(x-t)\,dt && (5.20)\\ &= \int \left[\sum_{k\in\mathbb{Z}} f(k)r(m^{-1}(t)-k)\right] h(x-t)\,dt && (5.21)\\ &= \sum_{k\in\mathbb{Z}} f(k)\,\rho(x,k) && (5.22) \end{aligned}$$

wobei

$$\rho(x,k) = \int r(m^{-1}(t)-k)h(x-t)\,dt \tag{5.23}$$

das *Resampling Filter* ist, das die Gewichtung eines Abtastpunkts des Inputs an Position k für einen Abtastpunkt des Outputs an Position x spezifiziert.

5.4 Interpolation

Wie in den vorherigen Abschnitten bereits mehrfach erwähnt, ist *Interpolation* der Prozess, Werte einer Funktion zu berechnen, deren Positionen zwischen ihren Abtastpunkten liegt. Dies wird erreicht, indem eine kontinuierliche Funktion durch die diskreten Abtastpunkte gelegt wird. Während die Samplingoperation ein unendliches Bandsignal generiert (vgl. Abschnitt 5.2.2), spielt Interpolation eine gegenteilige Rolle: sie reduziert die Bandbreite eines Signals, indem

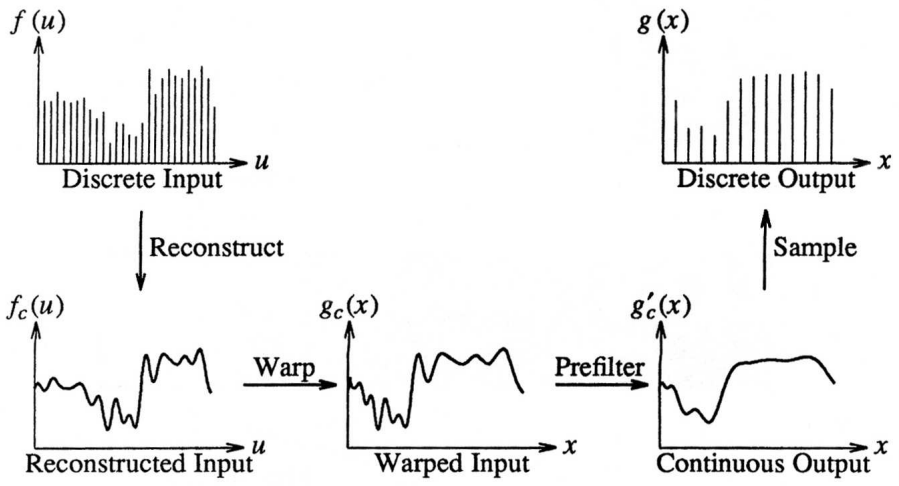


Abbildung 5.9: Ideales Image Resampling

sie einen Low-Pass-Filter auf das diskrete Signal anwendet und dadurch das Signal, das im Sampling Prozess verloren gegangen ist, rekonstruiert. Für gleichmäßig unterteilte Daten, ist Interpolation als Faltungsoperation wie folgt definiert:

$$f(x) = \sum_{k=0}^{K-1} c_k h(x - x_k) \tag{5.24}$$

Dabei ist h der Interpolationskern, der durch die Koeffizienten c_k gewichtet wird und auf K Abtastwerte x_k angewandt wird. h ist in der Praxis meistens ein symmetrisches Filter, d.h. $h(-x) = h(x)$. Weiterhin sind die Koeffizienten c_k in den meisten Fällen die Abtastwerte selbst. Die Berechnung eines interpolierten Punkts ist in Abbildung 5.10 zu sehen.

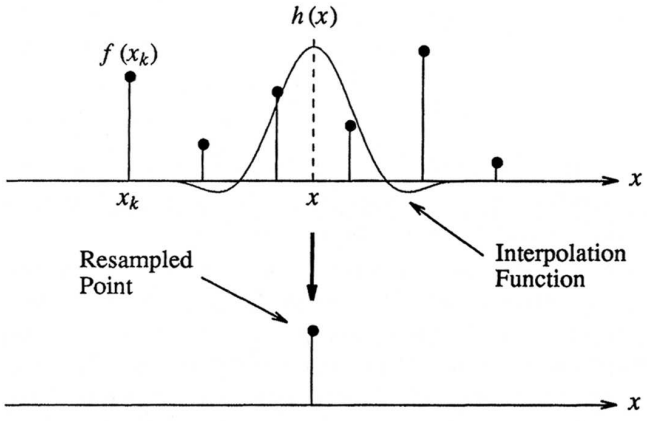


Abbildung 5.10: Interpolation.

In der Praxis wird die Interpolationsoperation oftmals nicht als Faltung implementiert. Stattdessen ist es einfacher, das entsprechende Interpolationspolynom direkt für den zu interpolierenden Punkt zu berechnen. Trotzdem ist es sinnvoll, den Interpolationskern und dessen Spektrum genauer zu betrachten. Dabei wird v.a. sein Verhalten im passband und stopband untersucht, um somit im Vergleich zum idealen Low-Pass-Filter eine Aussage über die Genauigkeit des Interpolationsverfahrens machen zu können. In Abschnitt 5.2.3 wurde das ideale Low-Pass-Filter $H(f)$, die Rechteckfunktion, bereits beschrieben. Es unterdrückt das stopband vollständig und lässt das passband intakt.

Im Folgenden werden die wichtigsten Interpolationsverfahren für den eindimensionalen Fall vorgestellt.

5.4.1 Nearest Neighbor

Der einfachste Interpolationsalgorithmus vom Grad 0 ist der *Nearest Neighbor*-Algorithmus. Jedem interpolierten Outputpixel wird dabei der Wert des am nächsten gelegenen Abtastpunkts im Inputbild zugewiesen.
Das Interpolationspolynom ist gegeben durch

$$f(x) = f(x_k) \qquad \text{wobei} \quad \frac{x_{k-1} + x_k}{2} < x \leq \frac{x_k + x_{k+1}}{2} \tag{5.25}$$

Im Ortsbereich wird das Bild mit einer 1-Pixel breiten Rechteckfunktion gefaltet. Der Interpolationskern ist daher definiert als

$$h(x) = \begin{cases} 1 & 0 \leq |x| < .5 \\ 0 & .5 \leq |x| \end{cases} \tag{5.26}$$

In Abbildung 5.11 ist der Interpolationskern und dessen Fourier-Transformation zu sehen.
Das Spektrum $H(f)$ zeigt, dass das stopband nicht vollständig unterdrückt wird - dafür aber Frequenzen im passband, die eigentlich intakt bleiben sollten. Demzufolge ist der Nearest Neighbor-Algorithmus ein relativ schlechtes Interpolationsverfahren.

5.4.2 Lineare Interpolation

Die *Lineare Interpolation* zieht eine Linie durch zwei aufeinanderfolgende Punkte des Inputsignals. Gegeben ein Intervall $[x_0, x_1]$ und Funktionswerte f_0 und f_1 der Endpunkte, so ist das Interpolationspolynom definiert als

$$f(x) = f_0 + \left(\frac{x - x_0}{x_1 - x_0}\right)(f_1 - f_0) \tag{5.27}$$

Im Ortsbereich entspricht die Lineare Interpolation einer Faltung mit folgendem Interpolationskern:

$$h(x) = \begin{cases} 1 - |x| & 0 \leq |x| < 1 \\ 0 & 1 \leq |x| \end{cases} \tag{5.28}$$

h wird auch als *Dreiecksfunktion* bezeichnet und ist zusammen mit seinem Spektrum in Abbildung 5.12 zu sehen. Verglichen mit der Nearest Neighbor-Interpolationsfunktion liefert die Dreiecksfunktion bessere Ergebnisse. Die Abschwächungen im passband führen zur Bildglättung.
Die Lineare Interpolation ist ein weit verbreitetes Verfahren, da es einigermaßen gute Ergebnisse bei einem geringen Rechenaufwand liefert.

5.4.3 Cubic Convolution

Cubic Convolution ist ein Interpolationsalgorithmus dritten Grades und ist als effiziente Näherung zu der theoretisch optimalen *sinc*-Interpolationsfunktion entwickelt worden.
Der Interpolationskern besteht aus stückweisen kubischen Polynomen, die über die einzelnen Teilintervalle $[-2, -1]$, $[-1, 0]$, $[0, 1]$ und $[1, 2]$ definiert sind. Außerhalb des Intervalls $[-2, 2]$ ist der Interpolationskern = 0. Als Ergebnis ist

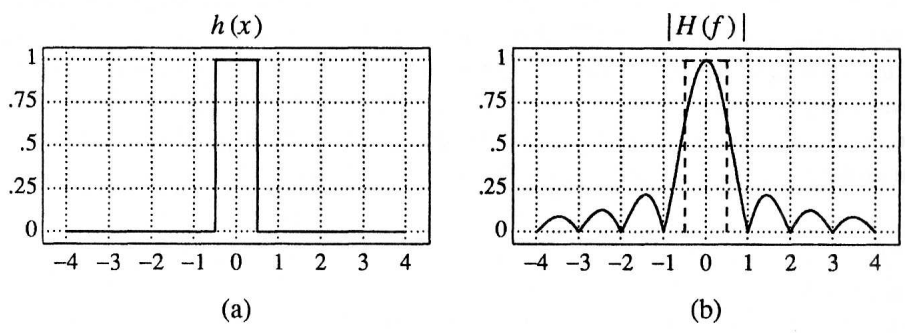


Abbildung 5.11: Nearest Neighbor: Interpolationskern (a) und dessen Fourier-Transformation (b).

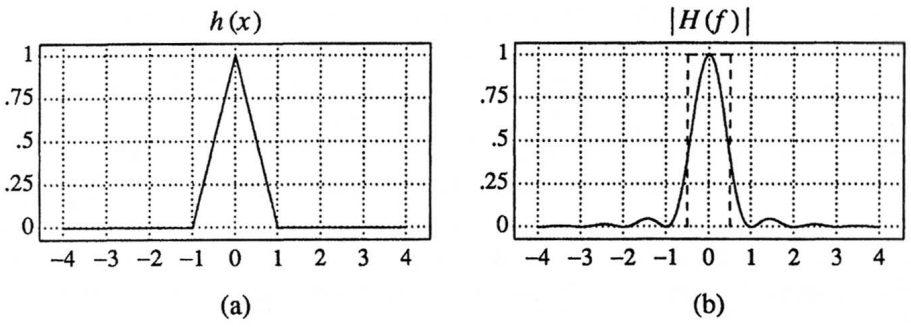


Abbildung 5.12: Lineare Interpolation: Interpolationskern (a) und dessen Fourier-Transformation (b).

jeder interpolierte Punkt eine gewichtete Summe von vier aufeinanderfolgenden Punkten des Inputsignals. Der symmetrische Interpolationskern hat die Form

$$h(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & 0 \leq |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & 1 \leq |x| < 2 \\ 0 & 2 \leq |x| \end{cases} \tag{5.29}$$

wobei a ein vom Benutzer frei wählbarer Parameter ist. Wählt man beispielsweise für a Werte zwischen -3 und 0, dann ähnelt h der *sinc*-Funktion. Ein Wert von $a = -1$ führt zur Bildschärfung. In Abbildung 5.13 ist der Interpolationskern h mit $a = -0.5$ dargestellt.
Allgemein können durch Cubic Convolution Werte entstehen, die außerhalb des Wertebereichs des Inputsignals liegen. Daher ist bei diesem Verfahren ein abschließendes *Clipping* erforderlich.

5.4.4 Cubic Spline

Eine *Cubic Spline* ist ein stückweise kontinuierliches Polynom dritten Grades. Sind n Punkte (x_k, y_k) für $0 \leq k < n$ gegeben, dann besteht eine Cubic Spline aus $n-1$ kubischen Polynomen, die durch die gegebenen Punkte gehen (vgl. Abbildung 5.14). Die gegebenen n Punkte nennt man auch *Kontrollpunkte*. Das k te Polynomstück f_k geht durch zwei aufeinanderfolgende Punkte im festen Intervall $[x_k, x_{k+1}]$. Das Interpolationspolynom f_k ist gegeben durch

$$f_k(x) = a_3(x - x_k)^3 + a_2(x - x_k)^2 + a_1(x - x_k) + a_0 \tag{5.30}$$

Die stückweisen Interpolationspolynome wurden entwickelt, ohne einen entsprechenden Interpolationskern zu definieren. In der Praxis werden *B-Splines* als Interpolationskern verwendet. Näheres hierzu ist in [2] zu finden.

5.4.5 Windowed Sinc Funktion

Aus der Sampling Theorie (vgl. Abschnitt 5.2.3) ist bekannt, dass die *sinc*-Funktion der ideale Interpolationskern ist. Obwohl dieses Filter exakt ist, ist es aufgrund der Tatsache, dass es über eine unendliche Summe definiert ist (Glei-

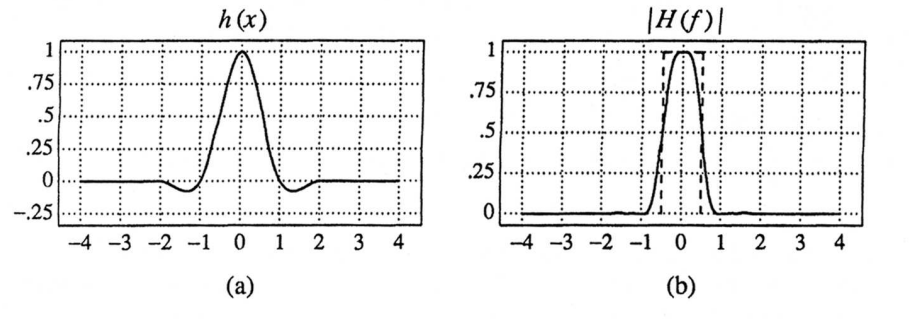


Abbildung 5.13: Cubic Convolution ($a = -0.5$): Interpolationskern (a) und dessen Fourier-Transformation (b).

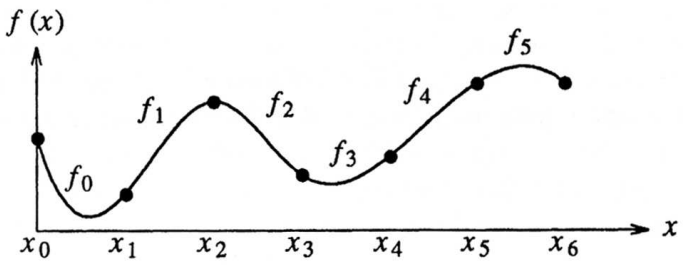


Abbildung 5.14: Cubic Spline, die aus 6 stückweisen kubischen Polynomen besteht.

chung (5.7)) nicht praktikabel. In der Praxis wird daher eine gekürzte und damit endliche Version der *sinc*-Funktion verwendet. Das Kürzen bzw. Abschneiden eines Signals entspricht einer Multiplikation mit der Rechteckfunktion $Rect(x)$, definiert als

$$Rect(x) = \begin{cases} 1 & 0 \leq |x| < .5 \\ 0 & .5 \leq |x| \end{cases} \tag{5.31}$$

Da die Multiplikation im Ortsbereich einer Faltung im Frequenzbereich entspricht, wird das Spektrum des Signals mit der *sinc*-Funktion, dem Transformationspaar der Rechteckfunktion, gefaltet. Es ist bereits gezeigt worden, dass dies zu sogenanntem *Klingeln* führt (Abbildung 5.8).

Die obige $Rect$-Funktion dient als *Fenster* oder *Maske* und legt die Gewichtung des Inputsignals fest. $Rect(6x)$ ist beispielsweise ein Fenster, das sich über jeweils 3 Punkte rechts und links des Zentrums erstreckt. Die *windowed sinc*-Funktion $h(x)$ erhält man, indem man die $sinc$-Funktion mit der Fensterfunktion $Rect(6x)$ multipliziert:

$$h(x) = sinc(x)Rect(6x) \tag{5.32}$$

Das Spektrum der *windowed sinc*-Funktion ist in Abbildung 5.15 dargestellt. Um das hier noch sehr deutliche Klingeln

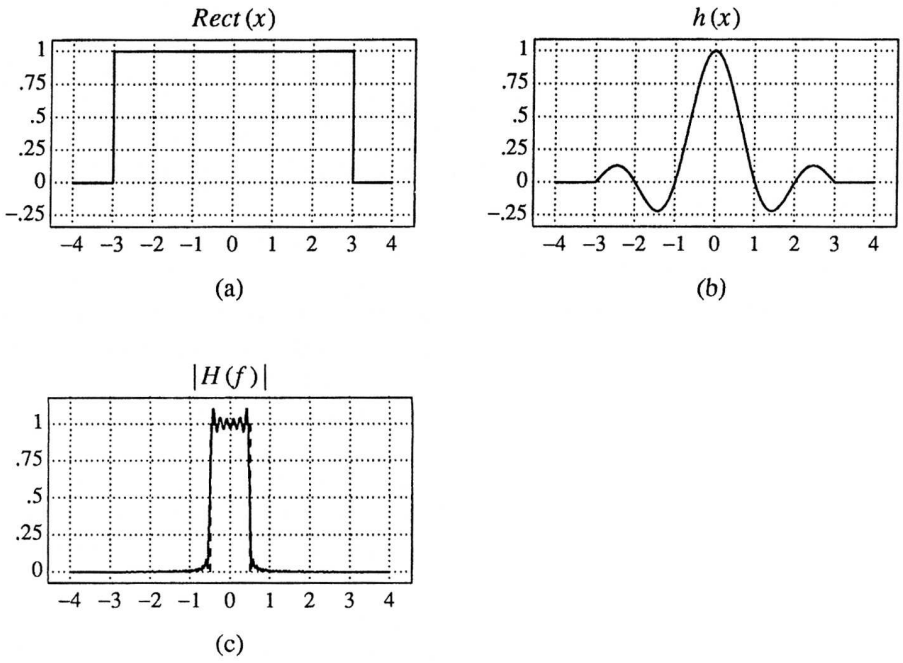


Abbildung 5.15: *Rect(6x)*: Fenster(a), windowed $sinc$-Funktion (b) und dessen Spektrum (c).

abzuschwächen, wurden verschiedene Fensterfunktionen entwickelt, die weichere Übergänge als die Rechteckfunktion haben. Dazu gehören das *Hann*- (Abbildung 5.16), *Hamming*-, *Blackman*-, *Kaiser*-, *Lanczos*- (Abbildung 5.17) und *Gauss*-Fenster.

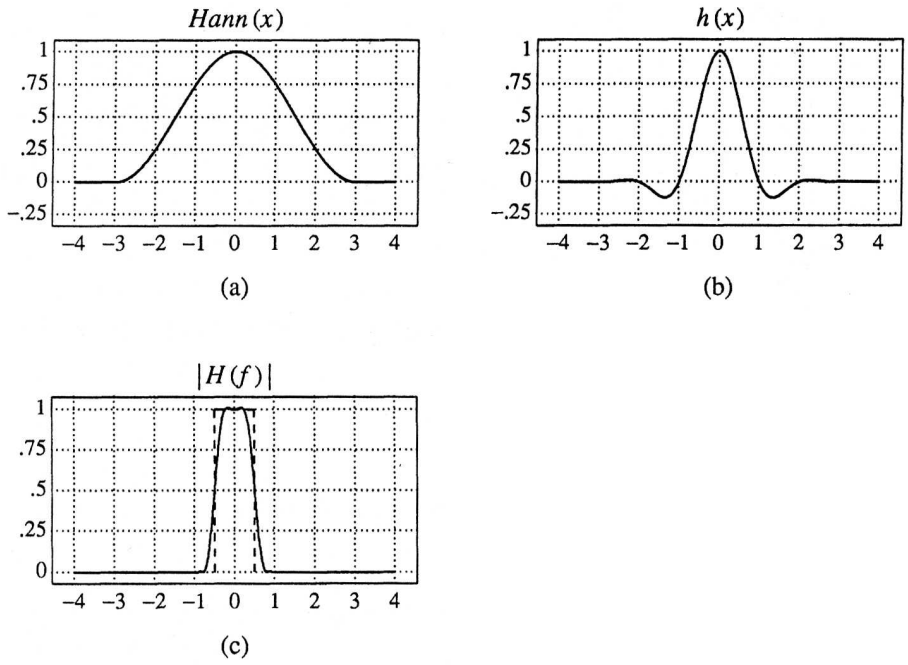


Abbildung 5.16: Hann: Fenster (a), windowed $sinc$ (b) und dessen Spektrum (c)

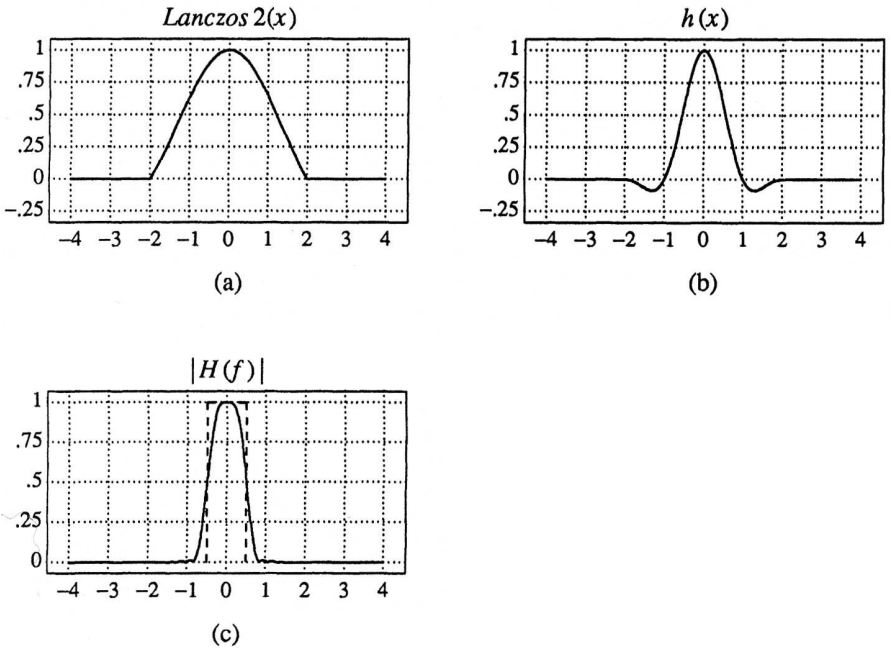


Abbildung 5.17: Lanczos2: Fenster (a), windowed $sinc$ (b), Spektrum (c)

5.5 Test der Interpolationsverfahren

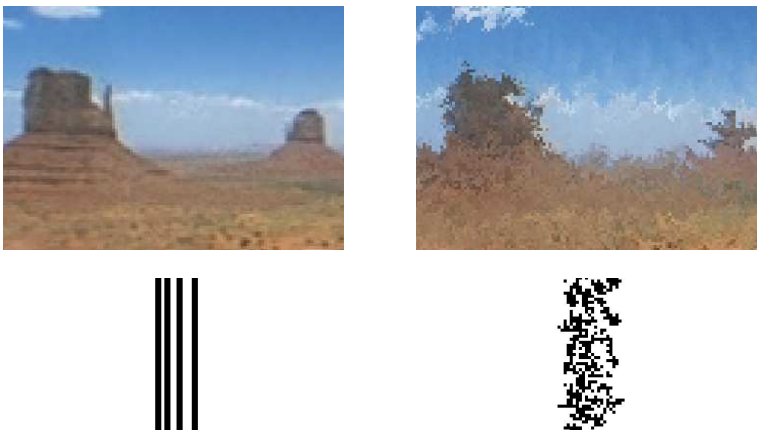
Helmut Dersch hat in [1] verschiedene Interpolationsmethoden getestet, um die Qualität seines eigenen Programms, dem Photoshop/Gimp Plug-In *Panorama Tools*, zu untersuchen. Bei dem Test handelt es sich um eine 36-fache Rotation zweier Testbilder (siehe Abbildung 5.18), um jeweils 5°. Dies entspricht 36 Resamplingschritten. Die Ergebnisse dieser Tests sind im nächsten Abschnitt zu sehen.

Abbildung 5.18: Testbilder.

5.5.1 Testergebnisse

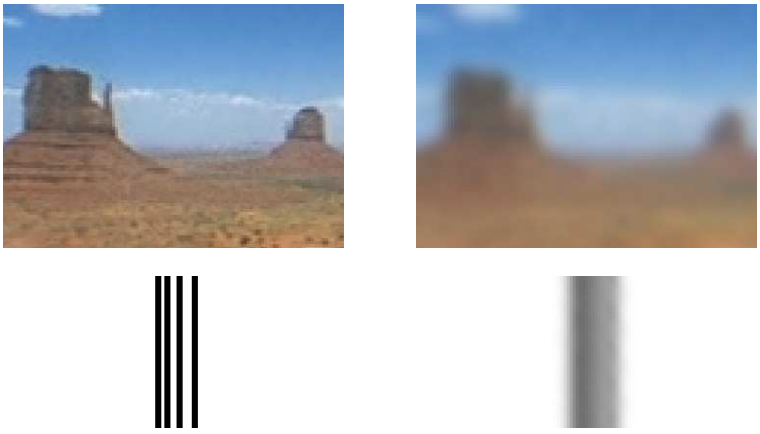
Picture Publisher (Windows 95)

Original



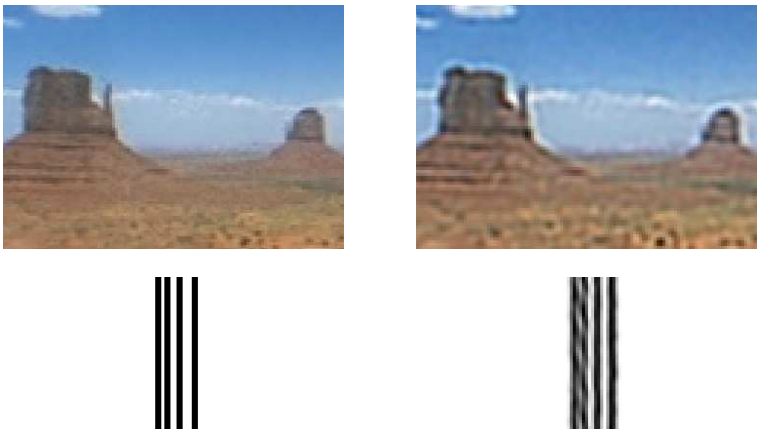
Photoshop: Bilineare Interpolation

Original



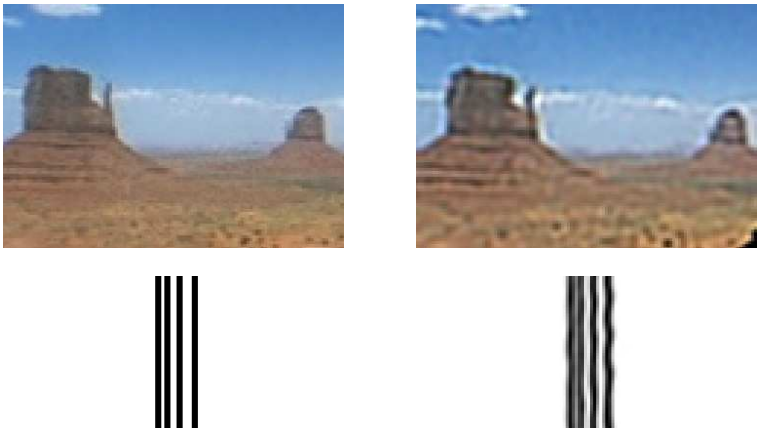
Photoshop: Bikubische Interpolation

Original



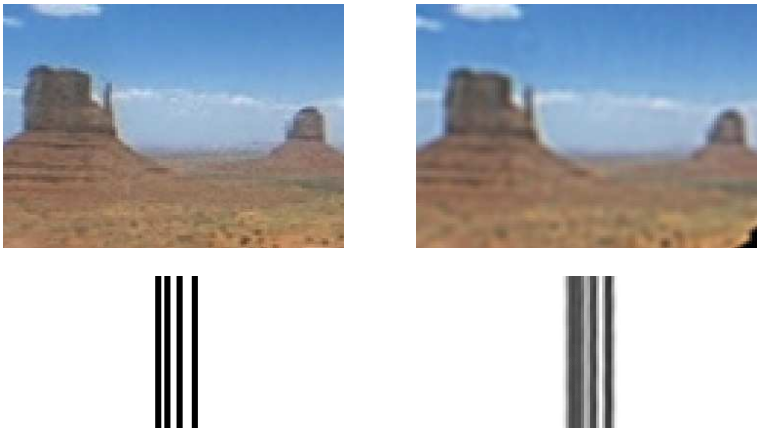
Panorama Tools: Cubic Interpolator *Poly3*

Original



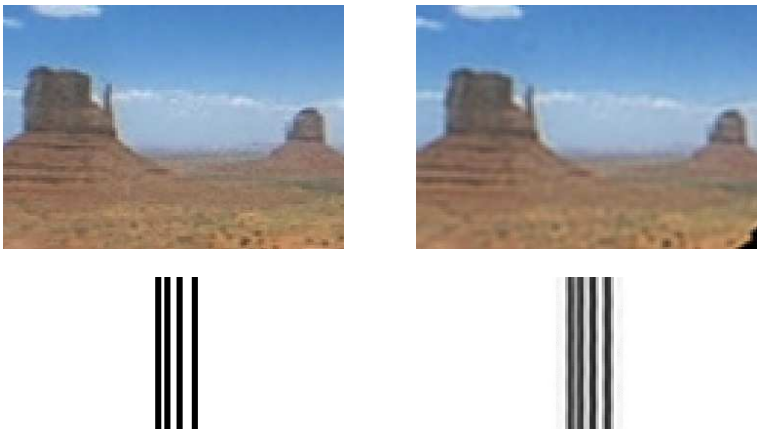
Panorama Tools: Spline 16 Pixel (4x4)

Original



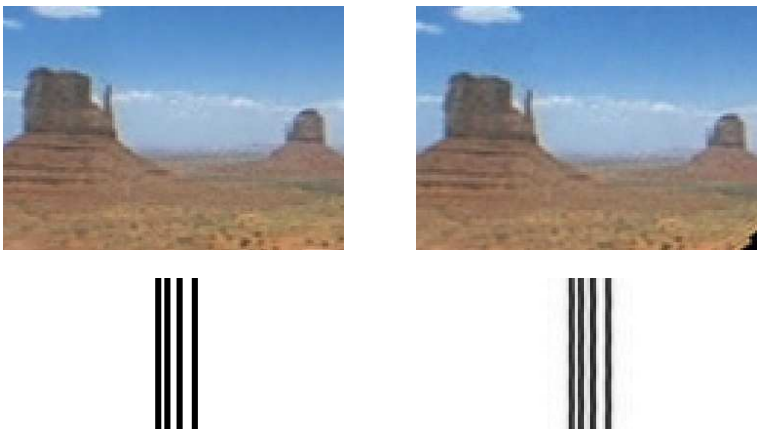
Panorama Tools: Spline 36 Pixel (6x6)

Original



Panorama Tools: Sinc 256 Pixel (16x16)

Original



5.5.2 Bewertung

Der Nearest Neighbor-Algorithmus, der offensichtlich im Picture Publisher implementiert ist, liefert die schlechteste Bildqualität. Es ist zu sehen, dass bei diesem Verfahren keine neuen Intensitätswerte hinzukommen, daher wird es auch manchmal als *Pixelwiederholung* bezeichnet. Die Bilineare Interpolation führt zu signifikanten Verlusten, ist aber oftmals akzeptabel, wenn nur ein oder zwei Resamplingschritte durchgeführt werden. Bei *Poly 3*, der bikubischen Interpolation von Panorama Tools, ist der frei wählbare Paramter a auf -0.75 gesetzt und ähnelt damit der bikubischen Interpolation von Photoshop. Beide haben einen deutlichen Schärfungseffekt, der zu Artefakten führt - hier deutlich zu sehen im Linienbild. *Spline 16* ist eine Cubic Spline und verwendet 4x4 = 16 Pixel, um das Outputbild zu generieren. Im Gegensatz zu den bikubischen Interpolationsmethoden wird das Bild hierbei geglättet. *Spline 36* ist ebenfalls eine Cubic Spline, verwendet aber 6x6 = 36 Pixel. Dies führt zu einer besseren Auflösung.
Die besten Ergebnisse liefert - wie zu erwarten - die *windowed sinc*-Funktion *Sinc 256* von Panorama Tools. Als Fensterfunktion wird hier das Lanczos-Window verwendet (vgl. Abbildung 5.17). Um ein Inputpixel zu interpolieren, werden 16x16 = 256 Pixel benötigt. Da die *sinc*-Funktion bereits die theoretisch optimale Interpolationsfunktion ist, kann das Ergebnis nur noch durch Wahl eines größeren Fensters verbessert werden.

Literaturverzeichnis

[1] H. Dersch. *Testing Interpolator Quality*. Technical University Furtwangen, 1999. http://www.path.unimelb.edu.au/˜ dersch/interpolator/interpolator.html.

[2] G. Wolberg. *Digital Image Warping*. IEEE CS Press, 1990.

Vortrag 6

Punktverfolgung

Christian Dietz

24. März 2004

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
snoopy@uni-koblenz.de
http://www.uni-koblenz.de/~snoopy

6.1 Einleitung

6.1.1 Was ist Punktverfolgung?

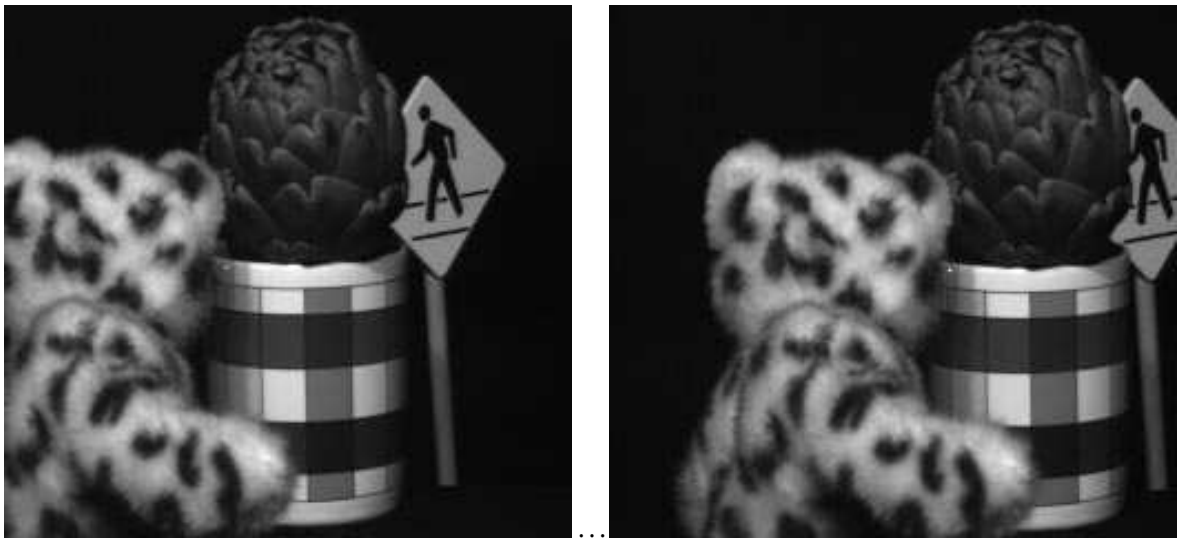
Mit Punktverfolgung bezeichnet man ein Verfahren, um in einer Bildfolge markante Stellen, sogenannte Features, zu finden und diese zu verfolgen. Man erhält dadurch eine Reihe von Punktkorrespondenzen.

6.1.2 Wofür braucht man Punktverfolgung?

Die Punktkorrespondenzen werden für das Rechnersehen benötigt, zum Beispiel für Verfahren, die mit Epipolargeometrie arbeiten. Im nächsten Abschnitt wird ein kleines Beispiel gezeigt.

6.1.3 Einführungsbeispiel

Hier ein kleines Beispiel, um zu veranschaulichen, worum es geht:



Zwischen dem linken und dem rechten Bild liegen 100 weitere Bilder (engl.: frames). Als erstes müssen Stellen im Bild gefunden werden, die sich gut verfolgen lassen. Diese werden dann von Bild zu Bild „getracked“, indem die Verschiebung von z.B. dem zweiten Bild zu dem ersten Bild berechnet wird.

Über viele Bilder hinweg können dabei Probleme auftreten, wie z.B. verschwindende Bildteile. Im linken Beispielbild etwa ist die rechte Ecke des Schildes im ersten Frame zu sehen, im einhundertsten jedoch nicht mehr. Daher muss der Trackingvorgang überwacht werden, was mittels Monitoring geschieht (Siehe auch Abbildung 6.1).

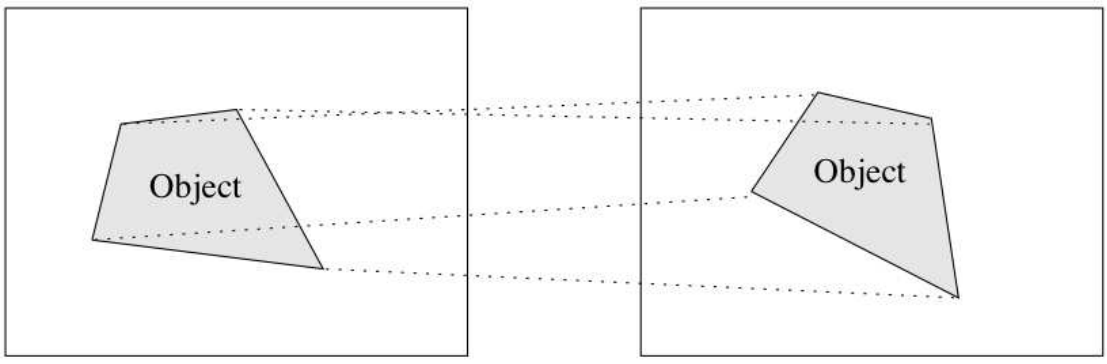


Abbildung 6.1: Veränderung eines Objekts über mehrere Frames hinweg

6.1.4 Übersicht

Dieses Kapitel ist folgendermaßen eingeteilt:

- Als erstes gibt es eine Einführung in Bildsequenzen und die Grundlagen der Punktverfolgung (6.2),
- dann wird gezeigt, mittels welcher Formeln die Bildverschiebung berechnet wird (6.3),
- anschließend folgt ein Abschnitt zum Thema Feature Auswahl (6.4),
- dann geht es um Monitoring, also darum, wie der Tracking Vorgang überwacht werden kann (6.5),
- und schließlich gibt es noch den obligatorischen Ausblick (6.6).

6.2 Grundlagen der Punktverfolgung

6.2.1 Einführung in Bildsequenzen

$I(x, y, t)$ steht für eine Bildsequenz: x und y sind räumliche Variablen, t ist die zeitliche Variable. Zeitlich nahe beieinander liegende Bilder sind sich für gewöhnlich sehr ähnlich. Die Funktion I hat die folgende Eigenschaft:

$$\underbrace{I(x, y, t + \tau)}_{\text{späteres Bild}} = \underbrace{I(x - \xi, y - \eta, t)}_{\text{früheres Bild}} \tag{6.1}$$

Zu deutsch: ein späteres Bild, aufgenommen zur Zeit $t + \tau$, entsteht aus einem früheren Bild, aufgenommen zur Zeit t, wenn man jeden Punkt im früheren Bild um einen geeigneten Betrag verschiebt:

$$\boldsymbol{d} = (\xi, \eta) \tag{6.2}$$

ist also die Verschiebung des Punktes $\boldsymbol{x} = (x, y)$ im Zeitraum zwischen t und $t + \tau$.

6.2.2 Grundlagen

- Ein einzelner Pixel könnte nur verfolgt werden, wenn sich seine Helligkeit sehr von allen seinen Nachbarn unterscheidet. Außerdem kann sich der Wert eines Pixel leicht durch Rauschen verändern und der Pixel kann mit angrenzenden Pixeln verwechselt werden.

  Daher verfolgt man keine einzelnen Pixel, sondern Fenster, die ausreichend Oberflächenstruktur enthalten. Punktverfolgung müsste also eigentlich Fensterverfolgung heißen; Im Englischen wird dies deutlicher, dort heißt es „feature tracking“.

- Man nimmt eher kleinere Fenster, z.B. 15x15. Diese sind zwar rauschempfindlicher, dafür ist es unwahrscheinlicher, dass sie auf Oberflächenfehler stoßen oder von Verzerrungen beeinflusst werden, die durch Änderungen des Blickwinkels erzeugt werden.

- Damit das gesamte Punktverfolgungsverfahren funktioniert, muss die Verschiebung ausreichend klein sein. Mehr dazu im Unterabschnitt 6.3.1.

- Die Verschiebung wird innerhalb des gewählten Fensters als konstant angenommen. Aufgrund dieser Annahme kann die Bildverschiebung zunächst nur annäherungsweise berechnet werden. Anschließend wird dann eine subpixel-genaue Berechnung der Bildverschiebung durch Interpolation durchgeführt (siehe Unterabschnitt 6.3.4).

6.2.3 Vereinfachung der Bildsequenz in ein lokales Bildmodell

Zunächst redefiniert man $J(\boldsymbol{x}) = I(x, y, t + \tau)$ und $I(\boldsymbol{x} - \boldsymbol{d}) = I(x - \xi, y - \eta, t)$, wobei die Zeitvariable der Übersichtlichkeit halber weggelassen wird. Nun lautet unser lokales Bildmodell:

$$\underbrace{J(\boldsymbol{x})}_{späteres\ Bild} = \underbrace{I(\boldsymbol{x} - \boldsymbol{d})}_{früheres\ Bild} \tag{6.3}$$

6.2.4 Das Registrierungsproblem

Unterschiede zwischen zwei aufeinander folgenden Fenstern, die nicht durch eine Verschiebung erklärt werden können, werden als Fehler aufgefasst. Das Registrierungsproblem besteht darin, den Verschiebungsvektor $\boldsymbol{d}$ so zu wählen, dass dieser Restfehler ϵ möglichst klein ist:

$$\epsilon = \int_W [I(\boldsymbol{x} - \boldsymbol{d}) - J(\boldsymbol{x})]^2 \, w \, dx \tag{6.4}$$

Dieses klassische Fehlermaß leitet sich aus der Formel zur annäherungsweisen Berechnung der Varianz ab [7]:

$$V[X] = \frac{1}{N} \sum_{i=1}^{N} (x_i - E[X])^2$$

Der Hauptunterschied besteht darin, dass in Formel 6.4 statt einer Summe ein Integral über das Fenster W verwendet wird; In der Praxis wird dennoch mit diskreten Pixelabständen gerechnet, die Integrale spiegeln also lediglich ein mathematisches Ideal wider.

Der Unterschied zwischen dem früheren und dem späteren Bild (Die Differenz $I(\boldsymbol{x} - \boldsymbol{d}) - J(\boldsymbol{x})$) wird also quadriert und anschließend über $\boldsymbol{d}$ integriert. Dieses doppelte Integral[1] über das Fenster W kann man sich als zwei for-Schleifen vorstellen, die über alle Pixel im Fenster W laufen und dabei alle quadrierten Differenzen addieren.

w ist eine Gewichtungsfunktion, die normalerweise auf 1 gesetzt wird. Alternativ könnte w zum Beispiel eine Gauss-ähnliche Funktion sein, um den mittleren Bereich des Fensters zu betonen.

6.3 Berechnung der Bildverschiebung (Tracking)

6.3.1 Einleitung

Wenn die Verschiebung $\boldsymbol{d}$ bedeutend kleiner ist als die Fenstergröße, dann ist die Linearisierungsmethode [1] der effizienteste Weg um fortzufahren. In den folgenden Berechnungen und Umformungen wird vorgestellt, wie Tomasi und Kanade [5] mit einer neuen Version dieser Methode folgendes zeigen:

Wenn die Verschiebung zwischen zwei Bildern klein genug ist, kann der Verschiebungsvektor annäherungsweise als die Lösung eines 2x2-Gleichungssystems geschrieben werden.

6.3.2 Intensitätsfunktion annähern

Mit Hilfe der Taylor-Reihen kann die Intensitätsfunktion angenähert werden. Verkürzt man anschließend noch auf den linearen Anteil, erhält man:

$$I(\boldsymbol{x} - \boldsymbol{d}) = I(\boldsymbol{x}) - \boldsymbol{g} \cdot \boldsymbol{d} \tag{6.5}$$

[1] $\int_W$ ist eine Kurzform von $\int \int_W$

Abbildung 6.2: Dank geht an Brook Taylor

Der Gradient

$\boldsymbol{g}(\boldsymbol{x}) = \nabla I(\boldsymbol{x})$ steht hier für den Gradienten, hier kurz geschrieben als $\boldsymbol{g}$. Zur Erklärung des Gradienten, wie er in der Bildverarbeitung verwendet wird, eine kleine Veranschaulichung [7]:

Interpretiert man ein Bild als eine Funktion $z(x, y)$, die jedem Pixel einen Grauwert an dieser Stelle im Bild zuordnet, dann ist der Gradient von z an einer Stelle (x, y) ein Vektor, der in die Richtung des steilsten Anstieges zeigt, und dessen Länge ein Maß für die Steigung ist.

Kurz gesagt, der Gradient gibt die Richtung und Stärke der „Steigung“ zwischen Grauwerten an (siehe Abbildung 6.3).

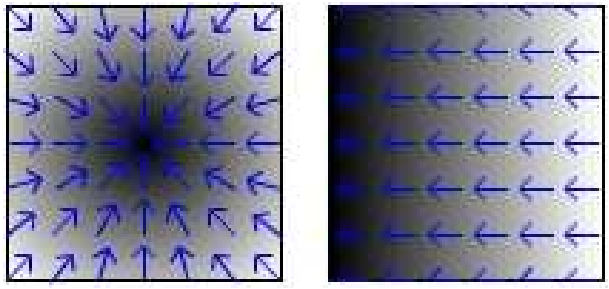


Abbildung 6.3: Der Gradient

Man unterscheidet außerdem zwischen horizontalem und vertikalem Gradienten (siehe Abbildung 6.4).

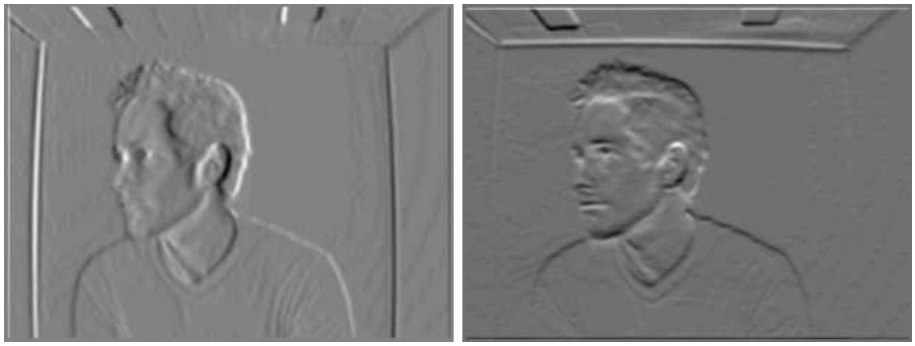


Abbildung 6.4: Horizontaler (links) und vertikaler (rechts) Gradient

6.3.3 Bildverschiebung berechnen

Der Restfehler aus Formel 6.4 kann jetzt so geschrieben werden:

$$\epsilon = \int_W [I(\boldsymbol{x}) - \boldsymbol{g} \cdot \boldsymbol{d} - J(\boldsymbol{x})]^2 \; w \; dx$$
$$\Rightarrow \qquad \epsilon = \int_W (h - \boldsymbol{g} \cdot \boldsymbol{d})^2 \; w \; dx \tag{6.6}$$

wobei

$$h = I(\boldsymbol{x}) - J(\boldsymbol{x}) \tag{6.7}$$

Dieser Restfehler (Formel 6.6) ist jetzt eine quadratische Funktion des Verschiebungsvektors $\boldsymbol{d}$. Daher kann die Minimierung in geschlossener Form erfolgen; es ist also möglich, $\int_W (h - \boldsymbol{g} \cdot \boldsymbol{d})^2 \; w \; dx$ nach $\boldsymbol{d}$ abzuleiten und anschließend das Ergebnis auf 0 zu setzen:

$$\int_W (h - \boldsymbol{g} \cdot \boldsymbol{d}) \boldsymbol{g} \; w \; dA = 0$$

Statt dx schreibt man ab jetzt dA; das sollte nicht weiter verwirren, es handelt sich lediglich um eine Konvention. Durch „Auseinanderziehen“ des Integrals erhält man:

$$\int_W h\boldsymbol{g} \; w \; dA - \int_W (\boldsymbol{g} \cdot \boldsymbol{d}) \boldsymbol{g} \; w \; dA = 0$$

Da $(\boldsymbol{g} \cdot \boldsymbol{d})\boldsymbol{g} = (\boldsymbol{g}\boldsymbol{g}^T)\boldsymbol{d}$ gilt und man $\boldsymbol{d}$ innerhalb von W als konstant annimmt, erhält man

$$\underbrace{\left(\int_W \boldsymbol{g}\boldsymbol{g}^T \; w \; dA \right)}_{\boldsymbol{G}} \boldsymbol{d} = \underbrace{\int_W h\boldsymbol{g} \; w \; dA}_{\boldsymbol{e}}$$

Kurz:

$$\boldsymbol{G}\boldsymbol{d} = \boldsymbol{e} \tag{6.8}$$

Dies ist der grundlegende Schritt der Tracking Prozedur. Zusammengefasst waren folgende Schritte nötig, um zu dieser Formel zu gelangen:

- Für jedes Paar aufeinander folgender Bilder kann die Matrix $\boldsymbol{G}$ anhand eines Bildes berechet werden, indem der Gradient abgeschätzt wird. Dies wurde in Abschnitt 6.3.2 gemacht.
- Der Vektor $\boldsymbol{e}$ wurde aus der Differenz der beiden Bilder (Formel 6.7) und aus dem Gradienten $\boldsymbol{g}$ berechnet.
- Hat man $\boldsymbol{G}$ und $\boldsymbol{e}$, erhält man mittels der Formel 6.8 die Verschiebung $\boldsymbol{d}$ als Annäherung.

6.3.4 Subpixel-genaue Berechnung durch Interpolation

Nun wird das Bild $I(\boldsymbol{x} - \boldsymbol{d})$ bilinear interpoliert, neu abgetastet und die Verschiebung $\boldsymbol{d}$ neu berechnet; dies wird solange wiederholt, bis die Gesamtverschiebung bis auf 1/100tel Pixel genau berechnet ist; normalerweise reichen dafür weniger als fünf Wiederholungen.

6.4 Feature Auswahl

6.4.1 Einleitung

In diesem Abschnitt geht es darum, wie gute Features ausgewählt werden. Zunächst einmal ist zu beachten, dass nicht alle Teile eines Bildes Bewegungsinformation enthalten. Beispiele für solche Bildbereiche sind einfarbige Flächen oder grade Kanten (siehe Abschnitt 6.4.4).

Abbildung 6.5: Beispielbild

6.4.2 Intuitiver Ansatz

Man wählt Bildbereiche aus, die ausreichend stark texturiert sind, also zum Beispiel Ecken oder Bereiche, die hohe räumliche Frequenzen enthalten. Es gibt zahllose weitere solcher „Interessenoperatoren“; sie alle haben den Nachteil, dass sie unabhängig von der Trackingmethode gewählt sind – daher ist nicht garantiert, dass die gewählten Features die besten sind, um beim Tracking gute Ergebnisse zu erzielen.

6.4.3 Ansatz von Tomasi und Kanade

Der Ansatz von Tomasi und Kanade [5] hat den Vorteil, dass er auf der Tracking-Methode basiert:

Ein Feature ist gut, wenn es gut verfolgt werden kann.

Dieser Ansatz basiert auf der Tracking-Methode (Abschnitt 6.3), weil er zwei Forderungen an die Matrix $\boldsymbol{G}$ aus Formel 6.8 stellt – sie muss

- sich deutlich vom Rauschen des Bildes absetzen und
- gut konditioniert sein.

Die Matrix $\boldsymbol{G}$ setzt sich deutlich vom Rauschen des Bildes ab, wenn beide Eigenwerte von $\boldsymbol{G}$ groß genug sind. Sie ist gut konditioniert, wenn beide Eigenwerte nicht zu weit auseinander liegen. Zwei kleine Eigenwerte würden bedeuten, dass innerhalb des Fensters ein ziemlich gleichmäßig gefärbtes Gebiet vorliegt. Ein großer und ein kleiner Eigenwert würden bedeuten, dass ein Gebiet, welches unter dem Apertur Problem leidet, vorliegt.

6.4.4 Das Apertur Problem [engl.: aperture problem]

„Aperture problem“ könnte mit „Blendenproblem“ oder „Fensteröffnungsproblem“ übersetzt werden. Es tritt zum Beispiel bei Bildbereichen auf, die nichts weiter als eine grade Kante enthalten, wie zum Beispiel der Pfosten des Schildes oder die Kanten im Muster der Tasse in Bild 6.5. Wenn man sich nur einen solchen Bildausschnitt (wie durch eine Blende) anschaut, kann man über mehrere Bilder hinweg höchstens eine Komponente der Verschiebung $\boldsymbol{d}$ bestimmen. Allgemein ausgedrückt, kann nur eine Bewegung orhogonal zur Kante erkannt werden.

6.4.5 Praktisches Vorgehen

Man geht mit einem Fenster, das genauso groß ist wie das des Trackers, über das Bild (Abbildung 6.5), errechnet für jede Position des Fensters die Matrix $\boldsymbol{G}$ und bestimmt anschließend den kleineren der beiden Eigenwerte von $\boldsymbol{G}$ (Abbildung 6.6). Erzeugt man aus diesen Eigenwerten ein Histogramm, erhält man Abbildung 6.7. Es gibt ziemlich

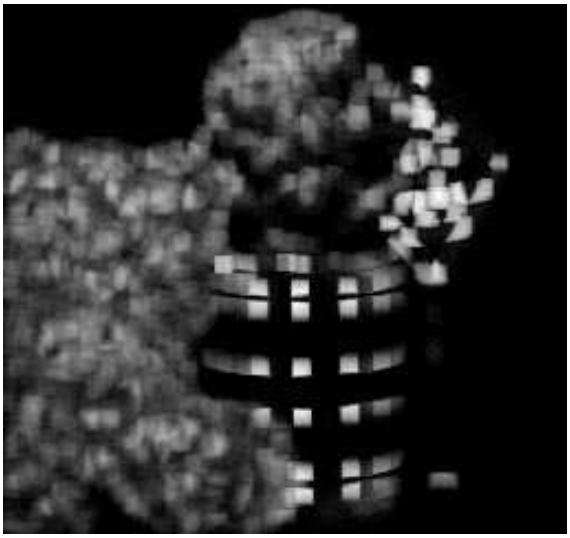


Abbildung 6.6: Die kleineren Eigenwerte von $\boldsymbol{G}$ (siehe Formel 6.8) von Abbildung 6.5. Hellere Stellen entsprechen höheren Werten.

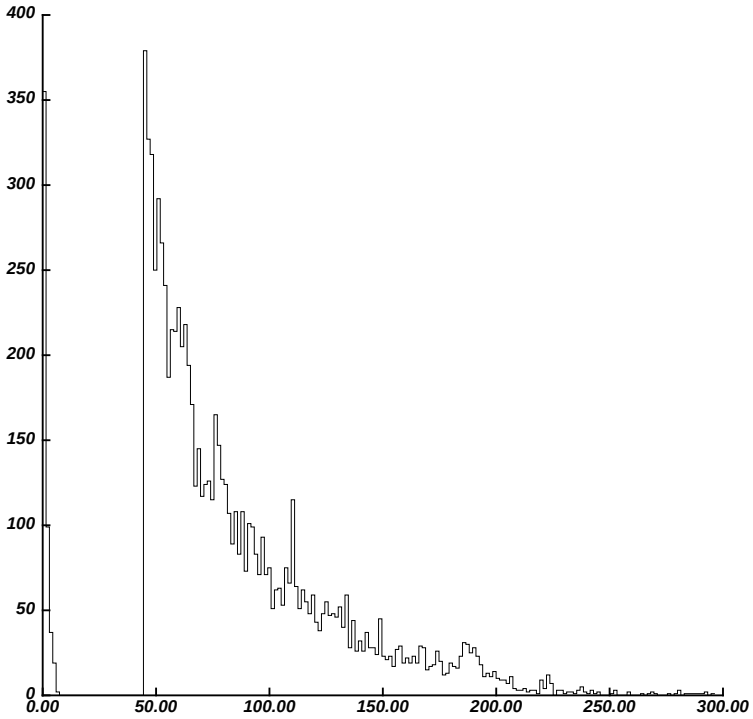


Abbildung 6.7: Histogramm der kleineren Eigenwerte. Die X-Achse stellt die Eigenwerte dar, die Y-Achse die Anzahl der jeweiligen Eigenwerte.

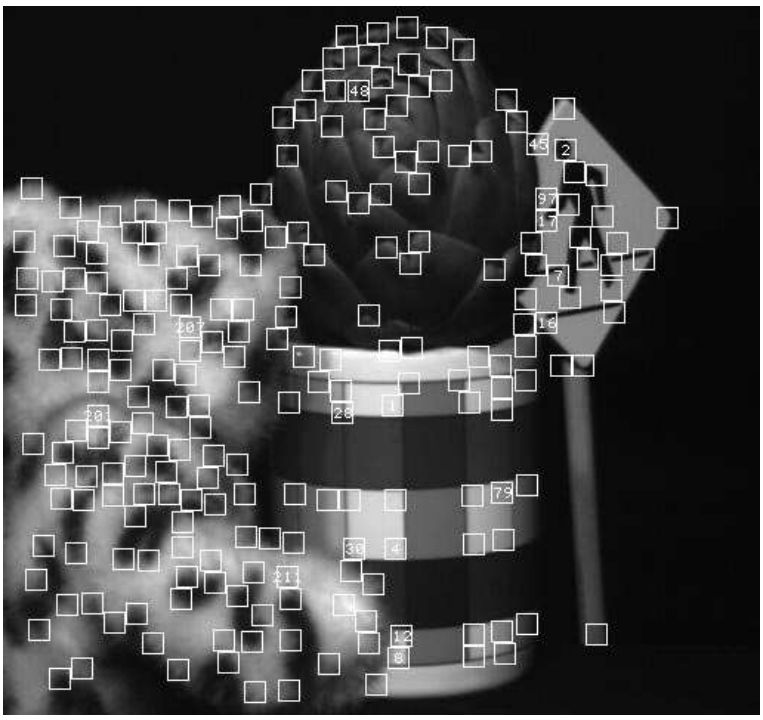


Abbildung 6.8: Die verbliebenen, guten Features

viele Eigenwerte, die knapp über 0 liegen - alle dazugehörigen Features liegen in gleichmäßig gefärbten Bereichen wie dem Hintergrund, den Quadraten auf dem Becher oder dem Hintergrund des Schildes. Auch Bereiche, die unter dem Apertur Problem (siehe Unterabschnitt 6.4.4) leiden, gehören hierzu. Diese Features sind nicht zum Tracken geeignet und sollen weggelassen werden.

Um nun die guten Features auszuwählen, wählt man einen Schwellwert aus der Lücke zwischen den Eigenwerten, die fast Null sind, und den höheren Eigenwerten: z.B. den Wert 10. Nur die Eigenwerte, die über diesem Schwellwert liegen, werden weiterverwendet und zunächst absteigend sortiert. Nun geht man, bei den höchsten Eigenwerten angefangen, durch alle Eigenwerte und lässt alle Features weg, die sich überlappen. Alle übrigen Features sind gute Features (siehe Abbildung 6.8).

Da man alle Features, deren kleinerer Eigenwert unter der Schwelle 10 lag, nicht weiter berücksichtigt hat, sind beide Eigenwerte der guten Features groß genug – Die Features heben sich also deutlich vom Rauschen ab und sind stark texturiert.

Die Matrix $\boldsymbol{G}$ der verbleibenden Features ist ebenfalls gut konditioniert; deren beide Eigenwerte können nicht beliebig weit auseinander liegen, da die Intensitätsunterschiede durch den maximal erlaubten Pixelwert begrenzt sind.

6.4.6 Beispielhafte gute Features

Wie man in Abbildung 6.8 sieht, wählt das Eigenwert-Kriterium die Ecken auf dem Becher und das Muster auf der Puppe als Features aus. Auf der Artischocke werden Stellen gewählt, an denen das Intensitätsmuster sehr unterschiedlich ist.

Der Hintergrund sowie die relativ gleichmäßig gefärbten Stellen auf dem Schild und auf dem Becher enthalten keine Features.

Die graden Kanten auf dem Becher und der Pfosten des Schildes enthalten auch keine Features: diese Kanten sind ein gutes Beispiel für Bereiche, die unter dem Apertur Problem leiden.

6.5 Monitoring

6.5.1 Einleitung

Nach Shi und Tomasi [4] ist die annäherungsweise Berechnung der Verschiebung ausreichend, wenn man ein Bild mit dem nächsten vergleicht. Hierbei wird angenommen, dass sich alle Pixel in einem Fenster *gleich* verschieben (siehe Unterabschnitt 6.2.2).

Tatsache bleibt dennoch, dass sich verschiedene Punkte innerhalb eines Trackingfensters *unterschiedlich* verhalten können:

- Die zugehörige dreidimensionale Fläche kann stark abgeschrägt sein,
- Die Pixelwerte können sich von Bild zu Bild verzerren,
- Das Fenster kann entlang eines verdeckenden Randes sein,
- Pixel bewegen sich mit verschiedenen Geschwindigkeiten oder verschwinden und tauchen wieder auf.

Dadurch ergeben sich zwei Probleme:

- Wie weiß man, dass man noch das gleiche Fenster verfolgt, wenn sich dessen Inhalt ändert?
- Wie werden die Verschiebungen der einzelnen Pixel innerhalb des Trackingfensters korrekt berücksichtigt?

Zur Lösung dieser Probleme verwendet man Monitoring:

Man vergleicht ein Feature aus dem ersten Bild mit dem dazugehörigen Feature aus dem aktuellen Bild.

Da zwischen diesen beiden Bildern viele andere Bilder liegen können, reicht eine einfache Verschiebung nicht mehr aus, um das Feature aus dem ersten Bild im aktuellen Bild wiederzufinden. Um alle Veränderungen zwischen den Bildern berücksichtigen zu können, verwendet man daher ein „Affines Bewegungsfeld“ (siehe Formel 6.9).

Die von Shi und Tomasi [4] vorgestellte Monitoring-Methode erkennt Features, die verdeckt werden, solche die wieder auftauchen und Features, zu denen es keine entsprechenden Punkte in der Wirklichkeit gibt. Beispiele hierfür:

- Der Rand einer Reflektion auf einer glänzenden Oberfläche. Die Reflektion kann sich von Bild zu Bild verändern, die Oberfläche bleibt dennoch die gleiche.
- Bei dem Bild eines Baumes kann sich ein horizontaler Zweig im Vordergrund mit einem vertikalen Zweig im Hintergrund schneiden. Dieser Schnittpunkt existiert nur auf dem Bild, nicht in der Realität, da die beiden Zweige weit voneinander entfernt sind.

6.5.2 Zwei Modelle der Bildbewegung

Es ist am besten, zwei Modelle der Bildbewegung zu verwenden: Für das Tracking liefert die Berechnung der Bildverschiebung (siehe Abschnitt 6.3) zuverlässigere Ergebnisse als die Affine Formel 6.9, wenn die Kameraverschiebung zwischen zwei Bildern gering ist. Die Affine Formel wiederum ist notwendig, um die Unähnlichkeit (siehe Formel 6.10) weit auseinanderliegender Bilder zu bestimmen.

Um die einzelnen Verschiebungen innerhalb eines Fensters zu repräsentieren, verwendet man ein „Affines Bewegungsfeld“:

$$\delta = \boldsymbol{D}\boldsymbol{x} + \boldsymbol{d}^{2} \tag{6.9}$$

mit $\boldsymbol{D} = \begin{pmatrix} d_{xx} & d_{xy} \\ d_{yx} & d_{yy} \end{pmatrix}$ als Deformationsmatrix und $\boldsymbol{d}$ als die Verschiebung des Fenstermittelpunkts.

Ein Punkt $\boldsymbol{x}$ im aktuellen Bild I verschiebt sich zu dem Punkt $\boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}$ im späteren Bild J, wobei $A = \mathbf{1} + \boldsymbol{D}$ und $\mathbf{1}$ die 2 x 2 Einheitsmatrix ist:

$$\underbrace{J(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{d})}_{aktuelles\ Bild} = \underbrace{I(\boldsymbol{x})}_{erstes\ Bild}$$

Siehe zum Vergleich Formel 6.3.

[2] Setzt man $\boldsymbol{D} = 0$, erhält man wieder die einfache Verschiebung: $\delta = \boldsymbol{d}$

6.5.3 Bildbewegung berechnen

Anhand eines Unähnlichkeitsmaßes kann man ein Feature überwachen. Unähnlichkeit ist der quadratische Mittelwert (engl.: root mean square) eines Features zwischen dem ersten und dem aktuellen Bild:

$$\epsilon = \int_W [J(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}) - I(\boldsymbol{x})]^2 \; w \; dx^3 \tag{6.10}$$

Siehe zum Vergleich Formel 6.4.

Aufgrund von Rauschen und dadurch, dass das Modell des Affinen Bewegungsfeldes nicht perfekt ist, ist Gleichung 6.9 im Allgemeinen nicht exakt erfüllt. Um die Parameter der Bewegung zu bestimmen, muss man daher diejenige Matrix $\boldsymbol{A}$ und denjenigen Vektor $\boldsymbol{d}$ finden, welche die Unähnlichkeit (Formel 6.10) minimieren. Zu diesem Zweck wird nach $\boldsymbol{D}$ und nach $\boldsymbol{d}$ abgeleitet und das Ergebnis gleich 0 gesetzt. Das resultierende System wird anschliessend mit Hilfe der Taylor-Reihen angenähert und auf den linearen Anteil verkürzt (wie auch im Unterabschnitt 6.3.2) und man erhält:

$$J(\boldsymbol{A}\boldsymbol{x} + \boldsymbol{d}) = J(\boldsymbol{x}) + \boldsymbol{g}^T(\boldsymbol{u}) \tag{6.11}$$

Daraus erhält man das folgende lineare 6 x 6 Gleichungssystem:

$$\boldsymbol{T}\boldsymbol{z} = \boldsymbol{a} \tag{6.12}$$

Um kurz auf die Bestandteile dieser Formel einzugehen, sei hier gesagt, dass der Vektor $\boldsymbol{z}$ am wichtigsten ist, da er die gesuchten Werte, nämlich die Bestandteile der Deformationsmatrix $\boldsymbol{D}$ und der Verschiebung $\boldsymbol{d}$ enthält:

$$\boldsymbol{z}^T = (d_{xx} \; d_{yx} \; d_{xy} \; d_{yy} \; d_x \; d_y) \tag{6.13}$$

Der Fehlervektor $\boldsymbol{a}$ leitet sich aus der Differenz der beiden Bilder ab (so wie der Vektor $\boldsymbol{e}$ aus Formel 6.8) und die Matrix $\boldsymbol{T}$ kann anhand eines Bildes berechnet werden (so wie die Matrix $\boldsymbol{G}$ aus Formel 6.8). Die ausgeschriebenen Formeln für $\boldsymbol{a}$ und $\boldsymbol{T}$ können bei [4] nachgeschlagen werden.

Aufgrund der Linearisierung der Gleichung 6.11 ist Gleichung 6.12 nur annäherungsweise erfüllt. Die korrekte affine Veränderung kann jedoch gefunden werden, indem man Gleichung 6.12 iterativ anwendet.

6.5.4 Konvergenz

In diesem Abschnitt wird anhand einer Simulation gezeigt, dass der iterative Tracking Algorithmus 6.12 konvergiert. Dies ist selbst dann der Fall, wenn das Ergebnis der ersten Berechnung vom richtigen Ergebnis weit entfernt ist.

Wie bereits erwähnt, bedeutet Monitoring, dass man das aktuelle mit dem ersten Bild vergleicht [4]. Um zu zeigen, dass ein Bild, welches sich über viele Frames hinweg stark verändert, trotzdem beim Vergleich mit dem ersten Bild noch erkannt wird, zeigt Abbildung 6.9 drei verschiedene Simulationen, in jeder Zeile eine. Das Originalbild in Spalte 1 entspricht dem ersten Bild einer Bildsequenz. Die simulierten Bilder in Spalte 5 entsprechen späteren Bildern aus der jeweils gleichen Sequenz, die mittlerweile stark verzerrt sind. Sie wurden verzerrt, verschoben und anschließend mit einem zufälligen Gauss-Rauschen versehen, um den Test realistischer zu machen.

Der Tracking-Algorithmus berechnet jetzt aus dem Originalbild und dem simulierten Bild den Vektor $\boldsymbol{z}$ (siehe Gleichung 6.13). Dieser enthält die Deformation und die Verschiebung, die nötig ist, um aus dem Originalbild das simulierte Bild zu machen. Wendet man diese Deformation und Verschiebung dann auf das Originalbild an, erhält man ein neues Bild; wir bezeichnen es mit „1 Iteration“. Im nächsten Iterationsschritt wird dann $\boldsymbol{z}$ neu berechnet, diesmal aus Bild „1 Iteration“ und aus dem simulierten Bild und so weiter ...

Die drei mittleren Spalten von Abbildung 6.9 zeigen diese „Iterations-Bilder“ nach 4, 8 und 19 Iterationen. Man sieht leicht, dass der Algorithmus funktioniert – die Bilder in der vierten Spalte sind denen in der fünften Spalte so ähnlich wie möglich.

[3] Mit dieser Formel kann auch die einfache Verschiebung berechnet werden, wenn man statt $\boldsymbol{A}$ die Einheitsmatrix nimmt (das entspricht der vorigen Fussnote)

[4] Der Einfachheit halber heißt es in diesem Abschnitt immer „das Bild“, obwohl es eigentlich „Feature aus dem Bild“ heißen müßte

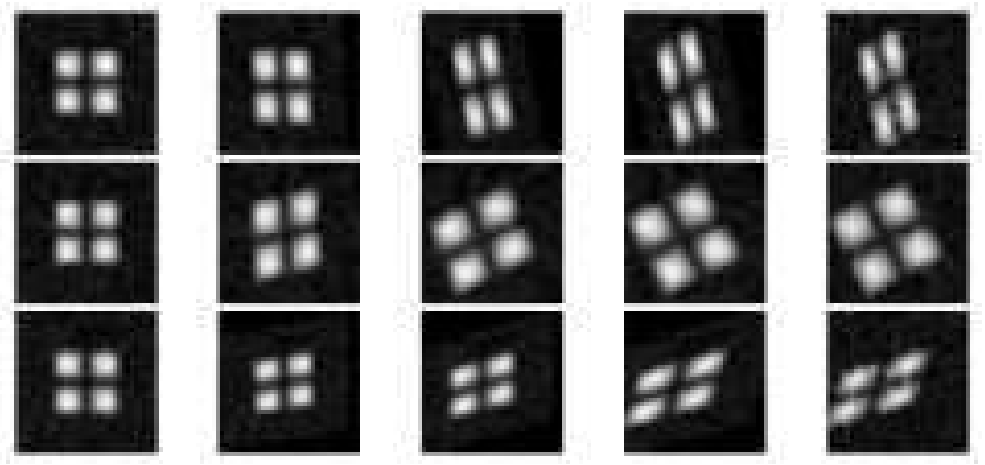


Abbildung 6.9: Konvergenz-Simulation

6.5.5 Monitoring in Aktion

In diesem Abschnitt wird in zwei Experimenten gezeigt, wie Features während des Trackings mittels Monitoring überwacht werden können, um potentiell schlechte Features zu entdecken. Abbildung 6.10 zeigt den ersten Frame einer Sequenz aus 26 Frames. Die Kamera bewegt sich von Bild zu Bild vorwärts, sodass die Features von Bild zu Bild größer erscheinen.

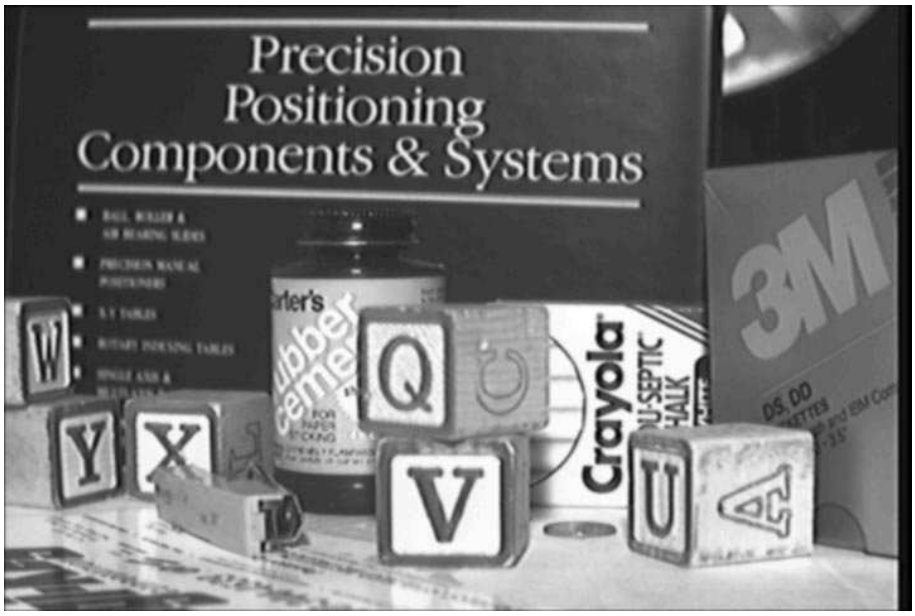


Abbildung 6.10: Beispielframe

Abbildung 6.11 zeigt die 102 guten Features.

Abbildung 6.12 zeigt die Unähnlichkeit eines jeden Features, wobei in Formel 6.10 statt $\boldsymbol{A}$ die Einheitsmatrix genommen wurde; es handelt sich also um ein Unähnlichkeitsmaß, das ohne affine Abbildung auskommt, und daher nicht alle möglicherweise auftretenden Veränderungen berücksichtigen kann. Der generelle Aufwärtstrend ist durch das Größerwerden der Features durch den Kamerazoom zu erklären.

Wie man an der Abbildung 6.12 sehen kann, ist dieses Unähnlichkeitsmaß für das Monitoring nahezu nutzlos: Abgesehen von den Features 58 und 89 haben alle Features eine vergleichbare Unähnlichkeit, weshalb nicht gut zwischen guten und schlechten Features unterschieden werden kann. Es würden zwar die Features 58 und 89 aufgrund ihrer hohen Unähnlichkeit als schlechte Features erkannt werden; zugleich würden jedoch viele andere schlechte Features unbemerkt bleiben.

Abbildung 6.11: Features des Beispielframes

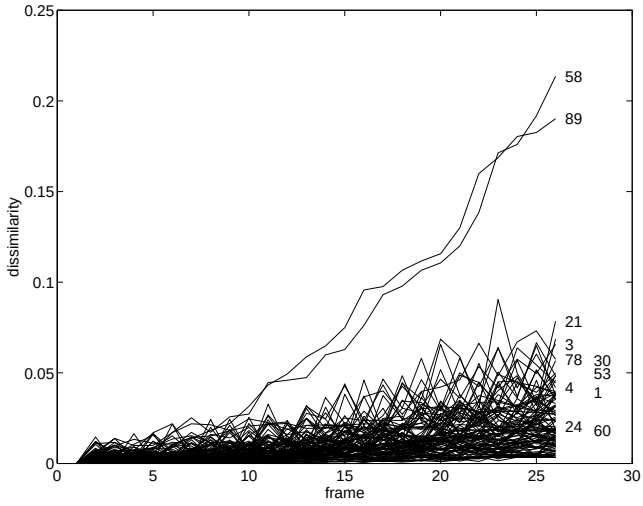


Abbildung 6.12: Unähnlichkeit, berechnet mittels einfacher Verschiebung

Nun zum Vergleich die Unähnlichkeiten, die mittels des affinen Unähnlichkeitsmasses (Formel 6.10) berechnet werden:

Der dicke Streifen am unteren Ende repräsentiert alle guten Features. Zusätzlich sind jetzt auch die schlechten Features 3,4,58,78 und 89 deutlich zu erkennen. Die Unähnlichkeitskurven der Features 24 und 60 sind sehr unregelmäßig. Die Ursache dafür ist Glättung: die Features enthalten sehr kleinen Text, dessen Größe in etwa mit der Pixelgröße des Bildes vergleichbar ist und dessen Buchstaben schlecht geglättet sind. Dieses Verhalten stellt jedoch kein Problem dar: unregelmäßige Unähnlichkeitskurven bedeuten Probleme, weshalb die dazugehörigen Features nicht weiter beachtet werden.

6.6 Zusammenfassung und Ausblick

In diesem Kapitel wurde eine Methode zur Feature Auswahl, ein Tracking Algorithmus, der auf einem Modell affiner Bildveränderungen basiert, sowie eine Technik zum Überwachen der Features während des Trackings vorgestellt. Die Auswahlmethode maximiert die Qualität des Trackers und ist daher „per Konstruktion optimal“.

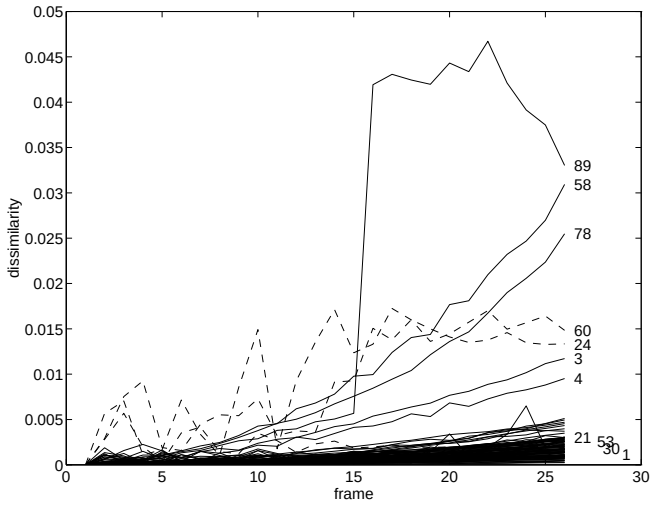


Abbildung 6.13: Unähnlichkeit, berechnet mittels des affinen Unähnlichkeitsmasses

Das Rechenaufwand für das Monitoring ist gering. Durch das Monitoring werden jedoch nicht alle Probleme des Trackings gelöst. Es mag Situationen geben, in denen ein heller Fleck auf einer glänzenden Oberfläche ein schlechtes Feature ist, sich jedoch nur wenig über viele Frames hinweg verändert: in solch einem Falll kann es sein, dass das Unähnlichkeitsmaß das Feature nicht als schlechtes Feature erkennt.

Allerdings kann schon vom Prinzip her nicht jedes Problem lokal entschieden werden. Durch das Monitoring können die schlechten Features jedoch auf ein paar Ausreißer reduziert werden; wendet man anschließend in höheren Stufen der gesamten Bildverarbeitungskette Methoden an, um Ausreißer zu entdecken, werden diese Methoden dann umso erfolgreicher sein.

Literaturverzeichnis

[1] B.D. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. In *IJCAI81, pages 674-679*, 1981.

[2] Dietrich Paulus. *Aktives Bildverstehen*. Der andere Verlag, Osnabrück, 2001. Habilitationsschrift in der Praktischen Informatik, Universität Erlangen-Nürnberg, Mai 2000.

[3] Dietrich Paulus. *Structure from Motion (Skript Sommersemester 2003)*. July 2003.

[4] Jianbo Shi and Carlo Tomasi. *Good Features to Track*. Seattle, June 1994.

[5] Carlo Tomasi and Takeo Kanade. *Detection and Tracking of Point Features*. Carnegie Mellon University, April 1991.

[6] Benno Heigl und Dietrich Paulus. *Punktverfolgung in Farbbildsequenzen*. IRB-Verlag, Stuttgart, 1997.

[7] Verschiedene. *Wikipedia - Die freie Enzyklopädie*. 2004.

Vortrag 7

Rekonstruktion aus Bildern anhand der Orthographischen Projektion

Ansgar Quernhorst

January 15th 2004

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
ansgar@uni-koblenz.de
http://www.uni-koblenz.de/~ansgar

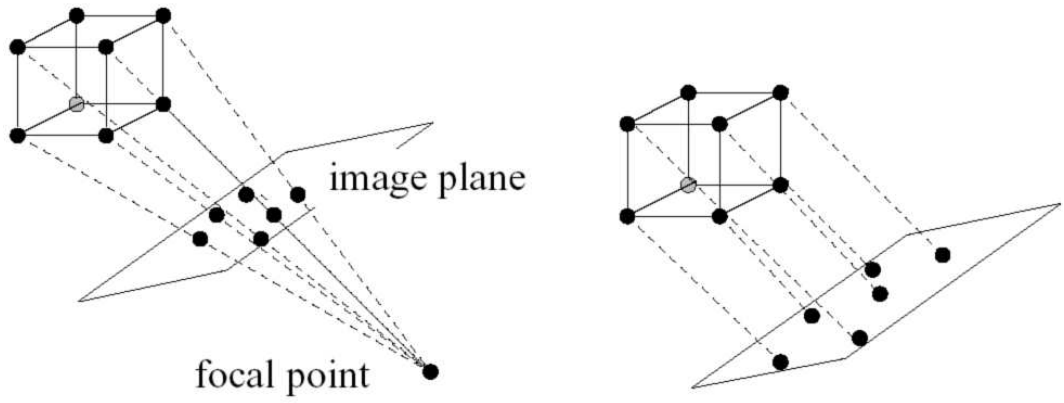


Abbildung 7.1: Perspektivische Projektion: Die Sehstrahlen laufen auf einen Fluchtpunkt zu; Orthographische Projektion: Sehstrahlen verlaufen parallel.

7.1 Einleitung

Die vorliegende Ausarbeitung beschäftigt sich mit einer Lösung des "Struktur aus Bewegung Problems", also Szenengeometrie und Kamerabewegung aus einer Bildfolge rückzurechnen. Grundlage ist die Faktorisierungsmethode von Carlo Tomasi und Takeo Kanade aus dem Jahr 1992 ("Shape an Motion from Image Streams: a Factorization Method, Full Report on the Orthographic Case"). Der größte Unterschied zu anderen Ansätzen liegt in der Tatsache, dass Tomasi und Kanades Methode keinen Zwischenschritt in Form einer Tiefenberechnung benötigt, aber dennoch qualitativ gute Ergebnisse erzielen.

7.2 Grundlagen der Orthographische Projektion

Die Orthographische Projektion ist die simpelste Methode um 2D-Bildkoordinaten aus 3D-Kamerakoordinaten zu errechnen. Unter Orthographie verlaufen alle Sehstrahlen parallel, d.h. jegliche (realistische) Perspektive entfällt. Bei zwei im Bild nebeneinander dargestellten Objekten, lässt sich nicht mehr rekonstruieren, ob die beiden im 3D-Raum auf einer Ebene lagen oder einen Abstand in der Tiefe zueinander hatten. Ein Würfel der sich direkt vor der Kamera befindet wird genauso abgebildet wie ein gleichgroßer aber mehrere Meter entfernter Würfel.

Die Umrechnung von den Koordinaten eines Punktes im Raum (p^c) zu den entsprechenden Koordinaten im Bild (p^i), erfolgt durch Verwerfung der Tiefeninformation:

$$p^c = \begin{pmatrix} x \\ y \\ z \end{pmatrix} \rightarrow \begin{pmatrix} x \\ y \end{pmatrix} = p^i$$

Notation in Matrixform:

$$p^i = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix} p^c$$

Die Rekonstruktion einer Bildfolge stößt durch die Orthographische Projektion an ihre Grenzen, wenn die Objekte der Szene nicht hinreichend weit weg sind. So ist es zum Beispiel möglich eine Stadt von den Bildern einer am Flugzeug befestigten Kamera zu rekonstruieren. Nicht rekonstruiert werden könnte ein anderes Flugzeug, das knapp unter dem Ersten entlang fliegt. Der Algorithmus würde aufgrund der, durch die verschiedenen Entfernungen bedingten, unterschiedlichen Bewegungen einen zu großen Fehler machen.

7.3 Die registrierte Measurement-Matrix

Grundlage der Faktorisierungsmethode ist die Measurement-Matrix. Sie setzt sich aus den Koordinaten der getrackten Featurepunkte P über die F Frames der Bildfolge zusammen. Alle horizontalen Koordinaten u_{fp} werden in der $F \times P$

Matrix U, alle vertikalen Koordinaten in der Matrix V gespeichert. In diesen Matrizen repräsentiert somit jeder Reihe ein Frame und jede Spalte einen Featurepunkt. Kombiniert man diese beiden Matrizen, so erhält man die $2F \times P$ Measurement-Matrix W. Mit Kombination ist in diesem Fall ein einfaches "übereinander setzen" gemeint.

$$W = \begin{bmatrix} U \\ \hline V \end{bmatrix}$$

Für jedes Frame bzw. jede Reihe in U und V erfolgt eine Registrierung durch Subtraktion der Koordinatenmittelwerte von den Featurekoordinaten. Somit wird ab jetzt der Abstand der Featurepunkte vom Zentrum der Punktwolke eines jeden Frames gespeichert.

$$\tilde{u}_{fp} = u_{fp} - a_f \tag{7.1}$$
$$\tilde{v}_{fp} = v_{fp} - b_f \tag{7.2}$$

wobei

$$a_f = \frac{1}{P} \sum_{p=1}^{P} u_{fp} \tag{7.3}$$

$$b_f = \frac{1}{P} \sum_{p=1}^{P} v_{fp} \tag{7.4}$$

So entstehen $\tilde{U}$ und $\tilde{V}$ die zusammen die registrierte Measurement-Matrix ergeben:

$$\tilde{W} = \begin{bmatrix} \tilde{U} \\ \hline \tilde{V} \end{bmatrix} \tag{7.5}$$

7.4 Das Rang-Theorem

Nun wird der Ursprung des Welt-Referenzsystems ins Zentrum der P Punkte $s_p = (x_p, y_p, z_p)^T$ gelegt, so dass gilt:

$$\frac{1}{P} \sum_{p=1}^{P} s_p = 0 \tag{7.6}$$

Die Orientierung der Kamera wird für jedes Frame durch zwei Vektoren i_f und j_f festgelegt. Diese beiden Vektoren zeigen entlang der Zeilen und Spalten des Pixelrasters der Bildebene. Unter Orthographie sind alle "Sehstrahlen" parallel zueinander, also auch zum Kreuzprodukt von i_f und j_f (siehe Abb. 7.2):

$$k_f = i_f \times j_f \tag{7.7}$$

Um von den 3D-Weltkoordinaten die entsprechende Position in der Bildebene zu errechnen, muss eine Rotation um i_f bzw. j_f, sowie eine Translation um t_f erfolgen. $t_f = (a_f, b_f, c_f)^T$ ist der Vektor vom Weltursprung zum Ursprung der Bildebene des Frames f.

$$u_{fp} = i_f^T (s_p - t_f) \tag{7.8}$$
$$v_{fp} = j_f^T (s_p - t_f) \tag{7.9}$$

Jetzt kann man für $\tilde{u}_{fp}$ und $\tilde{v}_{fp}$ folgende Rechnung führen:

$$\begin{aligned} \tilde{u}_{fp} &= u_{fp} - a_f && (7.10)\\ &= i_f^T (s_p - t_f) - \frac{1}{P} \sum_{q=1}^{P} i_f^T (s_q - t_f) && (7.11)\\ &= i_f^T \left(s_p - \underbrace{\frac{1}{P} \sum_{q=1}^{P} s_q}_{=0 \text{ wg. } (7.6)} \right) && (7.12)\\ &= i_f^T s_p && (7.13) \end{aligned}$$

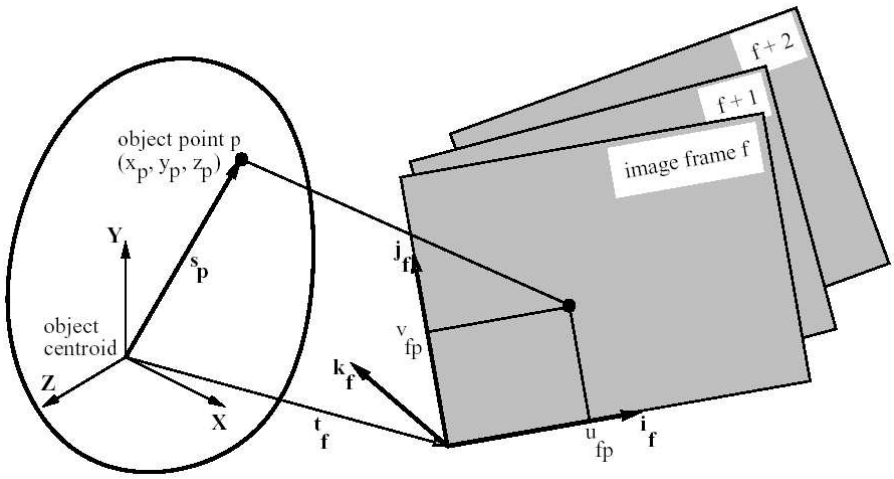


Abbildung 7.2: Der Ursprung des Weltkoordinatensystems liegt im Zentrum der Punktwolke, die Ausrichtung der Kamera wird durch i_f und j_f repräsentiert. t_f ist der Vektor vom Ursprung des Welt- zu dem des Bildkoordinatensystems.

Analog gilt $\tilde{v}_{fp} = j_f^T s_p$.
Im ersten Schritt erfolgt die Einsetzung der Gleichungen (7.3) und (7.8). Da der Vektor t_f für alle Punkte eines Frames konstant ist, konnte dieser Term im zweiten Schritt weggekürzt werden. Des weiteren macht sich die im ersten Abschnitt beschriebene Registrierung von u_{fp} und v_{fp} hier bezahlt um das Summenzeichen loszuwerden.

Da sowohl $\tilde{U}$ als auch $\tilde{V}$ die Dimension $F \times P$ haben, kann die registrierte Measurement-Matrix wie folgt in Matrixform notiert werden:

$$\tilde{W} = RS \tag{7.14}$$

$$R = \begin{bmatrix} i_1^T \\ \vdots \\ i_F^T \\ j_1^T \\ \vdots \\ j_F^T \end{bmatrix} \tag{7.15}$$

$$S = [s_1 \dots s_P] \tag{7.16}$$

Wobei die Matrix R (Rotation) die Ausrichtung der horizontalen und vertikalen Kameraachse für die gesamte Bildfolge enthält und in den Spalten von S (Shape) die Koordinaten der Featurepunkte stehen.
Da R eine $2F \times 3$ und S eine $3 \times P$ Matrix ist folgt das:

***Rang Theorem**: Ohne Rauschen hat die registrierte Measurement-Matrix $\tilde{W}$ maximal einen Rang von 3.*

Das Rang Theorem drückt aus, dass die ursprüngliche $2F \times P$ Matrix W hochgradig redundant war. Mit (7.10) kann die unregistrierte Measurement-Matrix W nun so geschrieben werden:

$$W = RS + te_P^T \tag{7.17}$$

$t = (a_1, \dots, a_F, b_1, \dots, b_F)^T$ ist ein $2F$-dimensionaler Vektor, der die Kameratranslationen über die Bildebene beinhaltet. $e_P^T = (1, \dots, 1)$ ist ein Vektor aus P Einsen. In Skalarform:

$$u_{fp} = i_f^T s_p + a_f \tag{7.18}$$

$$v_{fp} = j_f^T s_p + a_f \tag{7.19}$$

Verglichen mit Gleichung (7.8) wird klar, dass die beiden Komponenten der Kameratranslation entlang der Bildebene einfach die durchschnittlichen Reihenwerte von W sind.
Da i_f und j_f orthogonal zueinander sind, muss gelten:

$$|i_f| = |j_f| = 1 \quad \text{und} \quad i_f^T j_f = 0 \tag{7.20}$$

Um einen Referenzpunkt für die Kamerarotation zu bekommen, wird für das erste Frame festgelegt:

$$i_1 = (1,0,0)^T \quad \text{und} \quad j_1 = (0,1,0)^T \tag{7.21}$$

7.5 Singulärwertzerlegung

Um der, durch das Rang Theorem aufgezeigten Redundanz der Matrix $\tilde{W}$ entgegenzuwirken, verwenden Tomasi und Kanade die Singulärwertzerlegung. Aus $\tilde{W}$ entstehen bei dieser Methode folgende drei Matrizen: Die $2F \times P$ Matrix O_1, eine diagonale $P \times P$ Matrix Σ und eine $P \times P$ Matrix O_2:

$$\tilde{W} = O_1 \Sigma O_2 \tag{7.22}$$

Wobei $O_1^T O_1 = O_2^T O_2 = O_2 O_2^T = \mathcal{I}$, der $P \times P$ Einheitsmatrix sind.
Die wichtigste Information steckt in den diagonalen Einträgen von Σ. Dort stehen in absteigender Wertigkeit die Singulärwerte $\sigma_1 \geq \ldots \geq \sigma_P$. Da die registrierte Measurement-Matrix maximal einen Rang von drei hat, betrachten wir lediglich die ersten drei Spalten von O_1, die erste 3×3 Submatrix von Σ und die ersten drei Reihen von O_2:

$$O_1 = [O_1^{'} | O_1^{''}] \tag{7.23}$$

$$\Sigma = \left[\begin{array}{c|c} \Sigma^{'} & 0 \\ \hline 0 & \Sigma^{''} \end{array}\right] \tag{7.24}$$

$$O_2 = \left[\begin{array}{c} O_2^{'} \\ \hline O_2^{''} \end{array}\right] \tag{7.25}$$

Das heißt:

$$O_1 \Sigma O_2 = O_1^{'} \Sigma^{'} O_2^{'} + O_1^{''} \Sigma^{''} O_2^{''} \tag{7.26}$$

Wenn die Matrix $\tilde{W}$ rauschfrei ist, hat sie laut dem Rang Theorem maximal drei Singulärwerte ungleich Null. Da die Singulärwerte absteigend sortiert sind, muss $\Sigma^{'}$ alle Singulärwerte von $\tilde{W}$ enthalten. Das bedeutet, $O_1^{''} \Sigma^{''} O_2^{''}$ enthalten keinerlei relevanten Daten mehr und die beste Rang-3 Approximation von $\tilde{W}$ ist:

$$\tilde{W} = O_1^{'} \Sigma^{'} O_2^{'} \tag{7.27}$$

Für nicht so ideale Umstände, also verrauschte Bilder, folgt das:

Rang Theorem für verrauschte Bilder*: Die gesamte Form- und Rotationsinformation von $\tilde{W}$ steckt in den größten drei Singulärwerten, zusammen mit den korrespondierenden linken und rechten Eigenvektoren.*

Jetzt definieren wir folgendes:

$$\begin{aligned} \hat{R} &= O_1^{'} [\Sigma^{'}]^{\frac{1}{2}} \\ \hat{S} &= [\Sigma^{'}]^{\frac{1}{2}} O_2^{'} \\ \Rightarrow \hat{W} &= \hat{R}\hat{S} \end{aligned}$$

Die zwei Matrizen $\hat{R}$ und $\hat{S}$ haben die gleiche Größe wie die geforderte Rotationsmatrix R und die Formatrix S. Leider ist die Zerlegung von $\hat{W}$ nicht eindeutig, da:

$$(\hat{R}Q)(Q^{-1}(\hat{S})) = \hat{R}(QQ^{-1})\hat{S} = \hat{R}\hat{S} = \hat{W} \tag{7.28}$$

Q kann eine beliebige 3×3 Matrix sein, $\hat{R}Q$ sowie $Q^{-1}\hat{(}S)$ sind trotzdem gültige Zerlegungen von $\hat{W}$. $\hat{R}$ und $\hat{S}$ sind also grundsätzlich verschieden von R und S aber laut Tomasi/Kanade muss eine lineare Transformation zwischen ihnen existieren.
Es gibt also eine 3×3 Matrix Q, so dass:

$$R = \hat{R}Q \tag{7.29}$$

$$S = Q^{-1}\hat{S} \tag{7.30}$$

Um Q zu finden sehen wir, dass die Reihen der wahren Rotationsmatrix R Einheitsvektoren und die ersten F Reihen orthogonal zu den zweiten F Reihen sind. Das führt zu folgendem überbestimmten Gleichungssystem:

$$\hat{i_f}^T Q Q^T \hat{i_f} = 1 \tag{7.31}$$

$$\hat{j_f}^T Q Q^T \hat{j_f} = 1 \tag{7.32}$$

$$\hat{i_f}^T Q Q^T \hat{j_f} = 0 \tag{7.33}$$

Basierend auf den bisherigen Erkenntnissen können wir nun einen Grobalgorithmus für die Faktorisierung der registrierten Measurement-Matrix $\tilde{W}$ aufstellen.

1. Berechne die Singulärwertzerlegung $\tilde{W} = O_1 \Sigma O_2$
2. Definiere $\hat{R} = O_1^{'}[\Sigma^{'}]^{\frac{1}{2}}$ und $\hat{S} = [\Sigma^{'}]^{\frac{1}{2}} O_2^{'}$
3. Berechne die Matrix Q
4. Berechne $R = \hat{R}Q$ und $S = Q^{-1}\hat{S}$

7.6 Experimente

Tomasi und Kanade haben ihre Faktorisierungsmethode anhand zweier Bildsequenzen getestet. In der ersten Sequenz wurde das Modell eines Gebäudes (genannt "Hotel") in der kontrollierten Umgebung eines Labors aufgenommen. Die Kamera wurde dabei leicht gekippt und um das Modell herumbewegt. Um eine saubere, ruckfreie Bewegung zu gewährleisten, war die Kamera auf einen mechanischen Arm montiert. Die zweite Sequenz wurde mit einer handgetragenen Kamera gefilmt und zeigt die Front eines echten Hauses. Hier hatte der Algorithmus mit einigen Problemen wie wechselnden Lichtverhältnissen, Schatten und der wackelnden Kamera zu kämpfen.

1. "Hotel" Bildsequenz
   Die gesamte Animation ist 150 Bilder Frames lang. Zum Tracken der Featurepunkte wurde die Lucas-Kanade Methode verwand. Mit ihr werden die Featurepunkte automatisch gewählt und eventuell manuell wieder verworfen, falls sich die Punkte als schlecht herausstellen. Konkret wurden 430 Punkte gewählt, von denen sich 42 im Laufe der Sequenz als zu schwer zu tracken erwiesen. In die Measurement-Matrix wurden also die Koordinaten von 388 Features eingetragen.
   Dadurch, dass die Kamera auf einem mechanischem Arm angebracht war, konnte die reale Kamerabewegung gut mit der rekonstruierten verglichen werden. Es stellte sich heraus, das die größte Abweichung gerade einmal 0,4° betrug. Um die Rekonstruktion der Form zu beurteilen, wurden die errechneten Maße des Hotels mit den ausgemessenen des Modells verglichen. Bei einer Gesamtgröße des Hauses von ca. 8 Zentimetern war der Fehler geringer als ein Millimeter. Damit sind die Ergebnisse unter "sauberen" Laborbedingungen sehr zufriedenstellend. Grund für einen Teil der Ungenauigkeiten ist die verwendete Orthographie. Wie Eingangs kurz erwähnt, hat es sich in weiteren Experimenten jedoch gezeigt, dass der so entstandene Fehler in direktem Zusammenhang mit der Entfernung der Szene von der Kamera steht.

2. "Haus" Bildsequenz
   Die zweite, aufgrund der äußeren Umstände weitaus schwerer zu rekonstruierende Sequenz, ist 180 Frames lang. Es wurde eine Rotation von ungefähr 15° um einen Hauseingang aufgenommen. Das größte Problem bestand im Tracken der Featurepunkte, da viele Bilder durch schnelle Bewegungen verwaschen waren und wechselnde Lichtverhältnisse für Probleme sorgten. Deshalb wurden die Bereiche um das Fenster oben links und das Treppengeländer unten links ausmaskiert und nicht für weitere Berechnungen verwendet. 376 Punkte konnte der Tracker mehr oder weniger gut verfolgen.
   Um das Ergebnis der Rekonstruktion besser veranschaulichen zu können, haben Tomasi und Kanade die Featurepunkte trianguliert und mit einer Textur versehen. Dennoch sieht man deutlich, dass die Geometrie nur grob erkannt wurde. Da sämtliche andere Methoden mit solchen Szenarien ebenfalls ihre Probleme haben, hat sich die Faktorisierungsmethode etabliert und findet auch heute noch oft Verwendung.

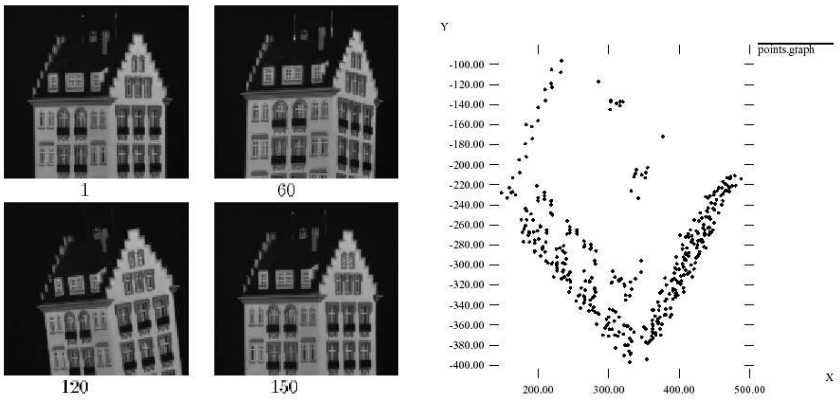


Abbildung 7.3: Links: Vier Frames aus der Bildsequenz, Rechts: Die Rekonstruierten Punkte aus einer Perspektive die in der Animation nicht vorkommt.

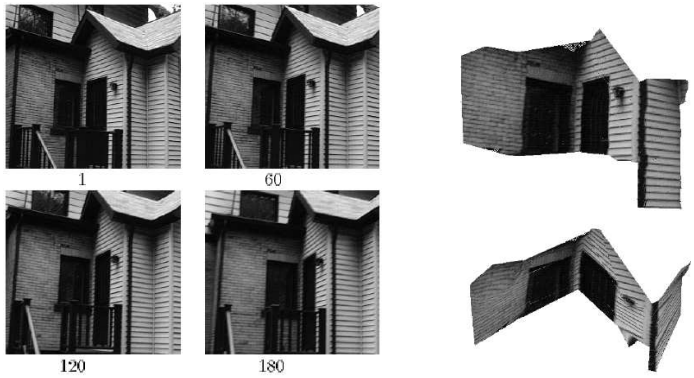


Abbildung 7.4: Links: Vier Frames aus der Bildsequenz, Rechts: Die triangulierte und mit Texturen versehne, rekonstruierte Geometrie

[1] Dietrich Paulus. *Structure from Motion*. 2003. Skript zur Vorlesung SS 2003.

[2] Carlo Tomasi and Takeo Kanade. Shape and Motion from Image Streams: A Factorization Method Part 2. Detection and Tracking of Point Features. Technical Report CMU-CS-92-104, März 1992.

Vortrag 8

Struktur aus Bewegung: Perspektivische Projektion

Andreas Thun

22. Januar 2004

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
anthun@uni-koblenz.de
http://www.uni-koblenz.de/∼anthun

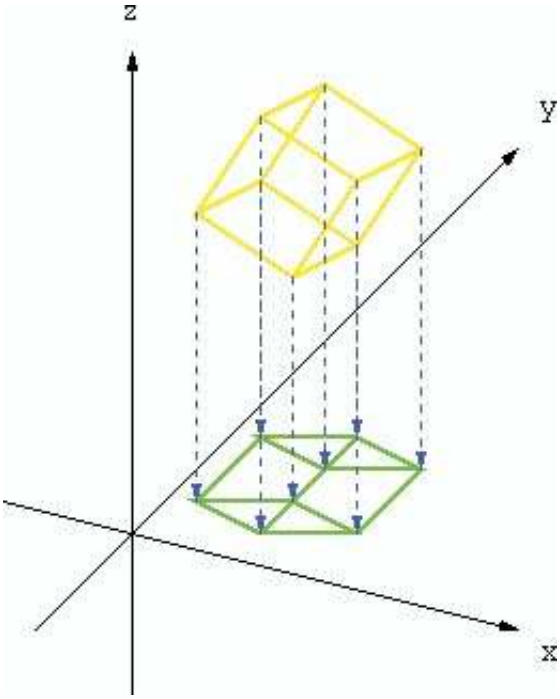


Abbildung 8.1: Orthogonale Projektion eines Drahtgitterwürfels auf die XY-Ebene

8.1 Einleitung

Der Acht-Punkte-Algorithmus stellt eine Möglichkeit dar, die Fundamentalmatrix auszurechnen. Dabei arbeitet er linear und ist damit deutlich schneller, als viele iterative Algorithmen. Der Vorteil der Geschwindigkeit geht keineswegs mit einem Qualitätsverlust einher, die Ergebnisse sind vergleichbar gut.

8.2 Grundlagen

Dieser Abschnitt dient dazu, die für das Verständnis des Algorithmus wichtigen Grundlagen kurz zu wiederholen. Sie können in den vorausgehenden Kapiteln detaillierter darüber lesen.

8.2.1 Perspektivische Projektion

Definition Projektion

Projektion heißt allgemein, dass ein Objekt aus einem n-dimensionalen Raum in einem Raum geringerer Dimensionalität abgebildet wird. Im Folgenden betrachten wir nur die Projektion von 3D-Objekten auf Flächen, so dass eine 2D-Abbildung entsteht.

Orthogonale Projektion

Abbildung 8.1 zeigt die orthogonale Projektion eines Würfels auf die XY-Ebene. Man könnte sich vorstellen, dass es eine unendlich weit entfernte Lichtquelle gibt, die den Schatten des Würfels auf den Boden wirft.

Die zugehörige Projektionsmatrix sieht so aus:

$$\begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

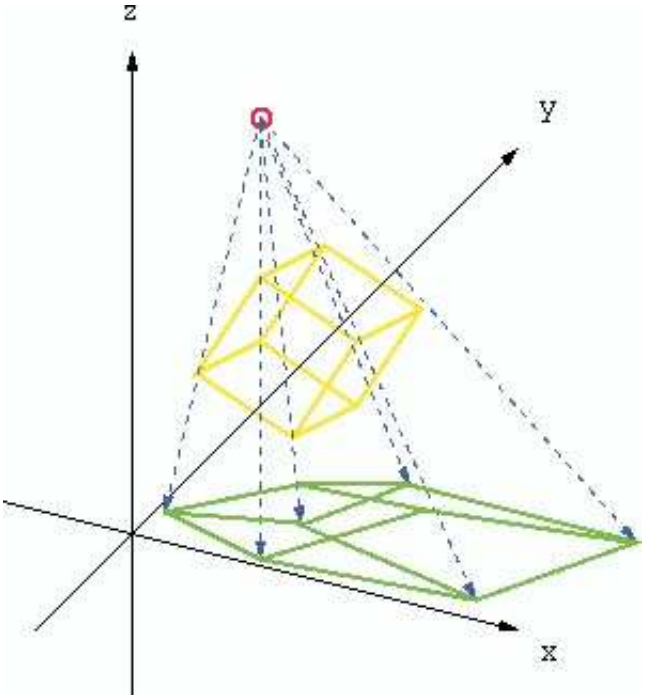


Abbildung 8.2: Perspektivische Projektion eines Drahtgitterwürfels auf die XY-Ebene

Man ignoriert einfach die Z-Komponente jedes 3D-Punktes.

Perspektivische Projektion

Um eine solche Projektion etwas realistischer zu gestalten, gibt es die perspektivische Projektion. Hier wird die Lichtquelle, die den Schatten des Drahtgitterwürfels auf die Ebene wirft, nun in endlicher Entfernung angenommen. Allgemein kann man sagen, dass Objekte gleicher Größe größer dargestellt werden, wenn sie näher an der Lichtquelle liegen, bzw. kleiner, wenn sie weiter von ihr weg sind.

Dies wird auch auf der Abbildung 8.2 deutlich. Bei einem Würfel sind natürlich alle Kanten gleich lang, die jeweiligen Schatten sind aber unterschiedlich groß: je näher an der Lichtquelle, desto größer. Es kommt also zu perspektivischen Verzerrungen. Eine Eigenschaft findet man auch später bei der Fundamentalmatrix wieder: man kann bei der Abbildung nicht unterscheiden, ob man nun einen kleinen Würfel nah an der Kamera oder einen größeren weiter weg aufgenommen hat.

Auch für die perspektivische Projektion lässt sich eine Projektionsmatrix herleiten. Man betrachte Abbildung 8.3.

Sie zeigt einen Schnitt quer zur Y-Achse unserer Beispielszene mit dem Würfel. Abgebildet sind nur noch die Lichtquelle mit einem Abstand d zur Bildebene, eine Würfelkante und deren Abbildung auf der Ebene.

Diese Darstellung wird nun in Abbildung 8.4 normalisiert.

Die Lichtquelle - auch Augpunkt, also der Ort des Betrachters, genannt - liegt nun mit Abstand $d = 1$ von der Abbildungsebene auf der positiven Z-Achse.

$$\text{Augpunkt} : \begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}$$

Die Projektion P' eines Punktes P kann man durch einfachen Dreisatz errechnen.

$$P = \begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

$$P' = \begin{pmatrix} \frac{x}{1-z} \\ \frac{y}{1-z} \\ 0 \end{pmatrix}$$

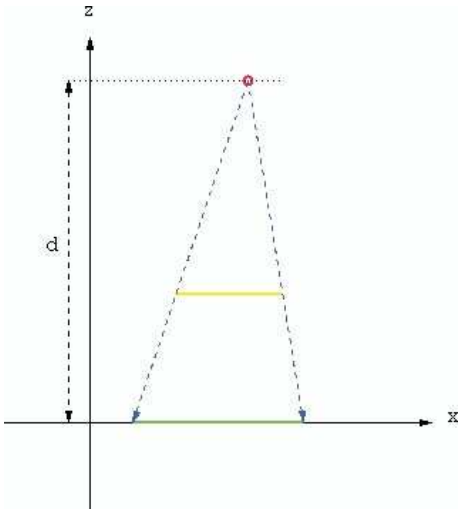


Abbildung 8.3: Persp. Projektion: Querschnitt der Beispielszene mit Augpunkt, einer Würfelkante und deren Abbildung

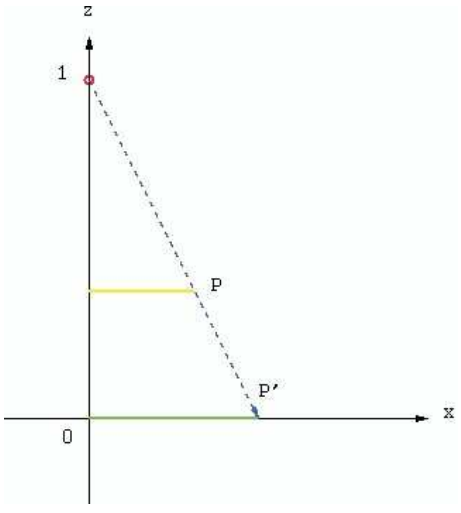


Abbildung 8.4: Persp. Projektion: Normalisierte Variante des Querschnitts

Bei $z = 0$ liegt der abzubildende Punkt bereits in der Abbildungsebene, x und y werden also nicht verändert. Bei $z = 1$ würde der Punkt im Augpunkt liegen, das ist nicht erlaubt. Ist z größer als 0 und kleiner als 1 liegt P entsprechend zwischen Augpunkt und Abbildungsfläche.

Die Projektionsmatrix ergibt sich nun wie folgt:

$$\begin{pmatrix} \frac{1}{1-z} & 0 & 0 & 0 \\ 0 & \frac{1}{1-z} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

x und y werden entsprechend ihrem Abstand zur Abbildungsebene skaliert und z wird ignoriert.

Für ein beliebiges d ergibt sich nun die folgende Projektionsmatrix.

$$\begin{pmatrix} \frac{d}{d-z} & 0 & 0 & 0 \\ 0 & \frac{d}{d-z} & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix}$$

Für $d \to \infty$ nähert sich diese Matrix der für die orthogonale Projektion ($diag \begin{pmatrix} 1 & 1 & 0 & 1 \end{pmatrix}$) an.

8.2.2 Epipolargeometrie

Die perspektivische Projektion wird von der Epipolargeometrie genutzt. Dabei gibt es Kameras, die aus verschiedenen Perspektiven eine Szene mittels perspektivischer Projektion aufnehmen. Hier wird nur die Epipolargeometrie für zwei Kameras wiederholt.

Auf beiden Bildern in Abbildung 8.5 sehen wir eine Szene, aber die Bilder sind etwas unterschiedlich, da die Kameras verschiedene Perspektiven haben.

Abbildung 8.5: Beispielszene aus zwei Perspektiven

Die mathematischen Zusammenhänge, die die Epipolargeometrie beschreibt, werden mit Hilfe der Grafik 8.6 deutlich.

- P ist irgendein Punkt in der Welt, den die beiden Kameras auf die jeweiligen Projektionsflächen B_1 und B_2 abbilden.
- C_1 und C_2 sind die Projektionszentren der Kameras.
- Die drei Punkte P, C_1 und C_2 spannen die so genannte Epipolarfläche auf (das Dreieck; man kann auch Epipolarebene sagen, das ist dann die Ebene, in der die Epipolarfläche liegt).
- P wird von der ersten Kamera auf p_1 und von der zweiten auf p_2 abgebildet. p_1 und p_2 nennt man korrespondierende Punkte.
- Die Epipole E_1 für das Bild der ersten Kamera und E_2 für das der zweiten sind die Schnittpunkte der Grundlinie von C_1 nach C_2 mit der jeweiligen Bildebene. In dem Epipol schneiden sich alle Epipolarlinien eines Bildes.
- Die Epipolarlinien l_1 und l_2 lassen sich auf zwei Arten beschreiben:

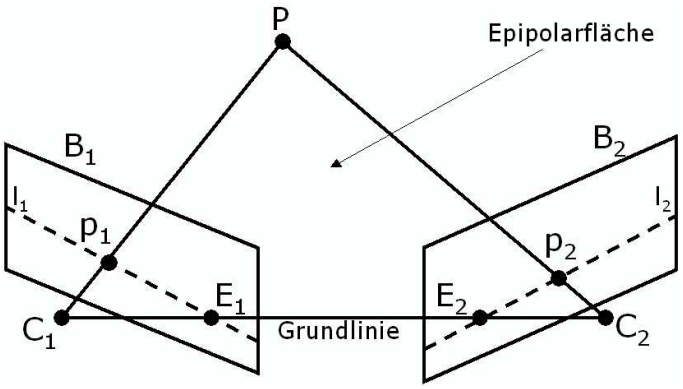


Abbildung 8.6: Epipolargeometrie

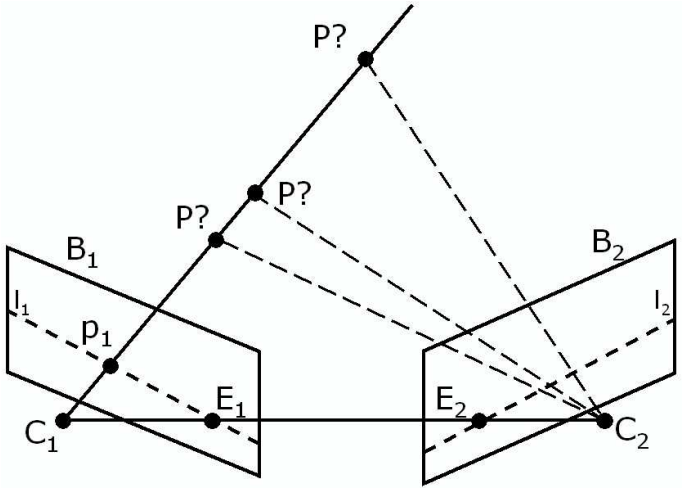


Abbildung 8.7: Epipolarbedingung

- Gerade durch Abbildungspunkt (z.B. p_1 oder p_2) und zugehörigen Epipol.
- Schnittgerade der Bildebene mit der Epipolarebene.

Daraus kann man nun zwei Bedingungen formulieren:

Epipolarbedingung

Die Epipolarbedingung (s. Abb. 8.7) sagt aus, dass zu einem Punkt p_1 der korrespondierende p_2 auf der zum ersten Punkt im anderen Bild liegenden Epipolarlinie l_2 liegen muss.

Hat man nur das erste Bild, kann man keine Aussage darüber treffen, wie weit P entfernt ist. Die Gerade durch C_1 und p_1 (und damit auch durch P) verbindet man in jedem Punkt mit C_2, so dass neue Geraden entstehen. Diese Geraden bilden die Epipolarebene, deren Schnittgerade mit der Bildebene B_2 die Epipolarlinie l_2 ist.

Koplanaritätsbedingung

Die zweite Bedingung ist die Koplanaritätsbedingung (s. Abb. 8.8). Daraus, dass der korrespondierende Punkt zu p_1 auf l_2 liegen muss, kann man folgern, dass, wenn p_1 und p_2 zu P gehören, p_1, p_2, C_1 und C_2 auf einer Ebene liegen müssen.

Drei beliebige Punkte in der Welt können eine Ebene aufspannen, wenn sie nicht gerade alle auf einer Geraden liegen, was hier aber nicht der Fall ist. Zwei der Punkte - nämlich C_1 und C_2 - sind für alle Epipolarebenen gleich. Der dritte Punkt ist jeweils der, den wir gerade mit unseren Kameras betrachten, also auf Abbildung 8.8 eines der verschiedenfarbigen Ps. Die Geraden durch zwei dieser Punkte liegen natürlich auch wieder in der entsprechenden Ebene. Die

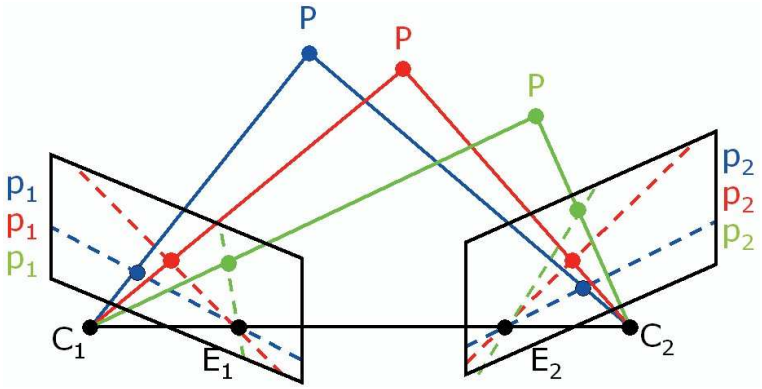


Abbildung 8.8: Koplanaritätsbedingung

Gerade durch einen Weltpunkt und ein Kamerazentrum bildet beim Durchstoßen der zugehörigen Bildebene gerade die Abbildung - also p_1 für C_1 bzw. p_2 für C_2 - und liegt damit natürlich auch in derselben Ebene.

Beispiel

Das Beispiel in Abbildung 8.9 verdeutlicht dies. Im oberen linken Bild wurden vier Punkte herausgesucht und markiert. Durch die Epipolargeometrie ergeben sich im Bild rechts oben die zugehörigen Epipolarlinien. Unten rechts sind dann dazu die korrespondierenden Punkte markiert. Das geht natürlich auch umgekehrt, also rechts unten die Punkte heraussuchen, links unten ergeben sich die Epipolarlinien und links oben wieder die korrespondierenden Punkte.

8.2.3 Fundamentalmatrix

Die Epipolar- und die Koplanaritätsbedingung werden von der Fundamentalmatrix zusammengefasst. So wird ein Zusammenhang zwischen Punkten im einen und korrespondierenden Punkten im anderen Bild dargestellt. Mathematisch lässt sich das in der Epipolargleichung ausdrücken:

$$\mathbf{p_2^T} F \mathbf{p_1} = 0$$

Dabei ist die Fundamentalmatrix F eine 3×3 Matrix vom Rang 2. Sie ist nur bis auf einen skalaren Faktor definiert, da diese Gleichung auch noch gilt, wenn man F mit einem beliebigen Skalar multipliziert. Hier kommt die Basis der perspektivischen Projektion zum Vorschein, bei der man nicht unterscheiden konnte, ob man ein kleines Objekt nah an der Kamera oder ein großes weiter weg abbildet.

F hat 9 Elemente, ist aber nur durch 7 Parameter bestimmt.

- 6 Koeffizienten, um zwei linear unabhängige Spalten $\mathbf{f_1}$ und $\mathbf{f_2}$ zu definieren
- $\mathbf{f_3}$ ist liniear abhängig und kann als beliebige Linearkombination von $\mathbf{f_1}$ und $\mathbf{f_2}$ geschrieben werden. Dabei braucht man zwei Parameter:

  $$\mathbf{f_3} = \alpha \mathbf{f_1} + \beta \mathbf{f_2}$$

  Da F nur bis zu einem skalaren Faktor definiert ist, kann man nun die gesamte Matrix durch α dividieren und erhält so:

  $$\mathbf{f_3} = \mathbf{f_1} + \frac{\beta}{\alpha} \mathbf{f_2}$$

  Es reicht also auch ein Parameter, um diese Linearkombination auszudrücken.

  In der Praxis schreibt man eher $\mathbf{f_3} = \gamma \mathbf{f_1} + (1 - \gamma)\, \mathbf{f_2}$, um Fälle wie $\mathbf{f_2} = \mathbf{f_3}$ abzudecken.

8.3 Acht-Punkte-Algorithmus

Der Acht-Punkte-Algorithmus wurde erstmals 1981 von Longuet-Higgins vorgestellt. Der Vorteil gegenüber anderen Algorithmen ist, dass diese deutlich komplizierter sind. Der Acht-Punkte-Algorithmus ist ein linearer Algorithmus und

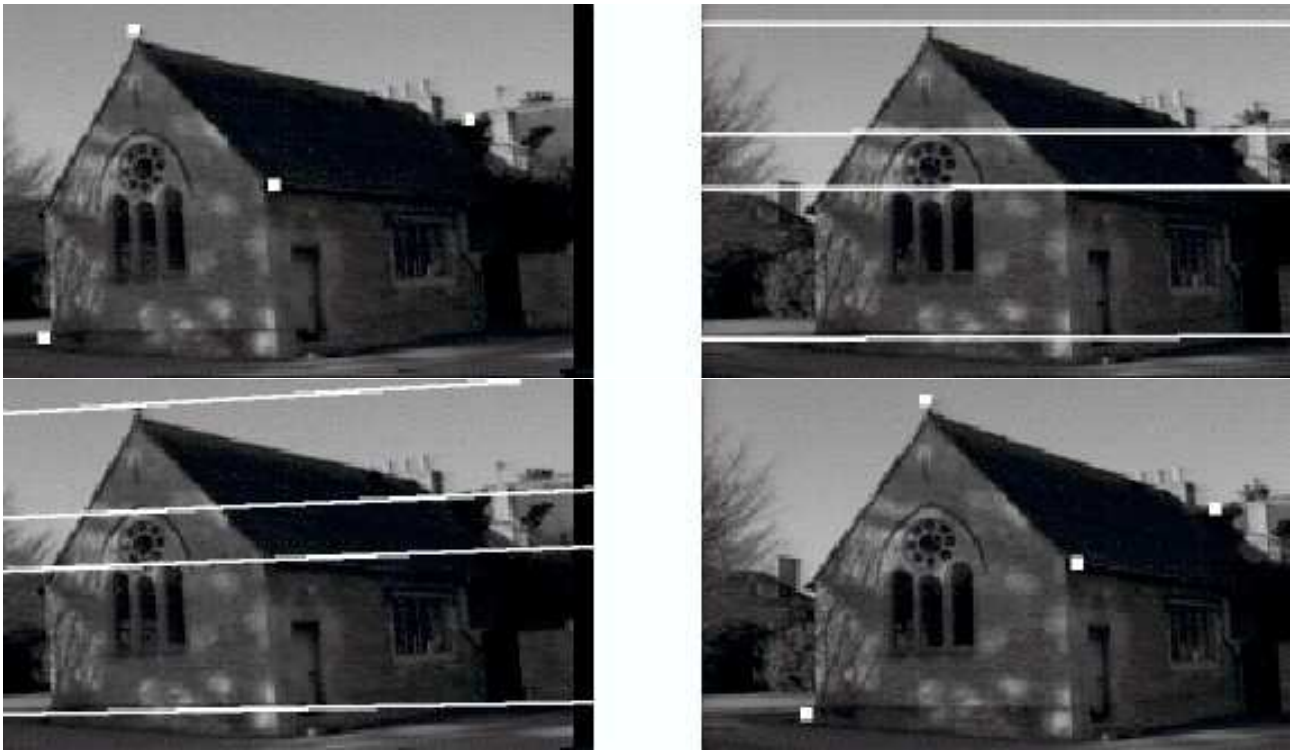


Abbildung 8.9: Beispielszene; o.l. Kamera 1 mit markierten Punkten; o.r. Kamera 2 mit Epipolarlinien zu diesen Punkten; u.r. korrespondierende Punkte zu denen o.l.; u.l. Epipolarlinien zu markierten Punkten u.r.

damit schnell und einfach zu implementieren. Lange Zeit ist er nicht benutzt worden, weil er als sehr störanfällig gegenüber Rauschen galt, was ihn für den Einsatz in der Praxis unbrauchbar macht. 1997 hat Richard I. Hartley aber gezeigt, dass diese Unzulänglichkeiten nur darauf basieren, dass man den Acht-Punkte-Algorithmus nicht gut implementiert hatte. Durch eine einfache Transformation der Koordinaten der korrespondierenden Punkte kommt der Algorithmus zu Ergebnissen, die sich mit denen anderer iterativer Algorithmen vergleichen lassen.

8.3.1 Original Acht-Punkte-Algorithmus

Hier folgt nun die Grundidee des Acht-Punkte-Algorithmus. Man betrachtet erneut die Epipolargleichung 8.1.

$$\mathbf{p_2^T} F \mathbf{p_1} = 0 \tag{8.1}$$

Dabei sind die Elemente der Gleichung folgendermaßen aufgebaut: F hat neun Eingänge F_{11} bis F_{33} und $\mathbf{p_1}$ und $\mathbf{p_2}$ sind in homogenen Koordinaten angegeben.

$$F = \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix}$$

$$\mathbf{p_1} = \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix}$$

$$\mathbf{p_2} = \begin{pmatrix} x_2 \\ y_2 \\ 1 \end{pmatrix}$$

Dies setzt man jetzt wieder in die Epipolargleichung ein und erhält:

$$\begin{pmatrix} x_2 & y_2 & 1 \end{pmatrix} \begin{pmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ y_1 \\ 1 \end{pmatrix} = 0 \tag{8.2}$$

Ausmultipliziert ergibt sich:

$$x_1x_2F_{11} + x_1y_2F_{21} + x_1F_{31} + y_1x_2F_{12} + y_1y_2F_{22} + y_1F_{32} + x_2F_{13} + y_2F_{23} + F_{33} = 0 \tag{8.3}$$

Dies ist also die Epipolargleichung für ein Paar korrespondierender Punkte aufgelöst. Diese Formel wird nun in einen Zeilenvektor $\mathbf{a}$, der die Punktkoordinaten enthält, und einen Spaltenvektor $\mathbf{f}$, der die Eingänge der Fundamentalmatrix enthält, aufgeteilt.

$$\mathbf{a} = \begin{pmatrix} x_1x_2 & x_1y_2 & x_1 & y_1x_2 & y_1y_2 & y_1 & x_2 & y_2 & 1 \end{pmatrix}$$

$$\mathbf{f}^T = \begin{pmatrix} F_{11} & F_{21} & F_{31} & F_{12} & F_{22} & F_{32} & F_{13} & F_{23} & F_{33} \end{pmatrix}$$

Für $n \geq 8$ Paare korrespondierender Punkte erhält man so jeweils ein $\mathbf{a_i}$ mit $i \in \{1 \dots n\}$. $\mathbf{f}$ bleibt natürlich immer gleich. Aus den $\mathbf{a_i}$ erstellen wir nun die $n \times 9$ Gleichungsmatrix A, die in jeder Zeile einen Vektor $\mathbf{a_i}$ enthält.

$$A = \begin{pmatrix} \mathbf{a_1} \\ \mathbf{a_2} \\ \vdots \\ \mathbf{a_n} \end{pmatrix}$$

Die umgeformte Epipolargleichung für mindestens acht Paare korrespondierender Punkte lautet dann:

$$A\mathbf{f} = 0 \tag{8.4}$$

Da die Fundamentalmatrix nur bis zu einem skalaren Faktor definiert ist, muss nun eine zusätzliche Bedingung eingeführt werden.

$$\|\mathbf{f}\| = 1$$

Dadurch wird auch die triviale Lösung verhindert.

Die Gleichung ist nur lösbar, wenn A den Rang 8 hat. Da aber meist mehr als acht Punktepaare zur Lösung des linearen Gleichungsystems 8.4 gegeben sind und diese nie völlig perfekt sind, es also eigentlich immer ein wenig Rauschen gibt, wird A den Rang 9 haben und dann ist die Gleichung nicht lösbar.

8.3.2 Singulärwertzerlegung (SVD)

SVD steht für Singular Value Decomposition und heißt übersetzt Singulärwertzerlegung. Dieses Verfahren kann man beispielsweise dazu nutzen, den Rang von Matrizen zu bestimmen oder anzupassen. Wenn eine $m \times n$ Matrix X gegeben ist, kann diese per Singulärwertzerlegung wie folgt in drei Matrizen aufgeteilt werden:

$$X^{m\times n} = U^{m\times m}\, D^{m\times n}\, V^{T^{n\times n}}$$

U ist eine $m \times m$ und V eine $n \times n$ Matrix, sie sind also quadratisch und sie sind auch orthogonal.

D ist eine Diagonalmatrix, was bedeutet, dass für alle Elemente d_{ij} mit $i \in \{1 \dots m\}$ und $j \in \{1 \dots n\}$ $d_{i\neq j} = 0$ gilt. Die $d_{i=j}$ sind die Singulärwerte von X und die Anzahl der Singulärwerte ungleich Null gibt den Rang der Matrix X an.

$$D^{m\times n} = \begin{pmatrix} \sigma_1 & 0 & 0 & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & 0 & \dots & \sigma_{\min(m,n)} \end{pmatrix}$$

Außerdem sind die Singulärwerte in absteigender Größe in der Diagonalen sortiert:

$$\sigma_1 \geq \sigma_2 \geq \dots \geq \sigma_{\min(m,n)} \geq 0$$

SVD für Gleichungsmatrix A

Betrachtet man nun die Singulärwertzerlegung der Gleichungsmatrix A:

$$A = UDV^T \tag{8.5}$$

so kann man mit D leicht den Rang von A feststellen:

$$D = \begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \cdots & 0 & \sigma_9 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix}$$

Wenn mehr als 8 Punktepaare zum Aufstellen von A genutzt wurden, so wird A - wie schon angesprochen - höchstwahrscheinlich den Rang 9 haben und für jedes Punktepaar nach dem neuten wird D eine Zeile mit Nullen enthalten. Ist der Rang 8, muss nichts weiter getan werden, denn das Gleichungssystem ist bereits lösbar. Ein Rang von weniger als 8 sollte nicht vorkommen, dazu kann aber mehr in [2] nachgelesen werden.

Bei einem Rang von 9 wird einfach $\sigma_9 = 0$ gesetzt und man erhält:

$$\begin{pmatrix} \sigma_1 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & 0 \\ 0 & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 0 \\ 0 & 0 & 0 & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & 0 \end{pmatrix} = D'$$

Setzt man D' anstelle von D zurück in die Gleichung 8.5 ein, wird daraus nun:

$$UD'V^T = A'$$

und A' hat den Rang 8, so dass das entsprechende Gleichungssystem $A'\mathbf{f} = \mathbf{0}$ lösbar wird.

8.3.3 Transformation der Punktkorrespondenzen

Hartley hat nun in seinem Paper [2] überlegt, ob es Auswirkungen hat, wenn man für die Bilder verschiedene Koordinatensysteme hat - also ob man z.B. den Koordinatenursprung links oben oder in die Mitte des Bildes legt. Des Weiteren wollte er wissen, inwieweit sich die Wahl von Koordinaten im Bild auf das Ergebnis des Acht-Punkte-Algorithmus auswirkt.

Er konnte zeigen, dass es eben zu unterschiedlichen Ergebnissen kommen kann.[2] Das ist natürlich keine wünschenswerte Eigenschaft des Acht-Punkte-Algorithmus und daher hat Hartley weiterüberlegt, wie man durch Normalisierungen diesem Effekt entgegenwirken kann. Dies geschieht durch eine Transformation der Pixelkoordinaten vor Anwendung des eigentlichen Acht-Punkte-Algorithmus. Die Transformation wird in zwei Schritten und für beide Bilder einzeln durchgeführt.

Schritt 1: Translation

Der erste Schritt ist eine Translation. Die Pixel, die später verwendet werden, bilden eine Punktwolke. Nun muss man die Mitte dieser Punktwolke bestimmen und den Mittelpunkt anschließend in den Koordinatenursprung verschieben. Abbildung 8.10 verdeutlicht dies.

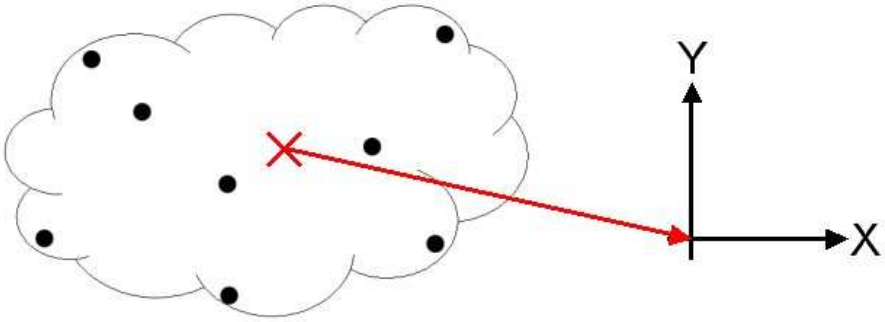


Abbildung 8.10: Transformation, Schritt 1: Translation des Mittelpunktes der Punktwolke der relevanten Punkte in den Koordinatenursprung

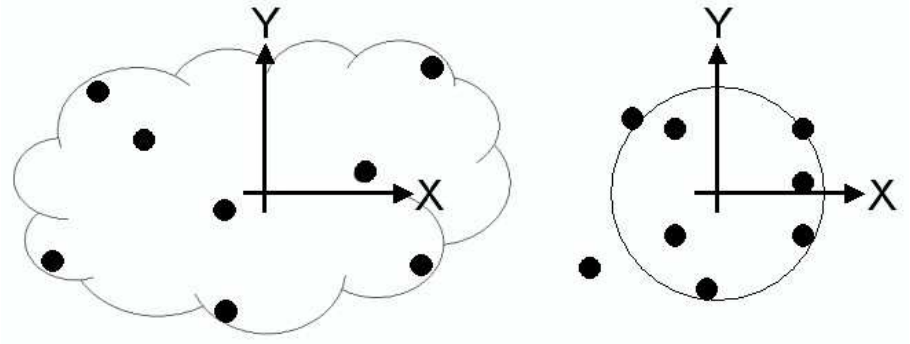


Abbildung 8.11: Transformation, Schritt 2: Skalierung der Punkte

Schritt 2: Skalierung

Der zweite Schritt ist eine Skalierung und wird in Abbildung 8.11 gezeigt. Der durchschnittliche Abstand der betrachteten Punkte vom Mittelpunkt muss auf $\sqrt{2}$ gebracht werden. Dadurch ist die Größenordnung der Koordinaten in etwa gleich.

8.3.4 Verbesserter Acht-Punkte-Algorithmus

Der gesamte Algorithmus läuft nun wie folgt ab:

- Zunächst bekommt man zwei Bilder mit $n \geq 8$ Paaren korrespondierender Punkte.
- Diese Punkte werden transformiert/normalisiert.
- Dann setzt man jedes Punktpaar in die Epipolargleichung ein und erhält so die Gleichungsmatrix A.
- Den Rang dieser Matrix muss man nun mittels Singulärwertzerlegung auf 8 bringen.
- Danach kann man die Gleichung $A\mathbf{f} = 0$ auflösen und erhält die Fundamentalmatrix F.
- Falls F den Rang 3 hat, muss man wieder mittels SVD den Rang auf 2 bringen.

8.3.5 Auswertung

Dieser Abschnitt soll zeigen, wie sich die Normalisierung beim Acht-Punkte-Algorithmus auswirkt. Außerdem wird der Acht-Punkte-Algorithmus mit einem iterativen Algorithmus verglichen.

Beispiel 1: Haus

Bei dieser Szene (8.12) fällt auf, dass die Epipole weit außerhalb der Bilder liegen. Bei der Abbildung 8.13 ist auf-

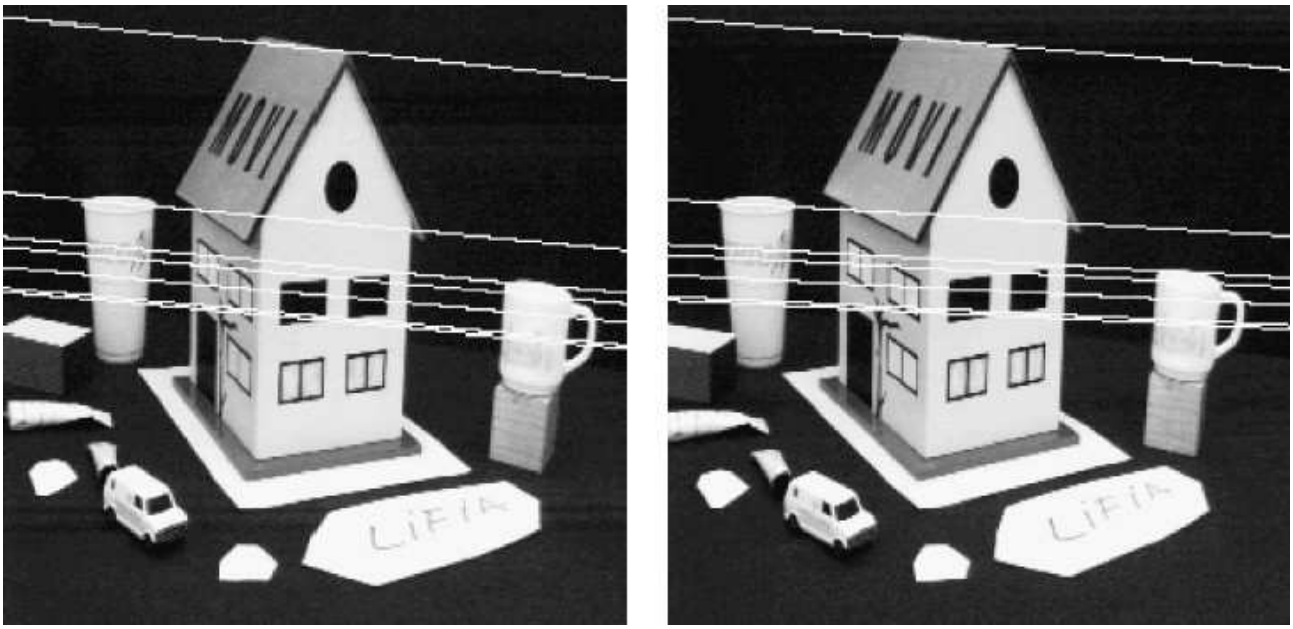


Abbildung 8.12: Beispiel 1: Haus; Epipole liegen weit außerhalb der Bilder

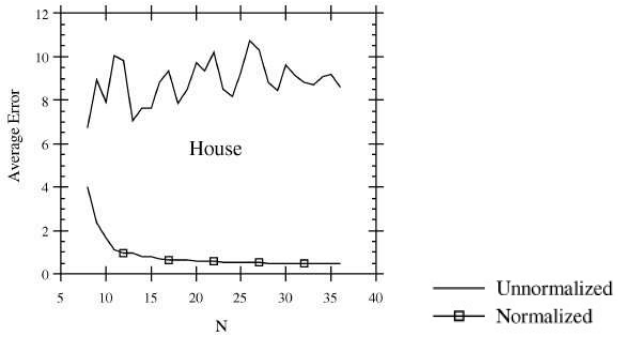


Abbildung 8.13: zu Beispiel 1; N: Anzahl der zufällig ausgewählten Paare von Punktkorrespondenzen in den Bildern; Average Error: Abstand des Punktes in Pixeln von der Epipolarlinie, über 100 Versuche je N gemittelt

Abbildung 8.14: Beispiel 2: Statue; Epipole liegen ein gutes Stück außerhalb der Bilder

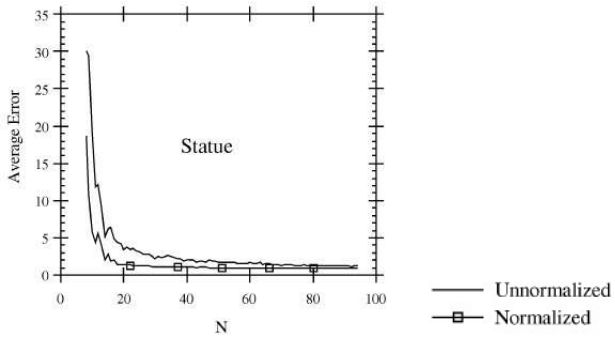


Abbildung 8.15: zu Beispiel 2; N: Anzahl der zufällig ausgewählten Paare von Punktkorrespondenzen in den Bildern; Average Error: Abstand des Punktes in Pixeln von der Epipolarlinie, über 100 Versuche je N gemittelt

gezeichnet, wie sich bei diesem Beispiel der Acht-Punkte-Algorithmus mit und ohne Normalisierung verhält. Auf der X-Achse ist die Anzahl der Paare von Punktkorrespondenzen notiert, die für die Berechnung der Fundamentalmatrix genutzt wurden. Für jede Anzahl - also für jedes N - wurde der Versuch 100 mal jeweils für den Acht-Punkte-Algorithmus mit und ohne Normalisierung durchgeführt. Dabei wurden die Paare von Punktkorrespondenzen jeweils zufällig gewählt. Auf der Y-Achse ist der durchschnittliche Fehler notiert, der angibt, wie viele Pixel sich ein Punkt von der Epipolarlinie weg befindet. Um das nochmal genau mit der Notation aus 8.2.2 zu formulieren: p_1 und die dazugehörige Epipolarlinie l_2 sind uns aufgrund der Epipolargeometrie bekannt. Der Fehler gibt nun an, wie viele Pixel weit weg sich der mit der errechneten Fundamentalmatrix rekonstruierte, korrespondierende Punkt p_2 von l_2 befindet.

Bei der Szene 8.12 liegt der Fehler für den unnormalisierten Acht-Punkte-Algorithmus teilweise bei zehn Pixeln und ist völlig unbrauchbar, beim der normalisierten Version nähert sich der Fehler einem halben Pixel!

Beispiel 2: Statue

Das zweite Beispiel ist die Szene 8.14. Auffällig ist nur, dass die Epipole ein gutes Stück außerhalb der Bilder liegen, aber nicht so weit wie beim ersten Beispiel. Abbildung 8.15 zeigt, dass der normalisierte Algorithmus zwar etwas besser ist, aber einen großen Effekt hat die Normaliserung nicht mehr. Beide liegen ab etwa 40 Punktepaaren bei einem Fehler von etwa ein bis zwei Pixeln. Allerdings wurden in [2] noch weitere Algorithmen mit dieser Szene getestet und durchweg liefern alle gute Ergebnisse.

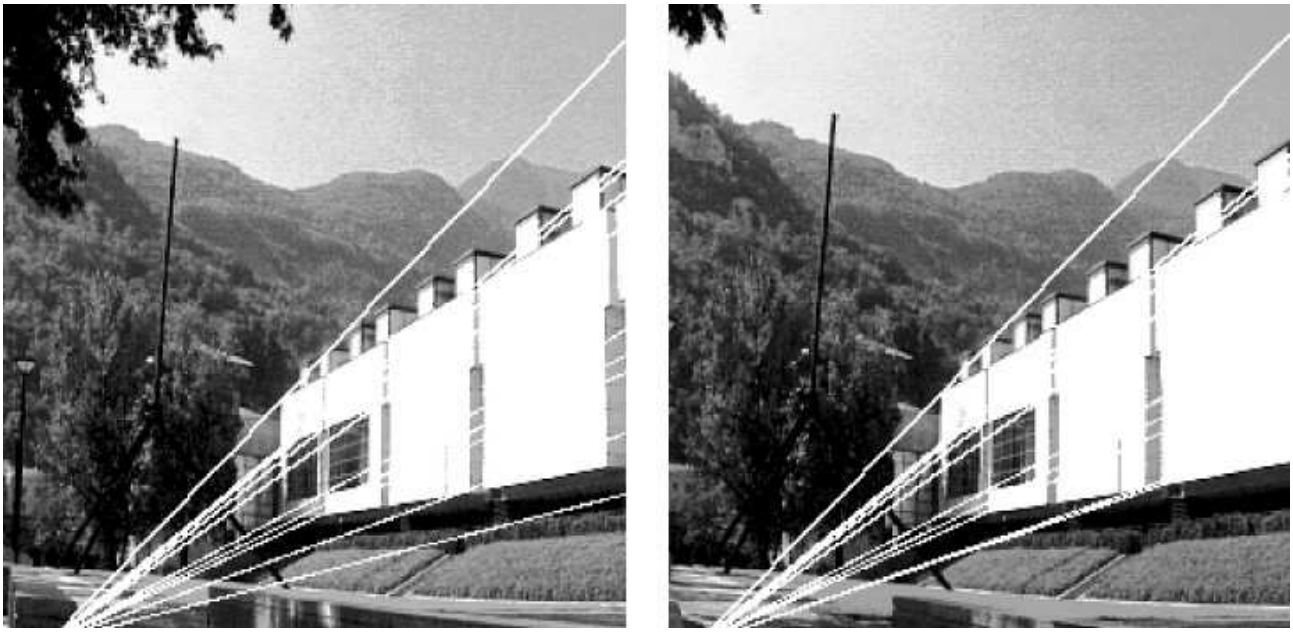


Abbildung 8.16: Beispiel 3: Museum; Epipole liegen nah an den Bildern; Punktkoordinaten sind eher ungenau

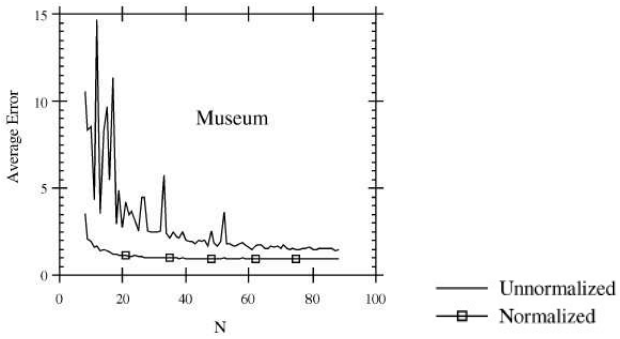


Abbildung 8.17: zu Beispiel 3; N: Anzahl der zufällig ausgewählten Paare von Punktkorrespondenzen in den Bildern; Average Error: Abstand des Punktes in Pixeln von der Epipolarlinie, über 100 Versuche je N gemittelt

Beispiel 3: Museum

Die dritte Beispielszene (8.16) zeigt ein Museum bei Grenoble. Diesmal liegen die Epipole nah an den Bildern und die Punktkoordinaten sind eher ungenau. Der Unterschied ist wieder recht deutlich, wie Abbildung 8.17 zeigt. Ab etwa zwölf Punktepaaren liefert der normalisierte Acht-Punkte-Algorithmus einen zwei-Pixel-Fehler, ohne Normalisierung sind die Ergebnisse erst ab 50 Punktepaaren langsam brauchbar.

Beispiel 4: Korridor

Beispiel vier zeigt eine Korridorszene (8.18). Hier liegen die Epipole in den Bildern und die Koordinaten der Paare korrespondierenden Punkte sind sehr genau bekannt. Die Normalisierung macht keinen großen Unterschied - der Fehler liegt nachher bei einem Viertel Pixel.

Beispiel 5: Kalibrierungsvorrichtung

Das letzte Beispiel ist eine Szene mit einer Kalibrierungsvorrichtung (8.20), bei der die Koordinaten der Paare korrespondierenden Punkte sind extrem genau bekannt sind.

Der unnormalisierte Acht-Punkte-Algorithmus kommt wie in Abbildung 8.21 ersichtlich, ganz gut mit einem Fehler von ein bis zwei Pixeln weg, aber die Normalisierung wirkt sich doch stark aus, denn der Fehler liegt hier schon bei wenigen

Abbildung 8.18: Beispiel 4: Korridor; Epipole liegen in den Bildern; Punktkoordinaten sind sehr genau bekannt

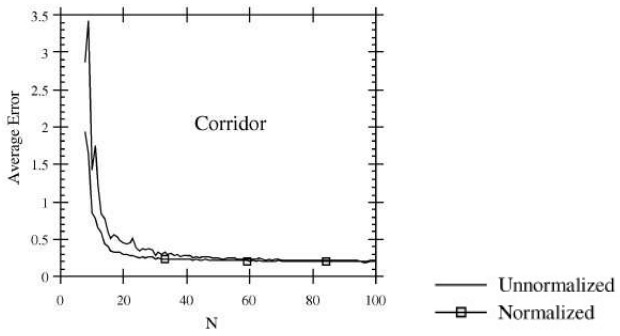
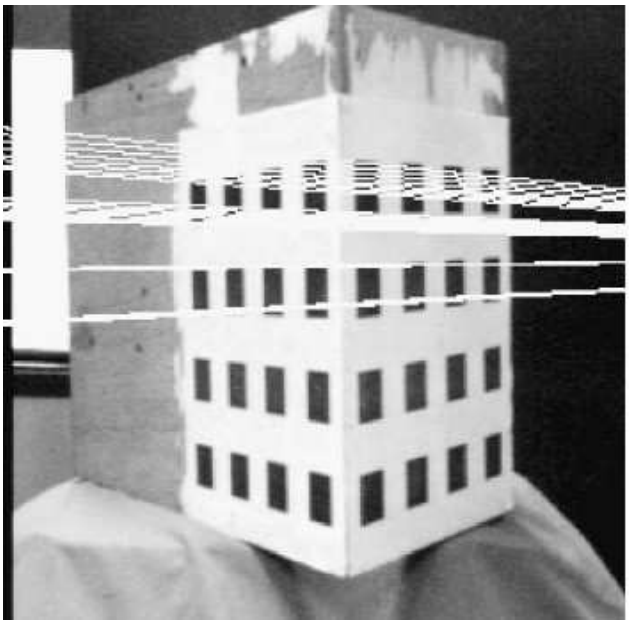


Abbildung 8.19: zu Beispiel 4; N: Anzahl der zufällig ausgewählten Paare von Punktkorrespondenzen in den Bildern; Average Error: Abstand des Punktes in Pixeln von der Epipolarlinie, über 100 Versuche je N gemittelt



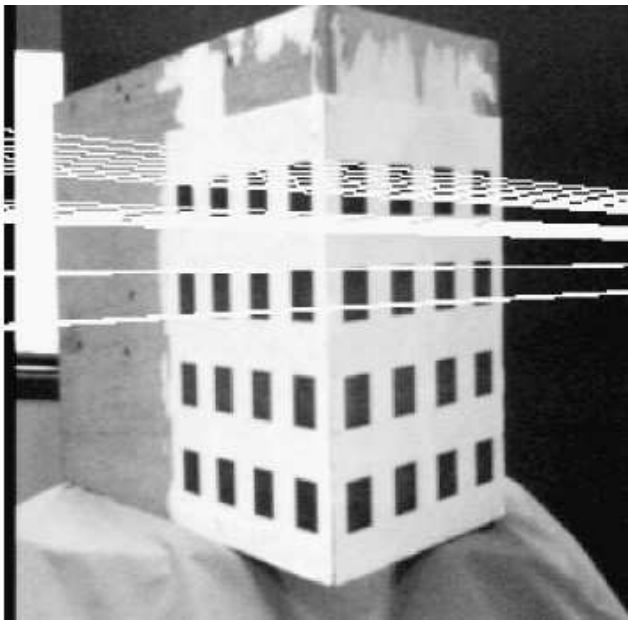


Abbildung 8.20: Beispiel 5: Kalibrierungsvorrichtung; Punktkoordinaten sind extrem genau bekannt

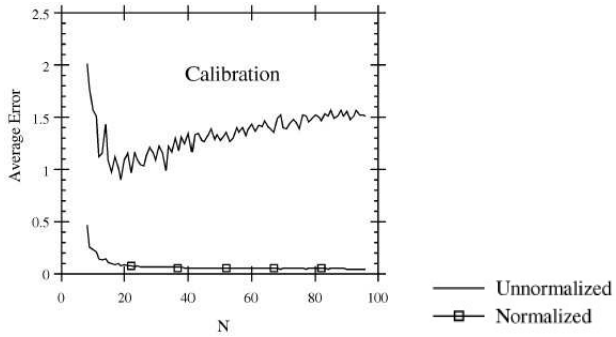


Abbildung 8.21: zu Beispiel 5; N: Anzahl der zufällig ausgewählten Paare von Punktkorrespondenzen in den Bildern; Average Error: Abstand des Punktes in Pixeln von der Epipolarlinie, über 100 Versuche je N gemittelt

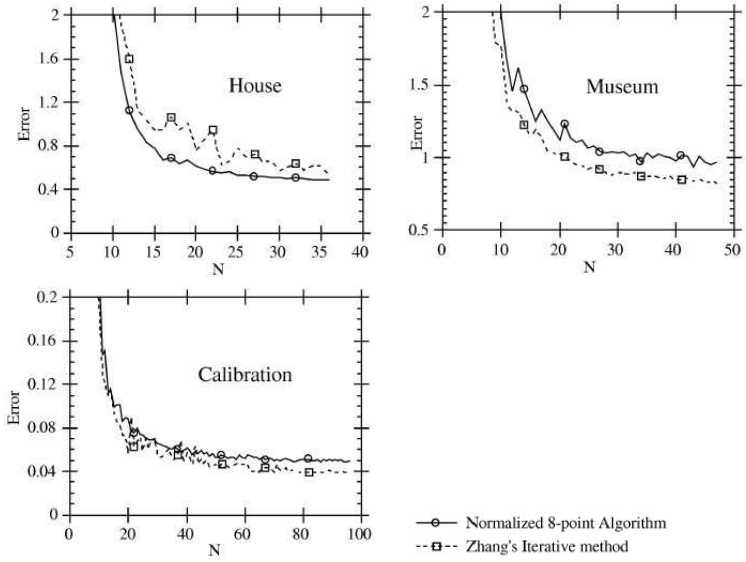


Abbildung 8.22: Vergleich: Normalisierter Acht-Punkte-Algorithmus - Zhangs Iterative Methode

Punkten bei einem Zehntel Pixel und darunter.

Zusammenfassend kann man also sagen, dass die Ergebnisse des normalisierten Acht-Punkte-Algorithmus immer besser sind, aber der Unterschied zwischen Acht-Punkte-Algorithmus mit und ohne Normalisierung auch von der betrachteten Szene abhängt.

Vergleich: Normalisierter Acht-Punkte-Algorithmus - Zhangs Iterative Methode

In Abbildung 8.22 ist der normalisierte Acht-Punkte-Algorithmus mit 'Zhangs Iterativer Methode' verglichen. Dieser ist wohl einer der besten Algorithmen, die es gibt, und man sieht, dass der Unterschied zum normalisierten Acht-Punkte-Algorithmus gering ist. Bei der ersten Szene mit dem Haus ist der Acht-Punkte-Algorithmus besser, beim Museum ist er schlechter und bei der Kalibrierungsszene ist er einen Tick schlechter. Bei den beiden anderen Szenen - Statue und Korridor - ist der Unterschied kaum messbar. Hier lieferte der Acht-Punkte-Algorithmus ohne Normalisierung bereits gute Ergebnisse.

Die beiden Algorithmen liefern also ähnlich gute Ergebnisse, aber der Acht-Punkte-Algorithmus mit Normalisierung tut das etwa 20 Mal so schnell, wie der Algorithmus von Zhang und ist dabei deutlich einfacher zu implementieren.

Literaturverzeichnis

[1] Nikolaus Augsten. *Robuste Schätzung der Fundamental Matrix*. ICG - Computer Graphics and Vision - TU Graz/Austria, 1998.

[2] Richard I. Hartley. In defense of the eight-point algorithm. *Pattern Analysis and Machine Intelligence*, 19(6):580–593, 6 1997.

[3] Richard I. Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000. Uni Koblenz Bib Signatur: INF 2002/7531 + INF 2002/8013.

[4] Quang-Tuan Luong and Olivier D. Faugeras. The fundamental matrix: Theory, algorithms, and stability analysis. *International Journal of Computer Vision*, 17:43–75, 1996.

[5] Dietrich Paulus. *Structure from Motion*. July 2003. Skript zur Vorlesung *Struktur aus Bewegung* im SS04.

[6] R. Schiedermeier. Projektionen. FH München, FB 07 Informatik/Mathematik, 1996.

Vortrag 9

Rekonstruktion von 3D-Oberflächen aus Punktwolken mittels Triangulierung

Jan Müller

January 29th 2004

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
fotn@uni-koblenz.de
http://www.uni-koblenz.de/~fotn

9.1 3D Rekonstruktion: zwischen Bildverarbeitung und Computergrafik

Die Rekonstruktion dreidimensionaler Oberflächen aus unterschiedlichsten Quellen ist ein sehr aktuelles Forschungsgebiet in der Datenverarbeitung. Vielzählige Branchen und Wissenschaften, wie etwa die Fahrzeugindustrie, aber auch die Medizin und Physik, sind daran interessiert und beteiligt.
Die Schwierigkeit liegt darin, aus Punkt-Datensätzen, die weder eine verlässliche Ordnung, noch zusätzliche Daten wie Normalen, also die Ausrichtung der Punkte, enthalten, lückenlos und optisch fehlerfrei glatte Oberflächen zu rekonstruieren. Solche Datensätze haben ebenso zahlreiche wie unterschiedliche Quellen: sie reichen von simplen Vermessungen, beispielsweise eines Architekten, und 'motion capture' Systemen, die anhand von Markern Bewegung und Form interpretieren, über 3D Scannern für kleine Objekte oder ganzen Personen, bis zu den Rekonstruktionsalgorithmen der modernen Bildverarbeitung.
Für die Bildverarbeitung schließt sich damit der Kreis, der eventuell damit begonnen hat dreidimensionale Objekte auf Bildfolgen zu identifizieren und zu markieren. So können Objekte verfolgt, Bewegung und Transformation nachvollzogen und schließlich auch ein Datensatz von Punkten gewonnen werden, die eine 3D-Rekonstruktion durch Annäherung möglich machen.
Das Arbeitsgebiet der Computergrafik hingegen beginnt erst richtig damit, aus 3D-Informationen Drahtgitter zu erstellen um diese zu schattieren und zu texturieren. So entstehen annähernd photorealistische Modelle dreidimensionaler Objekte, die wiederum beliebig in Lichtsimulationen, wissenschaftlichen Versuchen oder auch Computerspielen Verwendung finden.

9.1.1 Die Rekonstruktion von 3D Oberflächen

Im Folgenden betrachte ich die Grundlagen der Rekonstruktion dreidimensionaler Oberflächen aus beliebigen Punktwolken.
Solche Punktwolken stammen meist aus den bereits erwähnten 3D Scannern oder der Bilderkennung. Grundsätzlich messen solche Scanner in einem bestimmten Radius die Entfernung von Objekten und damit ihre Position im Raum: ein Laserscanner beispielsweise sendet hochfrequentes Licht aus und mißt die Zeit, die das reflektierte Licht benötigt, um zum Scanner zurückzukehren. Neben Informationen über die Entfernung wären so theoretisch auch Aussagen über die Beschaffenheit der Oberfläche, auf die das Licht gefallen ist, möglich, beispielsweise über dessen Reflektionseigenschaften.
Weiterhin kann man für ein Gerät dieser Bauart Annahmen über eine Ordnung der dreidimensionalen Punkte machen, da der Laser seine Umgebung in einer bestimmten Reihenfolge abscannt.
Hier betrachte Ich aber den allgemeinsten, abstraktesten Fall, in dem nichts über die Punkte unseres Datensatzes bekannt ist und alle Annahmen spekulativer Natur sind.

Das Szenario im Einzelnen: Als Eingabe dient ein Datensatz von dreidimensionalen Punkten; diese werden üblicherweise dargestellt als ein xyz-Vektor.

$$\begin{pmatrix} x \\ y \\ z \end{pmatrix}$$

Über diese Punkte sei nichts bekannt, außer dass sie zu einer Oberfläche gehören.
Die Zielsetzung ist es nun, aus diesem Datensatz zunächst ein Dreiecksgitter zu rekonstruieren und schließlich eine glatte Oberfläche, die die ursprüngliche Oberfläche mutmaßlich best möglich annähert. Dazu wird folgendermaßen vorgegangen:

- Rekonstruktion eines Drahtgitters unter Einbezug von Punktnachbarschaften
- Approximation der glatten Fläche mittels Parametrisierung
- Rückführung der Fläche

9.1.2 Gopi und Krishan: Rekonstruktionsalgorithmus für Dreiecksnetze aus Punktwolken

Ein effizienter Algorithmus zur Rekonstruktion eines Drahtgitters aus einer Punktwolke, wurde im Jahr 2000 von M. Gopi and S. Krishnan vorgestellt. Das zugehörige Paper trägt den Titel:

> 'A Fast and Efficient Projection Based Approach for Surface Reconstruction'

Dieser Algorithmus geht in einer Kettenreaktion vor: beginnend bei einem Startpunkt (der sogenannte Referenzpunkt: 'R') betrachtet er dessen Nachbarschaft als Parametergebiet. Darin sucht er nach Punktkorrespondenzen, die sich zu Dreiecken verbinden lassen, also triangulieren lassen, ohne eine Auswahl an Konsistenzkriterien, wie das übersteigen eines Maximalwinkels oder der Einschluss noch nicht triangulierter Punkte, zu verletzen. Die schliesslich triangulierten Vertices, die noch nicht von allen Seiten angeschlossen wurden, werden als nächstes zum Referenzpunkt, bis das Drahtgitter vollständig rekonstruiert wurde.

Der Aufbau der Algorithmus

Das Verfahren kennt 4 Zustände, die ein Punkt im Rekonstruktionsprozess einnehmen kann:

- 'free'
  Ein Punkt, der noch keine Zugehörigkeit hat. (zu keinem Dreieck gehört)
- 'fringe'
  Randpunkt, der betrachtet aber noch nicht trianguliert wurden.
- 'boundary'
  Randpunkt, der nicht an alle Nachbardreiecke angeschlossen wurde.
- 'completed'
  Der Punkt wurde an alle seine Nachbardreiecke angeschlossen. (wurde trianguliert)

Dazu ergeben sich zwei Invarianten, die in den Algorithmus einfließen:

1. Nur Punkte, die 'completed' sind, dürfen im Inneren eines Dreiecks liegen.
   Dies ergibt sich aus der propagierenden Natur des Algorithmus, der eine Punktnachbarschaft auf ein zweidimensionales Parametergebiet projeziert und dort immer direkte Nachbarn zu Dreiecken verbindet.
2. Nach einer Iteration wird der Referenzpunkt entweder 'completed' oder 'boundary'.
   Das heißt es muss entweder eine Verknüpfung zu einem Nachbarn erfolgen oder es wird eine Verletzung der Verknüpfungskriterien festgestellt, so dass der Punkt am 'Rand' des Drahtgitters bzw. an einem Loch im Dreiecksnetz liegt.

Obwohl, wie gesagt, alle Annahmen über das zu rekonstruierende Objekt eher spekulativer Natur sind, müssen zusätzlich einige Feststellungen gemacht werden. Ohne diese Einschränkungen wären wichtige Teile des Algorithmus nicht eindeutig terminierbar. Diese Einschränkungen lauten im Einzelnen:

- 'Das Objekt ist lokal uniform.'
  Das heißt, dass die Abstands-Differenz zwischen nähestem und entferntestem Nachbarn eines betrachteten Punktes durchgehend unterhalb einer zu bestimmenden Konstanten bleibt. Dies dient zur Einschränkung des Suchradius auf einen nicht nur endlichen, sondern auch über die Größe des zu rekonstruierenden Objektes lokal beschränkten Bereich.
- 'Das Kriterium der layer-Nähe ist erfüllt.'
  Der Abstand zwischen zwei Punkten unterschiedlicher Objektabschnitte ist größer, als der Abstand eines Punkt zu seinem direkten Nachbarn. Nur so kann das Objekt allgemeingültig in Unterbereiche unterteilt werden.
- 'Das Objekt ist smooth, sprich erfüllt die Vorraussetzungen für Glätte.'
  Das Kriterium hierfür ist, dass der Winkel zwischen zwei anliegenden Dreiecks-Normalen einen Maximalwert, im Fall dieses Algorithmus von 90°, nicht übersteigt.

Insbesondere die letzte Einschränkung auf 'smoothe' Objekte ist dabei als weitreichend anzusehen, da sie beispielsweise einen quadratischen Raum, in dem die Wände in Form von Dreiecken senkrecht aufeinander stehen, nur noch als Grenzfall erlaubt. Eine Erfassungsungenauigkeit, die den Winkel über ein senkrechtes Maß hinaus erhöht, würde bereits dazu führen, dass entsprechende Dreiecke potentiell nicht verbunden werden. Dennoch sind auch diese Einschränkungen

notwendig und dienen der Rekonstruktion glatter Oberflächen. Insbesondere organischen Objekten, aber auch Karosseriebauteilen aus der Fahrzeugindustrie, kommen diese Oberflächenbegrenzungen entgegen.

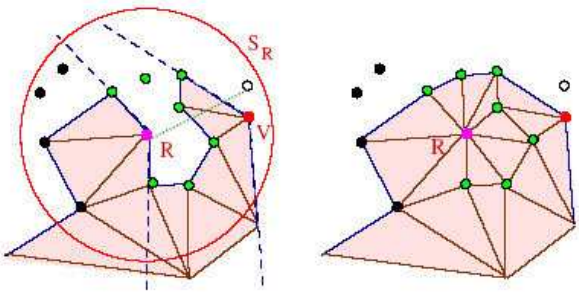
Der Algorithmus selbst ist in folgende vier Abschnitte aufgeteilt:

1. 'Initialisierung und Bucketing'
   Es wird eine sogenannte 'dexel'-Struktur verwendet, in die die 3D Punkte orthografisch abgebildet werden. Dabei fallen alle Punkte, die in der Projektion hintereinander liegen, in ein dexel-Feld und werden anhand ihres Tiefenwertes geordnet.
   Die Verwendung dieser bereits 1986 von T. Van Hook vorgestellte Ablagestruktur führt zu einer Ordnung der dreidimensionalen Punkte in quasi-zweidimensionalen Nachbarschaften. Man kann sich leicht vorstellen, dass eine Suche in den beiden Hauptdimensionen des dexel-Arrays unverzüglich zu einer Untersuchung der Nachbarschaft in der Ebene, in der der betrachtete Punkt liegt, führt. Gleichzeitig gehen auch Abstandsverhältnisse in die Tiefe nicht verloren und lassen sich beispielsweise durch einen 'bounding-box' Test nachvollziehen.

2. 'Beginn der Rekursion: Point Pruning'
   Die Nachbarkandidaten C_R des Referenzpunktes 'R' werden durch eine bounding box aus dem dexel Feld ermittelt.
   Die Dimension der box ergibt sich dabei aus den oben angesprochenen Abstandskriterien wie folgt:
   dim = μ * m , m = Abstand zum nähesten Nachbarn
   μ * m = Mindestabstand zum nähesten Punkt in einem anderen Layer (μ ist konstant)
   Auf gleiche Weise wird um R eine bounding sphere mit Radius (μ * m) gelegt, um die Kandidatenliste weiter zu verkürzen. Die Suche mit euklidischem Abstand wird erst nachträglich durchgeführt, da sie im Gegensatz zum bounding box Test, der sich die Sortierung der dexel-Struktur leicht zu Nutze machen kann, relativ rechenaufwändig ist und man die Abstandsberechnung auf möglichst wenige Nachbarn beschränken möchte.

3. 'Culling'
   Die so gefundenen Kandidaten werden in einen 2D Parameterbereich projeziert und dort nach ihrem Winkel zu R um R angeordnet. Dies geschieht folgendermaßen:
   statt eine Tangentenfläche zu R mit möglichst geringem Fehlerabstand zu allen Nachbarn zu ermitteln, wird die Normale der Fläche aus Effizienzgründen aus den Normalen vorhandener, anliegender Dreiecke gemittelt. Als Schwerpunkt der Tangentenfläche wird der Referenzpunkt selbst gewählt und eine Durchschnittsnormale gefunden, anhand derer die Ebene orientiert wird. Da von einem Winkelunterschied von weniger als 90° ausgegangen wird, können im Bereich der lokalen Nachbarschaft von R keine Punkt auf den selben 2D Punkt zusammenfallen und es entsteht eine hinreichende Projektion der Kandidaten um R.
   In diesem Projektionsraum erfolgt nun eine korrigierende Winkelanordnung um den zentralen Punkt R: In R wird ein lokales Koordinatensystem erzeugt; darin werden für jeden Quadrant einzeln die Kandidaten anhand von $sin^2(\theta)$ abgebildet;
   θ bildet dabei den Winkel zwischen der x-Achse des Koordinatensystems und dem Vektor zwischen R und dem Kandidatenpunkt P im zweidimensionalen Projektionsraum.
   Es werden daraufhin zwei Kriterien für die Sichtbarkeit bzw. Erreichbarkeit geprüft, um weitere Kandidaten auszuschließen und die tatsächlichen Anschlußpunkte auszumachen:

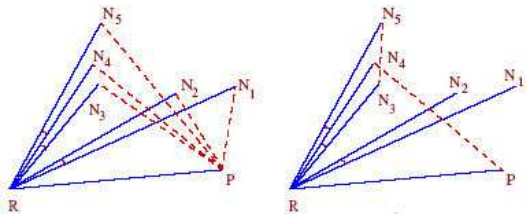
   - 'Das Visibilitätskriterium'
     Punkte, die von bestehenden Kanten verdeckt werden oder hinter den anliegenden Kanten des Referenzpunktes liegen, können ausgeschlossen werden. Denn Dreieckskanten im finalen Netz dürfen sich natürlich nicht schneiden.



   Um dabei nicht alle Kanten betrachten zu müssen, wurde folgendes Theorem aufgestellt: Nur sogenannte

Rand-Kanten zwischen Punkten C_R können, nach dem Entfernen der Punkte außerhalb des Sichtbereiches von R, noch Punkte im Kandidatenfeld C_R verdecken. Randkanten ist dabei als Gegensatz zu Innenkanten, also solchen die von weiteren Punkten und Kanten umgeben sind, zu verstehen.

- 'Das Winkelkriterium'
  Das Winkelkriterium dient zur Optimierung von Dreiecken mit sehr kleinem Winkel. Wie es in der Computergrafik allgemeingültig ist, sind besonders lange, spitzwinklige Dreiecke zu vermeiden.



Wird ein Minimalwinkel unterschritten, kann einer der Punkte offensichtlich entfernt werden. Dabei dürfen jedoch die Invarianten nicht verletzt und keine Punkte eingeschlossen werden.

4. Triangulierung
   Die restlichen Punkte in C_R werden der Reihenfolge nach mit R zu Dreiecken verbunden, also trianguliert. Wird dabei ein Maximalwinkel α überschritten, wird der Punkt nicht angeschlossen. Dieser Winkel beschreibt die Charakteristik von Löchern im Modell und R wird als Randpunkt angesehen.
   Abschließend werden alle freien Punkte in C_R als 'fringe' markiert, zur Warteschlange hinzugefügt und von dieser der nächste Referenzpunkt gewählt.
   Diese Rekursion erfolgt bis alle Dreiecke durch Triangulierung den Zustand 'completed' oder 'boundary' erreicht haben.

Der Rekonstruktionsalgorithmus von M. Gopi und S. Krishnan kann weiter spezialisiert werden, um z.B. Rauschen auf planaren Oberflächen auszugleichen. Dabei wird das dexel-Array als Polarwinkel-Feld aneinanderliegender Punkte angesehen und der Kandidatenradius für den nächsten Punkt so klein gehalten, dass hohe Frequenzen in der Oberfläche herausfallen.
Dies führt zu einer Glättung des Ungenauigkeits-Rauschens auf eigentlich ebenen (glatten) Oberflächen.

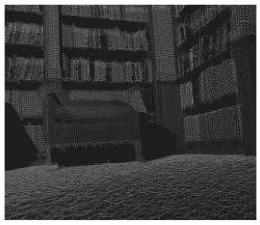


Figure 5: Microfacets showing the noise even on the planar floor of the room.

9.2 Annäherung glatter Oberflächen durch Parametrisierung

Um aus einer so gewonnenen Triangulierung eine glatte Oberfläche anzunähern, müssen die Verhältnisse unter den durch die Triangulierung gewonnen Punkte ein weiteres Mal untersucht werden. Dazu werden die 3D Punktdaten abermals in eine 2D Darstellung überführt, in der Nähe- und Winkelabhängigkeiten unter ihnen besser untersucht werden können. Diesen Vorgang, einen Datensatz in eine Darstellung geringerer Dimension zu projezieren, nennt man auch Parametrisierung.
Eis klassisches Einsatzgebiet der Parametrisierung ist zum Beispiel die Erzeugung von Texturkoordinaten in der Computergrafik.
Wichtige Terminologien des Verfahrens wurden dabei auf Basis der leicht zu beobachtenden 1D Abbildung zweidimensionaler Kurven festgelegt. Grundsätzlich kann man drei verschiedene Arten der Parametrisierung von Kurven unterscheiden:

- uniforme Parametrisierung
  Die Punkte werden einfach anhand ihrer Anzahl auf einen Bereich, normalerweise [0;1], abgebildet. Im 2D Kurvenfall wäre dabei beispielsweise der Punkt [0,y] der erste in der 1D Parametrisierung, der Punkt [1,y] der letzte und alle dazwischen würden mit uniformen Abständen über die Parametrisierung verteilt werden.

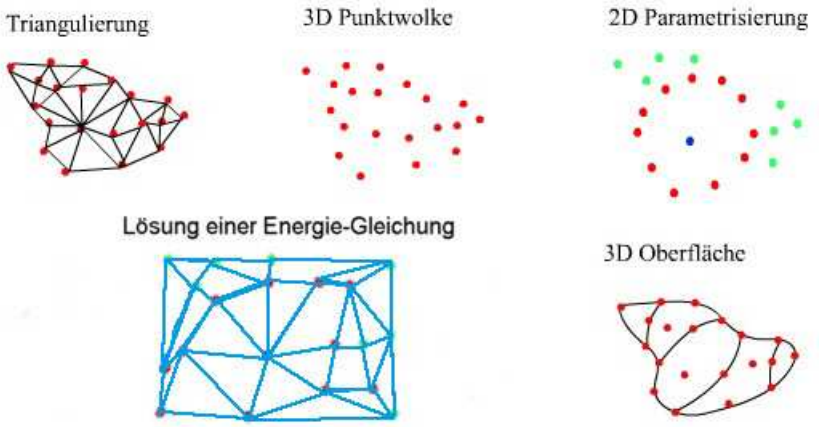
- chordale Parametrisierung
  Die chordale Parametrisierung ist das am häufigsten verwendete Verfahren, da der Einbezug der Kantenlänge in die Verteilung gute Ergebnisse hervorbringt.

$$p_k = p_{k-1} + \frac{\|P_k - P_{k-1}\|}{d}$$

  wobei d = Gesamtbreite des Polygons, Wertebereich [0,1]
  Im 2D Kurvenbeispiel hieße das, dass die Entfernungen der Kurvenpunkte untereinander in die Parametrisierung und damit ihrer eindimensionalen Position einginge.

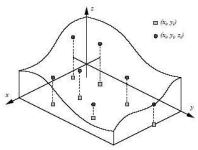
- zentripedale Parametrisierung
  Auch hier wird die Kantenlänge im Verhältnis zur Gesamtlänge eingerechnet, allerdings mit einer geringeren Gewichtung, was zu einer verbesserten Kurvenabbildung führt.

In Fall der Oberflächenglättung betrachten wir allerdings ein dreidimensionales Objekt, aus dem ein begrenzter Bereich lokal ins Zweidimensionale abgebildet wird. Gesucht wird folglich für jeden Datenpunkt P_i im R^3 ein Parameterwert p_i im R^2, so dass die Topologie des Dreiecksnetzes erhalten bleibt. D.h. die späteren Dreiecke im Parametergebiet dürfen sich nicht überlappen und die Reihenfolge muss erhalten bleiben.
Um dies zu gewährleisten sollte das Parametergebiet eine konvexe Fläche sein, was aber dem Wunsch nach einer optimalen Darstellung der 3D-Nachbarschaftsverhältnisse widerspricht. Es ist einleuchtend, dass eine konvexe Fläche insbesondere extreme Winkelverhältnisse zwischen den Dreiecken nicht abbilden kann und auch die Abstandverhältnisse nicht immer dargestellt werden können.



Eine allgemeingültige und optimale Parametrisierungsmethode gibt es daher nicht. Die in der Literatur beschriebenen Verfahren sind immer nur für spezielle Ausgangssituationen ideal. Das Ergebnis ist ein 'trade-off' zwischen Rechenzeit und Genauigkeit des Verfahrens, je nach Ausgangssituation:

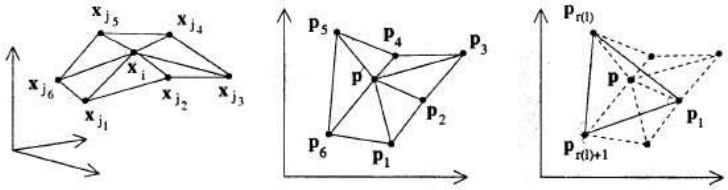
- Die Raumpunkte lassen sich in Teilbereiche unterteilen, so dass sich jeder dieser Bereiche bijektiv auf eine Fläche projezieren lässt. Damit hat man in jedem Teilbereich den Fall einer funktionalen Fläche, die sich ohne Verlust parametrisieren lässt.



- Man bedient sich zuvor eines Zwischenschrittes, nämlich der Vernetzung der Datenpunkte, z. B. einer Triangulierung. Durch die Nachbarschaftsinformationen, die über die Kanten der Vernetzung gegeben sind, können allgemeine Abbildungen der Datenpunkte auf ein Parametergebiet definiert werden, die keine Projezierbarkeit der Messpunkte voraussetzen.

Im Folgenden werden einige Parametrisierungsverfahren vorgestellt, die Triangulierung nutzen um Nachbarschaftsinformationen zu erlangen.

- 'shape-preserving' Parametrisierung [Floater 98]
  Wie auch bei den weiteren Verfahren wird davon ausgegangen, dass die anzunähernden Punkte bereits trianguliert sind.
  Es wird ein Parametrisierungsgebiet gewählt, das bei Floater zu einer Kreisscheibe homöomorph sein muss. Dieses wird in konvexe Form gebracht, indem vier extreme Randpunkte zu einem Einheitsquadrat aufgespannt werden.
  Davon ausgehend werden die Innenpunkte auf die Ebene projeziert und trianguliert:
  Zu jedem inneren Punkt P_i^3 wird eine Ausgleichsebene angelegt, die einen möglichst geringen Fehler zu P_i und k Nachbarn hat. Die Parameterwerte zu P_i werden mittels Konvexkombination dreier seiner Nachbarpunkte ermittelt. Selbstverständlich werden die ermittelten Parameter dabei nicht insitu, sondern werterhaltend in eine neue Ablagestruktur gespeichert. Unter allen Konvexkombinationen zur Berechnung des geglätteten Oberflächenpunktes wird diejenige gewählt, die den geringsten Fehler in der Approximation der Oberfläche liefert.
  Konvexkombination: $\boldsymbol{p} = \lambda_0 A + \lambda_0 B + \lambda_0 C$



  So erhält man eine Projektion beliebiger Oberflächen in eine Parameterebene unter Berücksichtigung der Abstandverhältnisse. Extreme Nachbarschaftsverhältnisse werden ausgeschlossen und das entstehende Drahtgitter gleicht sich automatisch, durch lokale Optimierung, zu einer glatten Oberfläche aus.

- 'Federenergie-Minimierung' [Greiner98]
  Der Federenergieansatz ist wohl der verbreiteste Relaxationsansatz und basiert auf den physikalischen Eigenschaften von Federn die, zu einem Netz aufgespannt, dazu neigen energetische Spannungen zu minimieren. Die Randpunkte des Parametrisierungsbereiches werden wiederum auf eine Ebene projeziert. Die Kanten zwischen den Innenpunkten werden als Federn mit gleicher Federkonstante aufgefasst. Dabei können die physikalischen Modelle der Federn natürlich beliebig komplex dargestellt werden, um eine möglichst genaue Energiesimulation zu gewährleisten. Die Rangpunkte werden fixiert, so dass sich das Netz 'auspendeln' kann. Dies wird über die Berechnung der Minimierung der Federenergie erreicht, wobei auch die Abstände zwischen den Fixpunkten (Längen der Federn) anteilhaft eingehen sollten. Das Verfahren geht wie folgt vor:

  1. Randpunkte finden und fixieren
     Die Fixierung der Randpunkte, wie sie bereits im 'shape-preserving' Verfahren durchgeführt wurde, um das Projektionsfeld konvex aufzuspannen, führt zu einem Widerspruch;
     einerseits möchte man eine möglichst genaue Darstellung von Entfernung und Lage der Punkte erzielen, andererseits muss die Form der zweidimensionalen Darstellung konvex sein, damit es nicht zu Überschneidungen der resultierenden Dreieckskanten kommen kann. Dennoch ist es offensichtlich, dass eine solche Verformung eine ideale Glättung der finalen Oberfläche im 3D von vornherein verhindert.

  2. Minimierung des Energiefunktionals
     Die Entspannung der Federn erfolgt durch die Minimierung des linearen Gleichungssystems, das neben den Federeigenschaften die Abstandverhältnisse, bzw. die quadratische Länge der Kanten, darstellt.

$$E = \frac{1}{2} \sum_{i,j} c_{ij} ||P_i - P_j||^2$$

     Dabei entspricht c_ij den Federeigenschaften, die auf sehr unterschiedliche Weise ermittelt und mehr oder minder komplex sein können. z.B.:

$$c_{ij} = \frac{1}{4}(\cot \alpha + \cot \beta)$$

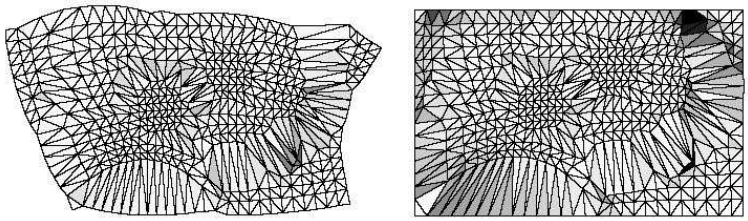
  Es gibt diverse Varianten dieses Verfahrens, die grundsätzlich die selbe Strategie verfolgen: zunächst werden Randpunkte gefunden und der beobachtete Bereich in einer konvexen Form fixiert; hierauf wird die Energie zwischen den Innenkanten minimiert und so das Federsystem entspannt.

- 'MIPS' [Greiner98]
  Die sogenannte Möglichst Isometrische Parametrisierungs Strategie ist ein eher junges Verfahren, vorgestellt vom

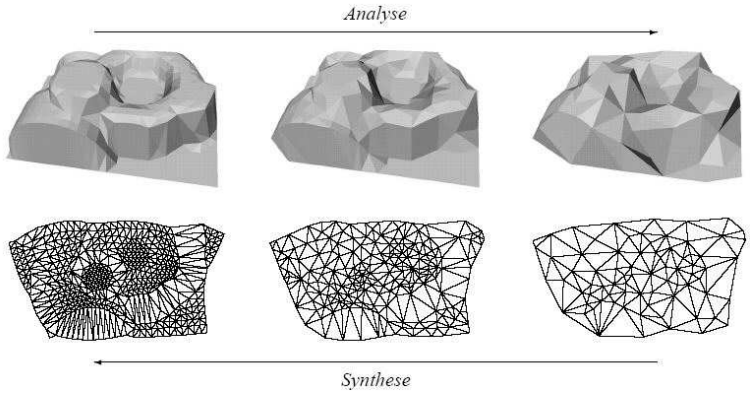
erlanger Professor Dr. G. Greiner.
Dabei wird das Augenmerk auf eine möglichst geringe Verzerrung in der Abbildung gelegt, indem Innen- und Randpunkte auf gleiche Weise behandelt werden. Die Einsparung einer Randpunktfixierung ermöglicht eine wesentlich natürlichere Glättung der gesamten Oberfläche. Allerdings entsteht dabei ein quadratisches, nicht wie bisher lineares Gleichungssystem, das es zu lösen gilt: Man finde die teil-lineare Funktion 'F', dargestellt durch Matrix 'A', die einen Oberflächenpunkt im R^3 isometrisch auf seinen Parameterwert im R^2 abbildet. Es wird nun die Konditionszahl der Funktion/Matrix betrachtet:

$$cond(A) = \left\|A^{-1}\right\| \cdot \|A\|$$

Die Konditionszahl ist ein gutes Qualitätsmaß für die Abbildung, wie für die algebraische Stabilität der Lösung, da sie effizient zu ermitteln ist, beispielsweise mittels Singulärwertzerlegung und dazu unabhängig ist von Translation, Skalierung sowie orthogonaler Transformation. Die Güte der Abbildung wird anhand der Summe der Einzelverzerrungen bewertet, die es zur Entspannung der Energieverhältnisse zu minimieren gilt.
Im linken Bild ist das formerhaltende MIPS Verfahren zu sehen, während im rechten Bild ein herkömmliches Verfahren mit Randpunktfixierung benutzt wurde. Je dunkler ein Abschnitt markiert wurde, desto größer ist der Fehler:



Hierarchische Methoden:
Alle vorgestellten Verfahren haben das Problem, dass sie um so schlechter konditioniert sind, je größer das Verhältnis von Innen- zu Randpunkten ist. Daher kann man sowohl einen Geschwindigkeits- wie auch einen Stabilitätsvorteil erzielen, indem man das Drahtgitter abstrahieren und gewonnene Daten für eine genauere Approximation wiederverwenden.



Rückführung der Fläche

Schließlich muss eine Rückführung der Flächen erfolgen; die rekonstruierte Oberfläche muss im 3D zu einem Gesamtobjekt fusioniert werden. Oftmals soll dabei die Form der Oberfläche nicht durch Drahtgitter, sondern beispielsweise durch mathematische Kurven wie Splines beschrieben werden. Der Prozess der Flächenrückführung sei hier nur grob zusammengefasst: Es seien Basisfunktionen B_j definiert, beispielsweise B-Splines zur Beschreibung der Oberfläche. In deren Funktionsraum F_B = span(B_j) wird eine Fläche gesucht, die der erzeugten Parametrisierung entspricht.

$$F = \sum\nolimits_j c_j B_j : \Omega \rightarrow R^3, F(p_i) \approx P_i$$

Es wird also über den gesamten Abbildungsraum der Darstellungskurve, in diesem Fall von B-Splines, eine Abbildung gesucht, die unsere rekonstruierte und geglättete Oberfläche bis auf einen zu vernachlässigenden Fehler approximiert. Um unerwünschte Krümmungen auszuschliessen, können verschiedene Interpolationsverfahren eingesetzt werden, die zusätzlich mit Glättungsoperatoren kombiniert werden.

- 'glatte Interpolation'
  Dabei werden zunächst die triangulierten Eingangsdaten geglättet. Nach der Parametrisierung der 3D Punkte erfolgt ein Remesh des Drahtgitters, bei dem die Daten anhand einer uniformen Verteilung eingepasst werden. Durch Spline-Interpolation wird das gleichmässige Kurvengitter dargestellt.
- 'glatte Approximation'
  Die Fläche wird z.B. unter Einbezug einer Glättungsfunktion mit Lawsons Algorithmus approximiert; es wird iterativ ein lineares System zur Minimierung des maximalen Fehlers gelöst:

$$B^t W_k B_{c_{k+1,i}} = B^t W_k P$$

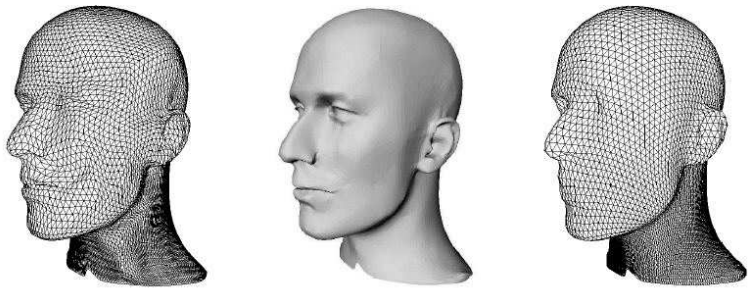
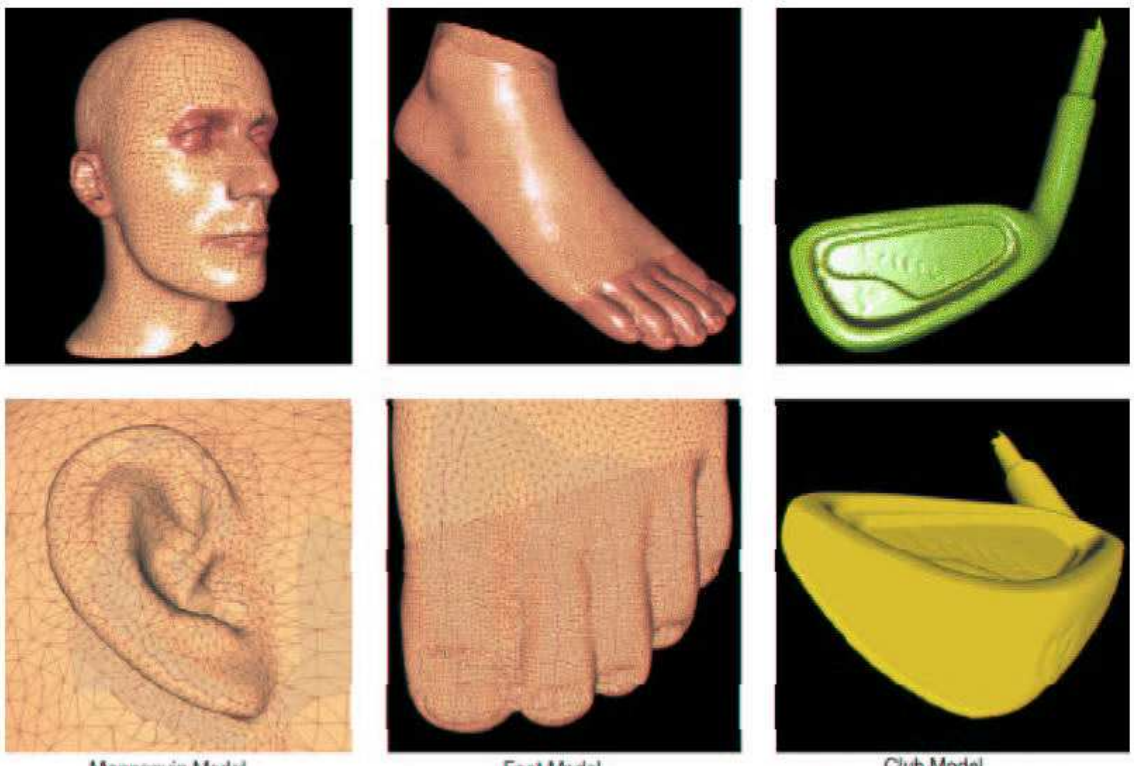


Bild 48: Remesh des Kopfdatensatzes (Mitte) mit chordaler (links) und MIPS (rechts) Parametrisierung.



Literaturverzeichnis

[1] M. Gopi and S. Krishnan. A fast and efficient projection based approach for surface reconstruction. In *SIBGRAPI02*, 2000/2001.

[2] Dr. G. Greiner. Rekonstruktion aus punktwolken, a2. In *SFB 603*, pages 86–129, 1998.

[3] Philipp Wagner. *Flächenrekonstruktion aus Finite-Elemente Netzen*. Stuttgart, 2003. Diplomarbeit, FH Stuttgart, SS 2003.

Vortrag 10

QuickTime VR

Andrea Fürsich

05.Februar 2004

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
fuersich@uni-koblenz.de
http://www.uni-koblenz.de/~fuersich

Dieser Artikel widmet sich einem Softwareprodukt der Fima Apple Inc. mit dem Namen QuickTime VR (QTVR). Dabei handelt es sich im wesentlichen um eine Erweiterung der bestehenden QuickTime-Technologie. Sie wurde in die QuickTime-Version 3.0 integriert und erstmal 1994 auf einer Fachmesse in Cannes vorgestellt. Informationen zu dieser Ausarbeitung wurden vor allem dem Artikel „QuickTime VR - An Image-Based Approach to Virtual Environment Navigation“von *Shenchang Eric Chen* [2] und den Ausführungen in „Plenoptic Modeling: An Image-Based Rendering System“von *Leonard McMillan und Gary Bishop* [4] entnommen.

10.1 Einleitung

Um etwas Gesehenes dauerhaft festhalten zu können, so dass auch andere es zu einem späteren Zeitpunkt betrachten können, gibt es verschiedene Möglichkeiten. Die einfachste Lösung ist, von der Szene ein Foto aufzunehmen. Das so entstandene Bild kann aber nur einen begrenzten Ausschnitt von dem zeigen, was in der Realität zu sehen ist. Die Szene ist für den Betrachter auf eine bestimmte Blickrichtung (*view direction*) und einen bestimmten Betrachterstandpunkt (*viewpoint*) beschränkt.
Eine bessere Möglichkeit Gesehenes für andere sichtbar zu machen bietet die Aufnahme eines Films. Hierbei werden dem Betrachter mehrere Standpunkte und Blickrichtung vermittelt, die sich aus der Kamerabewegung während der Aufnahme ergeben. Für den Betrachter ist die Bewegung in der Szene also fest vorgegeben.
Optimal wäre es jedoch, wenn ein völlig freies Bewegen in der Szene möglich wäre d.h. sowohl Standpunkt als auch Blickrichtung frei wählbar wären. Das zu verwirklichen ist die grundlegende Idee, die sich hinter QTVR verbirgt, nämlich es dem Benutzer zu ermöglichen, sich in einer virtuellen Umgebung umsehen und bewegen zu können.

Der folgende Abschnitt bildet das grundlegende Fundament, das für die Erzeugung neuer Bilder aus gegebenen Aufnahmen (das sog. *Image-Based Rendering*) notwendig ist. Anschließend wird die Umsetzung dieses Ansatzes als *Panorama Movie* in QTVR behandelt. Außerdem werden die QTVR-Komponenten *Objekt Movie* und *Zooming* beschrieben.

10.2 Beschreibung einer Szene durch die Plenoptische Funktion

Adelson und Bergen [1] prägten als erste den Begriff *plenoptische Funktion*. Er ist abgeleitet vom lateinischen *plenus* (= *ganz* oder *vollständig*) und *optisch*. Die plenoptische Funktion (10.1) beschreibt die Gesamtheit aller Lichtstrahlen in einer Szene völlig ohne explizites Wissen über die vorliegenden Geometrien. Sie ist in einem 5-dimensionalen Parameterraum definiert: Der Betrachterstandpunkt kann in jedem Raumpunkt (V_x, V_y, V_z) positioniert werden. Die Blickrichtung wird durch die Winkel θ in horizontaler Richtung und ϕ in vertikaler Richtung festgelegt (Abb. 10.1).

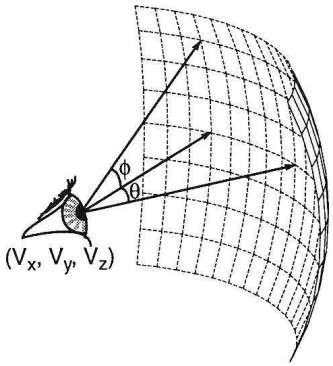


Abbildung 10.1: Anschauliche Deutung der plenoptischen Funktion

$$p = P(V_x, V_y, V_z, \theta, \phi) \tag{10.1}$$

10.3 QuickTime VR Panorama Movies

Die im vorherigen Abschnitt beschriebene Art der Darstellung einer Szene wird in QTVR durch die Panorama Movies umgesetzt. Panorama Movies sind letztlich Panoramabilder, in denen der Benutzer dynamisch agieren kann. Die freie Wählbarkeit der 5 Parameter der plenoptischen Funktion wird hier dahingehend beschränkt, dass eine Szene von einem festen Standpunkt aus betrachtet wird, die Parameter V_x, V_y, V_z sind also konstant. Bei der Wahl der Blickrichtung werden Bewegungen in horizontaler (360°) und nur eingeschränkt in vertikaler (<180°) Richtung zugelassen. Zur Vereinfachung wird für die anschließenden Berechnungen ϕ auch als fest gewählt. Somit bleibt als einziger freier Parameter θ, der Drehwinkel in horizontaler Richtung.
Als Repräsentationsform einer Szene ergibt sich somit ein *zylindrisches Panorama* (Abb. 10.2).

Abbildung 10.2: Zylindrisches Panorama

10.3.1 Erstellung eines zylindrischen Panoramas

QTVR ermöglicht es ein solches Panorama aus mehreren Einzelaufnahmen zu erstellen. Dazu müssen zunächst mehrere Fotos gemacht werden. Eine, auf einem Stativ befestigte Kamera wird dafür einmal komplett um die vertikale Achse gedreht. Optimal wäre hierbei, wenn sich die Rotationsachse genau im Brennpunkt der Kamera befindet (Abb. 10.3)

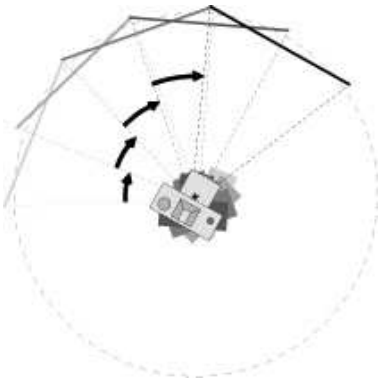


Abbildung 10.3: Aufnahme der Einzelbilder

Nach der Aufnahme und Digitalisierung liegt eine Serie von Bildern, genauer eine Serie von planaren Projektionen einer Szene vor. Diese gilt es nun zu einer einzigen zylindrischen Projektion zusammenzufügen. Der Vorgang des Zusammenfügens wird als *Stitching* bezeichnet. Wichtig hierfür ist, dass sich die Bilder in einem bestimmten Bereich überlappen, d.h. dass zu einem bestimmten Anteil in zwei Bildern die gleiche Information enthalten ist.

Entwicklung eines Mathematischen Modells, das die Projektionsabbildung der Kamera beschreibt

Zwei beliebige planare perspektivische Projektionen (Bilder) $\boldsymbol{I}$ und $\boldsymbol{I'}$ einer Szene von einem gemeinsamen Blickpunkt stehen durch eine homogene Transformation miteinander in Beziehung (10.2).

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{10.2}$$

$$x' = \frac{u}{w} \qquad\qquad y' = \frac{v}{w} \tag{10.3}$$

Dabei sind x und y die Pixelkoordinaten in Bild $\boldsymbol{I}$, x' und y' (10.3) die Pixelkoordinaten in Bild $\boldsymbol{I'}$. Damit können alle erlaubten Kamerabewegungen beschrieben werden.
Diese homogene Transformation $\boldsymbol{H}_{i+1}$ existiert für jedes benachbarte Bildpaar $\boldsymbol{I}_i$ und $\boldsymbol{I}_{i+1}$ aus der Serie $\{\boldsymbol{I}_0, ..., \boldsymbol{I}_{N-1}\}$ bestehend aus N Bildern und bildet einen Bildpunkt mit den Koordinaten $p_{i+1} = (x, y)$ in Bild $\boldsymbol{I}_{i+1}$ auf denselben Bildpunkt in Bild $\boldsymbol{I}_i$ mit den dortigen Koordinaten $p_i = (x', y')$ ab.

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \boldsymbol{H}_{i+1} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{10.4}$$

Wenn die Abbildung des gesamten Bildes $\boldsymbol{I}_{i+1}$ auf die Bildebene des Bildes $\boldsymbol{I}_i$ gemeint ist, schreibt man abkürzend:

$$\boldsymbol{I}_i = \boldsymbol{H}_{i+1}\boldsymbol{I}_{i+1} \tag{10.5}$$

Wie wir wissen unterscheiden sich zwei beliebige Bilder also nur durch die Drehung um die vertikale Achse, alle anderen Kameraparameter bleiben bei einem Rundumschwenk konstant. Die Tansformation $\boldsymbol{H}_i$ lässt sich deshalb in zwei Teile aufspalten, wobei der eine Teil die Rotation $\boldsymbol{R}_i$, also die extrinsische Transformation, beschreibt und der andere Teil die unveränderlichen Kameraparameter, die intrinsische Tranformation $\boldsymbol{S}$. Der intrinsische Anteil $\boldsymbol{S}$ wird Strukturmatrix der Kamera genannt und ist für alle $\boldsymbol{H}_i$ gleich. Durch die Zerlegung von $\boldsymbol{H}_i$ in

$$\boldsymbol{H}_i = \boldsymbol{S}^{-1}\boldsymbol{R}_i\boldsymbol{S} \tag{10.6}$$

werden die Projektions- und Rotationsanteile der homogenen Transformation entkoppelt. Die Rotation besteht nur aus einer Rotation um die y-Achse und kann beschrieben werden durch die Matrix:

$$\boldsymbol{R}_y = \begin{bmatrix} cos\theta & 0 & sin\theta \\ 0 & 1 & 0 \\ -sin\theta & 0 & cos\theta \end{bmatrix} \tag{10.7}$$

Wenn $\boldsymbol{S}$ bekannt ist, so genügt zur Bestimmung von $\boldsymbol{H}_i$ die Kenntnis des Rotationswinkels θ_i.

Bestimmung des Rotationswinkels θ_i

Um den Rotationswinkel θ_i zu bestimmen wird von Kanatani [3] eine lineare Approximation für infinitesimale Rotationswinkel durch Abschätzung der cos-Terme durch $1 + O(\theta^2)$ und der sin-Terme durch $\theta + O(\theta^3)$ vorgeschlagen, woraus sich folgende Abschätzung für x' und y' ergibt:

$$x' = x - f\theta - \frac{\theta(x - C_x)^2}{f} + O(\theta^2) \tag{10.8}$$

$$y' = y - \frac{\theta(x - C_x)(y - C_y)}{f} + O(\theta^2) \tag{10.9}$$

Diese Gleichungen zeigen, dass Drehungen um kleine Winkel für Pixel nahe der optischen Achse durch eine Translation angenähert werden können. Umgekehrt folgt daraus, dass man aus der Translation von Pixeln eine Approximation der Rotationswinkel erhalten kann (Abb.10.4). Damit gilt für die Drehwinkel:

$$tan\theta_i = \frac{t_i}{f} \qquad \text{bzw.} \qquad \theta_i = arctan\frac{t_i}{f} \tag{10.10}$$

Abbildung 10.4: Annäherung der Rotation durch eine Translation

Die Verschiebung t_i von einem Bild zum nächsten kann effizient durch eindimensionale Optimierung des Korrelationskoeffizienten der Bildmitten berechnet werden. Für die Näherung von f und θ macht man sich das Wissen zu Nutze, dass die Summe aller Winkel im Kreis 2π beträgt.

$$\sum_{i=0}^{N-1} \theta_i = 2\pi \tag{10.11}$$

Unter Verwendung von Gleichung (10.10) erhalten wir:

$$\sum_{i=0}^{N-1} \theta_i = \sum_{i=0}^{N-1} arctan\frac{t_i}{f} = 2\pi \tag{10.12}$$

Aus dieser Gleichung kann eine numerische Aufgabenstellung gewonnen werden, die Berechnung der Nullstelle von F. F erhalten wir durch Umformung von (10.12):

$$F(f) := 2\pi - \sum_{i=0}^{N-1} arctan\frac{t_i}{f} = 0 \tag{10.13}$$

Mit Hilfe des *Newton*-Verfahrens, das nach einigen Iterationen konvergiert, kann f gefunden werden. Damit kann auch θ und somit die Matrix $\boldsymbol{R_y}$ bestimmt werden.

Bestimmung der Strukturmatrix $\boldsymbol{S}$

Die Strukturmatrix $\boldsymbol{S}$ kann weiter zerlegt werden in:

$$\boldsymbol{S} = \boldsymbol{\Omega}_x \boldsymbol{\Omega}_z \boldsymbol{P} \tag{10.14}$$

$$\boldsymbol{P} = \begin{bmatrix} 1 & \sigma & -C_x \\ 0 & \rho & -C_y \\ 0 & 0 & f \end{bmatrix} \tag{10.15}$$

Dabei wird $\boldsymbol{P}$ die Projektionsmatrix genannt. Die einzelnen Parameter haben folgende Bedeutung:

- C_x, C_y sind die Pixelkoordinaten des Punktes, in dem sich Bildebene und optische Achse schneiden
- f ist die Brennweite

- σ ist der Scherungsfaktor, der angibt, inwieweit das Sensorgitter der Kamera von einem rechteckigen Gitter abweicht
- ρ bezeichnet die Seitenverhältnisse des Sensorgitters, gibt also an, in welchem Abstandsverhältnis die Sensoren in horizontaler und vertikaler Richtung angeordnet sind

Idealerweise sind $\sigma = 0$, $\rho = 1$ und (C_x, C_y) liegt in der Bildmitte. Die Kamera wird jedoch in der Praxis immer leicht nach vorne oder hinten gekippt sein, ebenso wie sie im oder gegen den Uhrzeigersinn um die z-Achse verdreht sein wird. Diese Abweichungen werden in den Rotationsmatrizen $\boldsymbol{\Omega}_x$ und $\boldsymbol{\Omega}_z$ beschrieben:

$$\boldsymbol{\Omega}_x = \begin{bmatrix} 1 & 0 & 0 \\ 0 & cos\omega_x & -sin\omega_x \\ 0 & sin\omega_x & cos\omega_x \end{bmatrix} \qquad \boldsymbol{\Omega}_z = \begin{bmatrix} cos\omega_z & -sin\omega_z & 0 \\ sin\omega_z & cos\omega_z & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{10.16}$$

Der Parameter f wurde bereits berechnet. Also bleiben noch sechs unbekannte Parameter $(C_x, C_y, \sigma, \rho, \omega_x, \omega_z)$ zu bestimmen. Für die Berechnung dieser Parameter gehen wir von der idealen Kamera aus. Somit haben die übrigen Kameraparameter folgende Werte:

$$C_x = \frac{Bildbreite}{2} \qquad C_y = \frac{Bildhoehe}{2} \qquad \omega_x = 0 \qquad \omega_z = 0 \qquad \sigma = 0 \qquad \rho = 1 \tag{10.17}$$

Wenn $\boldsymbol{H}_{i+1}$ exakt bekannt ist, so gilt Gleichung (10.5). Mit den Näherungen in (10.17) erhalten wir nur eine Approximation der Abbildung:

$$\boldsymbol{I}_i = \boldsymbol{H}_{i+1}^{ex} \boldsymbol{I}_{i+1} \approx \boldsymbol{H}_{i+1} \boldsymbol{I}_{i+1} = \boldsymbol{S}^{-1} \boldsymbol{R}_y(\theta_{i+1}) \boldsymbol{S} \boldsymbol{I}_{i+1} \tag{10.18}$$

Als numerisch berechenbares Maß für die Güte der Approximation $\boldsymbol{H}_{i+1}$ wird die Kreuzkorrelation verwendet. Damit gilt: Je besser $\boldsymbol{H}_{i+1} = \boldsymbol{S}^{-1} \boldsymbol{R}_y(\theta_{i+1}) \boldsymbol{S}$ die gesuchte exakte Transformation $\boldsymbol{H}_{i+1}^{ex}$ approximiert, umso näher liegt die *Correlation*$(\boldsymbol{I}_i, \boldsymbol{S}^{-1} \boldsymbol{R}_y(\theta_{i+1}) \boldsymbol{S} \boldsymbol{I}_{i+1})$ bei 1. Damit lässt sich die Bestimmung der Strukturmatrix $\boldsymbol{S}$ als Minimierungsproblem stellen: $\boldsymbol{S}$ ist optimal, wenn die Fehlerfunktion minimal wird.

$$error(C_x, C_y, \sigma, \rho, \omega_x, \omega_z) = \sum_{i=1}^{n} 1 - Correlation(\boldsymbol{I}_i, \boldsymbol{S}^{-1} \boldsymbol{R}_y(\theta_{i+1}) \boldsymbol{S} \boldsymbol{I}_{i+1}) \tag{10.19}$$

Zur Minimierung kann jedes dazu geeignete Verfahren eingesetzt werden, z.B. die *Powells Methode*, ein einfacher Algorithmus zur Lösung mehrdimensionaler Minimierungsprobleme.

Sind die acht Parameter $C_x, C_y, \sigma, \rho, \omega_x, \omega_z, \theta_i$ und f bestimmt, ist die homogene Transformation $\boldsymbol{H}_{i+1}$ eindeutig definiert. Der Modellierungsprozess resultiert somit in einem Kameramodell $\boldsymbol{S}(C_x, C_y, \sigma, \rho, \omega_x, \omega_z, f)$ und einer Reihe von relativen Rotationen θ_i zwischen jedem der Bilder. Anhand dieser Parameter können nun *mapping*-Funktionen von jedem Bild in der Sequenz zu jedem anderen Bild erstellt werden.

$$\boldsymbol{I}'_i = \boldsymbol{S}^{-1} \boldsymbol{R}_{y_{i+1}} \boldsymbol{R}_{y_{i+2}} \boldsymbol{R}_{y_{i+3}} ... \boldsymbol{R}_{y_j} \boldsymbol{S} \boldsymbol{I}_j \tag{10.20}$$

Mit den Matrizen $\boldsymbol{R}$ und $\boldsymbol{S}$ sind sämtliche internen und externen Kameraparameter für jede Aufnahme bekannt. Also legt jeder Bildpunkt zusammen mit dem Projektionszentrum einen Sehstrahl im dreidimensionalen Raum fest. Die Projektion der Einzelbilder auf die Zylinderhülle ist möglich. Hierfür müssen die Schnitte des Zylinders mit den gegebenen Halbstrahlen für alle relevanten Szenepunkte berechnet werden.
Um das Panorama wieder auf die Bildebene zu projizieren, können ebenso für alle relevanten Punkte auf dem Zylinder Projektionenstrahlen durch das Projektionszentrum definiert werden, die dann mit der Bildebene geschnitten werden.

10.3.2 Multinode Panoramen

Die bisherigen Betrachtungen erlauben dem Benutzer lediglich eine Veränderung der Blickrichtung. Multinode Panoramen ermöglichen es, dass sowohl Blickrichtung als auch der Standpunkt verändert werden können. Zur Vereinfachung werden Einschränkungen dahingehend vorgenommen, dass man die möglichen Kamerapositionen auf die diskreten Orte beschränkt, an denen auch tatsächlich *environment maps* vorliegen. Die einzelnen Panoramen (oder *nodes*) werden dafür zu einem Komplex verknüpft. Die Navigation zwischen den Panoramen funktioniert mittels ausgewiesener, interaktiver Stellen, so genannten *hot spots*. Sie können interaktiv reagieren (z.B. auf einen Mouseklick) und sowohl Panoramen untereinander verbinden als auch Panoramen mit Objekt-Movies (Abb.10.5) oder anderen multimedialen Inhalten.

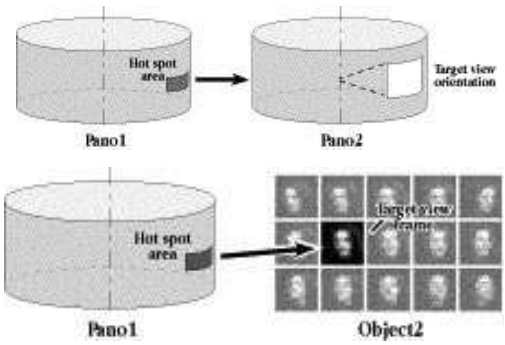


Abbildung 10.5: Hot spots zur Navigation

10.3.3 Panorama Movie Datei

Die so entstandenen Panoramen werden zur Speicherung in kleinere Teilbilder (sog. *tiles*) aufgeteilt. Diese werden unabhängig voneinander komprimiert und in einer normalen QT Videospur gespeichert. Da der Zugriff auf diese Spur jedoch notwendigerweise nicht linear ist, benötigt man bei einem Panorama Movie eine zusätzliche Spur, die Knoten (*nodes*) mit Informationen zu den einzelnen Panoramen und deren Verknüpfungen untereinander enthält. Diese wird als Panoramaspur bezeichnet. In einer optionalen, weiteren QT Videospur werden die zugehörigen *hot spot* Bilder gespeichert. Eine Panorama Movie Datei (Abb.10.6) besteht somit aus einer Panorama Spur und zwei Video Spuren, die alle dieselbe Länge haben und auf einer einheitlichen Zeitskala basieren.

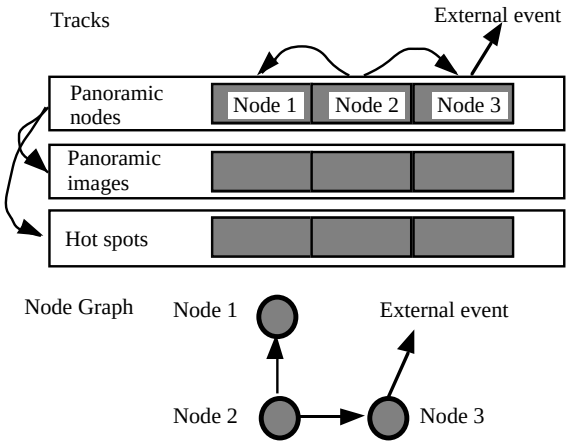


Abbildung 10.6: Aufbau einer Panorama Movie Datei

10.3.4 Panorama Movie Player

Der Zugriff auf die Panorama Movie Datei erfolgt mit dem Panorama Player, der kontinuierliche Bewegung und Zooming unterstützen soll. Dabei werden die am aktuellen Bildausschnitt beteiligten *tiles* in einen schnellen Zwischenspeicher (*cache*) kopiert und im *offscreen buffer* dekomprimiert. Dieser Puffer ist normalerweise kleiner als das vollständige Panorama, da jeweils auch nur ein Bruchteil des Panoramas am Bildschirm sichtbar ist. Solange der momentane Bildausschnitt in dieser Region bleibt, ist also auch keine weitere Dekompression nötig. Aus den dekomprimierten *tiles* werden die sichtbaren Teile extrahiert und mittles Image Warping in Echtzeit auf die Bildebene reprojiziert (Abb.10.7). Dabei wird das Bild, während sich der Benutzer in der Szene bewegt, in einer niedrigeren Auflösung und ohne Glättung angezeigt. Erst wenn keine Bewegung mehr stattfindet wird das Bild in höherer Qualität angezeigt und Aliasing-Effekte werden geglättet.

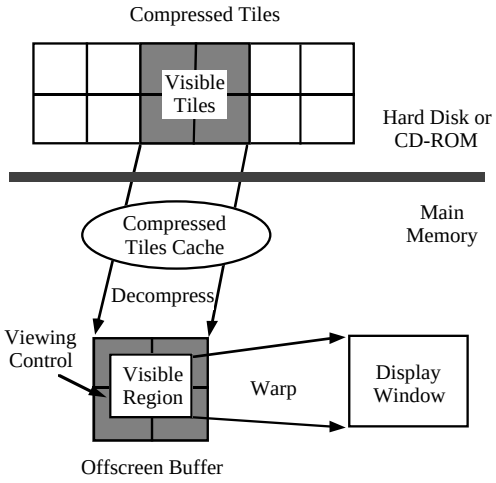


Abbildung 10.7: Zugriff auf Panorama Movies

10.3.5 Zusammenfassung: Panorama Movie Erstellungsprozess

Node Selection:
Um ein Panorama Movie zu erstellen, müssen als erstes die Kamerastandpunkte im Raum festgelegt werden. Der Abstand zwischen zwei angrenzenden Knoten ist dabei von der Größe der virtuellen Umgebung und dem Abstand zu Objekten in der Nähe abhängig. Bei Außenszenen ist der Abstand erfahrungsgemäß größer als bei der Darstellung von Innenräumen.
Panoramas können mittels Computer-Rendering, Panoramafotos, aufgenommen mit einer speziellen Panoramakamera oder durch das Zusammenfügen von Einzelaufnahmen erstellt werden.
Stitch:
Um die einzelnen Fotos zu einem Panorama verbinden zu können, müssen sich diese überlappen. In der Praxis scheint eine 50%ige Überschneidung am besten, da benachbarte Bilder sehr unterschiedliche Helligkeitsstufen haben können und diese Intensitätsunterschiede dann einfacher geglättet werden können. Der *Stitcher* fügt die Einzelbilder automatisch zusammen. Bei 8 von 10 Bildern geschieht das erfolgreich, so, dass keine manuelle Nachbearbeitung stattfinden muss.
Mark Hot Spots:
Hot spots identifizieren bestimmte Regionen eines Panoramabildes für Interaktionen, wie Navigation oder Aktivierung von Aktionen. Die Anzahl der *hot spots* pro Bild ist auf 256 begrenzt. Eine Art *hot spot* Bilder zu erzeugen ist, Pseudofarben über die Panoramabilder zu legen. Dabei muss das *hot spot* Bild nicht dieselbe Auflösung wie das Panorama Bild haben. Die Wahl der Auflösung ist abhängig von der Genauigkeit des darzustellenden Bereichs.
Link:
Der *Linking*-Prozess verknüpft und registiert die unterschiedlichen Blickrichtungen in benachbarten Panorama-Knoten.
Dice & Compress:
Schließlich werden die Panoramabilder in *tiles* aufgeteilt und komprimiert. Die Größe der *tiles* sollte in Bezug auf das Laden der Daten und die *offscreen buffer*-Größe optimiert sein. Eine Unterteilung in 24 vertikale Streifen hat sich dafür als am geeignetsten erwiesen.

10.4 QuickTime VR Objekt Movies

Invertiert man das Prinzip der Panorama Movies, bei dem der Betrachter im Mittelpunkt steht und sich in alle Richtungen drehen und so das Panorama betrachten kann, erhält man das Prinzip der Objektrotation. Dabei wird es dem Benutzer ermöglicht ein Objekt in alle Richtungen zu drehen.

10.4.1 Erstellung eines Objekt Movies

Um ein solches Objekt Movie zu erstellen wird jedoch nicht das Objekt vor einer festen Kamera rotiert, sondern die Kamera wird auf einer virtuellen Hüllkugel um das Objekt bewegt. Dabei wird, mit gleich bleibend ausgerichteter Kamera, das Objekt von allen Seiten aufgenommen. Der Bewegungsspielraum der Kamera wird dabei derart gerastert, dass die

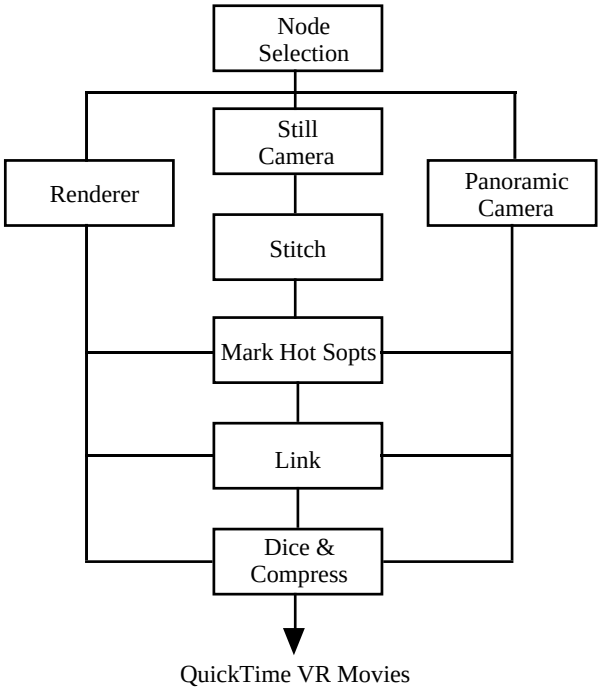


Abbildung 10.8: Panorama Movie Erstellungsprozess

möglichen Kamerapositionen den Schnittpunkten von virtuellen Längen- und Breitengraden entsprechen (Abb. 10.9).

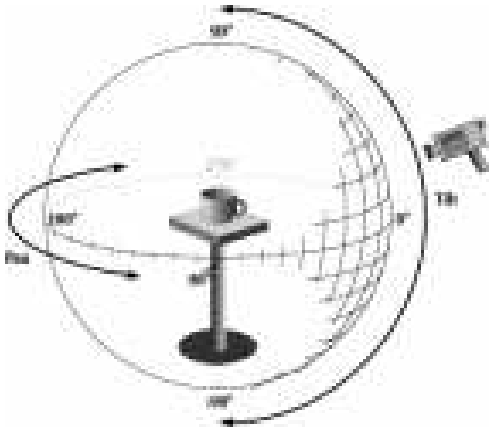


Abbildung 10.9: Virtuelle Hüllkugel

Eine Möglichkeit der Ablegung ist es, alle möglichen Orientierungen eines Objektes zu speichern (*navigable movie* Ansatz). Da dies jedoch sehr speicheraufwendig ist eignet sich der *view interpolation* Ansatz hier besser. Hierbei werden nur einige Schlüsselansichten gespeichert und die neuen Ansichten werden *on-the-fly* aus diesen interpoliert.
Die aufgenommenen Bilder werden in einem 2D-Array gabgelegt. Benachbarte Bilder sind sich dabei ähnlich. Dazu ordnet man alle Aufnahmen von einem virtuellen Breitengrad aus horizontal in einem Feld an, die von Längengraden vertikal. Werden für eine Kameraposition mehrere Bilder aufgenommen, kann durch hintereinander Abspielen ein zeitliches Verhalten dargestellt werden, d.h. Animation von Objekten ist möglich (z.B. Augenzwinkern).

10.4.2 Objekt Movie Datei und Objekt Player

Objekt Movies werden auf einer normale QT-Videospur gespeichert. Die Daten werden zeilenweise abgelegt. Zusätzliche Informationen über die Anzahl der Bilder pro Zeile und Spalte befinden sich im Movie-*header*. Wenn der Betrachter

das Objekt horizontal dreht, dann wird auf die nebeneinander angeordneten Bilder zugegriffen und die Zugriffszeit ist minimal. Die vertikale Bewegung erfordert mehr Suchaufwand beim Zugriff und dauert daher länger. Die Objekt Mo-

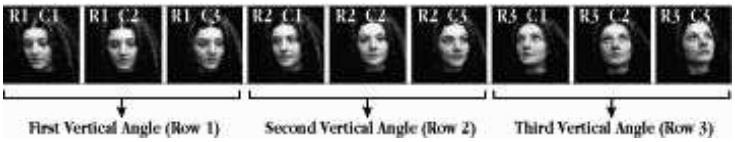


Abbildung 10.10: Lineare Speicherung eines Objekt Movies

vies können mit dem Objekt Player betrachtet werden. Dabei hat der Benutzer die Möglichkeit das Objekt mit Hilfe der Mouse zu rotieren.

10.5 Zooming

Das Vergrößern bzw. Verkleinern des aktuellen Bildausschnitts innerhalb eines Panoramas oder Objekt Movies wird von QTVR unterstützt. Für das Zoomen wird von QTVR das Prinzip der Bildskalierung verwendet, d.h. der entsprechende Ausschnitt wird z.B. vergrößert, indem den jeweiligen Bildpunkten jedoch lediglich eine größere Fläche zugewiesen wird. Feinere Details werden somit nicht sichtbar, man erhält nur eine gröbere Teilansicht mit effektiv niedrigerer Auflösung. Durch Einführung eines Parameters, der das Heranholen eines Bildausschnittes nur in bestimmten Schranken erlaubt, wird dieses Problem reduziert.
Bessere Ergebnisse ließen sich durch die Verwendung von *environment maps* unterschiedlicher Auflösung erzielen. In eine Baumstruktur (z.B. Quad Trees) integriert, kann dann, je nach Auswahl der Ansicht, zwischen den einzelnen Ansichten interpoliert werden.

10.6 Ausblick

Unter Verwendung von QTVR kann man Panorama Movies erstellen, bei denen die Möglichkeit besteht einzelne Panoramen über *hot spots* miteinander zu verbinden. Dadurch wird ein gewisser Bewegungsspielraum geschaffen, der jedoch den Betrachter beim Umherschauen auf die Betrachterstandpunkte der einzelnen Panoramen einschränkt. Das Ziel der freien Wahl von *viewpoint* und *view direction* kann also nur näherungsweise erreicht werden.
In anderen Ansätzen wird ebenfalls versucht dem Ziel der völligen Bewegungsfreiheit so nahe wie möglich zu kommen: Beim „Light Field Rendering"-Verfahren von *Marc Levoy und Pat Hanrahan* [6] handelt es sich ebenfalls um einen *Image-Based Rendering* Ansatz, der es jedoch erlaubt aus vorhandenen Aufnahmen neue Ansichten von beliebiger Betrachterposition zu erzeugen.
In „The Lumigraph"von *Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski und Michael F. Cohen* [5] wird die Idee von QTVR, den kompletten Lichtfluss in einer Region der Umgebung einzufangen, weiterentwickelt.

Literaturverzeichnis

[1] E.H. Adelson and J.R. Bergen. The plenoptic function and the elements of early vision. *Computational Models of Visual Processing, Chapter 1*, 1991. Edited by Michael Landy and J. Antony Movshon, The MIT Press, Cambridge, Massechusetts.

[2] Shenchang Eric Chen. QuickTimeVR—an image-based approach to virtual environment navigation. *Computer Graphics*, 29(Annual Conferende Series):29–38, 1995.

[3] K. Kanatani. Transformation of optical flow by camera transformation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, March 1988. Vol.10, No.2.

[4] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics*, 29(Annual Conferende Series):39–46, 1995.

[5] Richard Szeliski und Michael F. Cohen Steven J. Gortler, Radek Grzeszczuk. The lumigraph. *Computer Graphics*, 30(Annual Conferende Series):43–54, 1996.

[6] Marc Levoy und Pat Hanrahan. Light field rendering. *Computer Graphics*, 30(Annual Conferende Series):31–42, 1996.

Vortrag 11

Graphik/Texture Mapping/Rendering

Hendrik Wiebel

12. Februar 2004

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
avatarx@uni-koblenz.de
http://www.uni-koblenz.de/∼avatarx

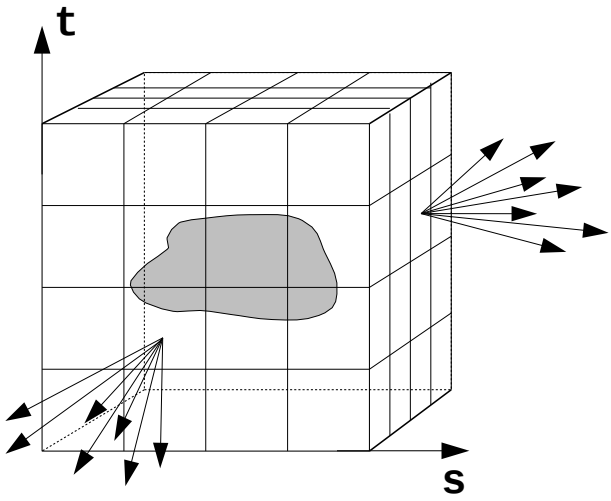


Abbildung 11.1: Die Würfeloberfläche repräsentiert die spezifische Lichtausstrahlung der eingeschlossenen Szene

11.1 Einleitung

Dieses Kapitel beschäftigt sich mit der Darstellung von Objekten und Szenen. In der Computergraphik werden dazu meist polygonale Modelle mit entsprechenden Texturen erstellt, deren Beleuchtung mit mehr oder weniger aufwendigen Verfahren simuliert wird. Um die Exploration des Dargestellten in Echtzeit zu garantieren, müssen bei der Komplexität der Objekte und der Qualität der Simulation Abstriche gemacht werden, wobei der Grad an Realismus sinkt.

Die hier vorgestellten Verfahren verfolgen dabei einen völlig anderen Ansatz. Um die realen Beleuchtungsverhältnisse zu simulieren, wird die Szene aus einer Vielzahl von Blickwinkeln heraus gefilmt und das resultierende Material als Grundlage zur Rekonstruktion der plenoptischen Funktion verwendet.

Die plenoptische Funktion gibt für jede Position (x, y, z) den Lichtstrom in alle Richtungen (Θ, Φ) an, d.h. es handelt sich um eine 5D Repräsentation. Im Folgenden wird schwerpunktmässig auf den Lumigraph von Gortler, Grzeszczuk, Szeliski und Cohen [1] eingegangen, welcher eine Untermenge der vollständigen plenoptischen Funktion darstellt, wobei die 5 Dimensionen auf 4 reduziert werden.

11.2 Repräsentation des Lumigraph

11.2.1 von 5D zu 4D

Man geht zunächst von folgenden Annahmen aus:

1. Die spezifische Lichtausstrahlung entlang eines Strahls bleibt konstant, d.h. atmosphärische Effekte usw. bleiben unberücksichtigt

2. Nur das Licht, welches die konvexe Hülle der Szene verlässt, ist von Interesse

Als Konsequenz daraus wird die Szene in einen gedachten Würfel (oder Boundingbox) eingeschlossen und von aussen das Licht betrachtet, welches strahlenförmig die 6 Würfelflächen verlässt (Abbildung 11.1).

Von jeder Position in der Szene kann man die Strahlen in alle Richtungen bis zu ihrem Schnittpunkt mit der jeweiligen Würfelfläche verfolgen und die lichttechnischen Grössen bestimmen. Die Strahlen sind pro Seitenfläche durch 4 Parameter eindeutig bestimmt, so dass der Lumigraph als vierdimensionale Funktion betrachtet wird.

Das Prinzip des umgebenden Würfels wird in der Computergraphik oft bei indirekten Beleuchtungsmodellen wie dem Radiosityverfahren verwendet, um den Lichtaustausch von komplizierten Strukturen auf wenige Flächen zu reduzieren.

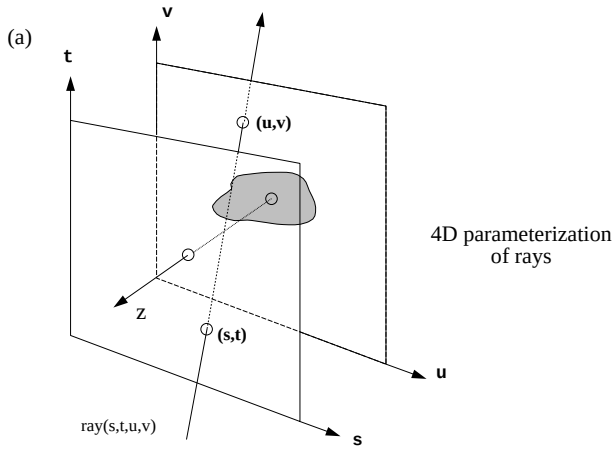


Abbildung 11.2: 4D Parametrisierung eines Strahls

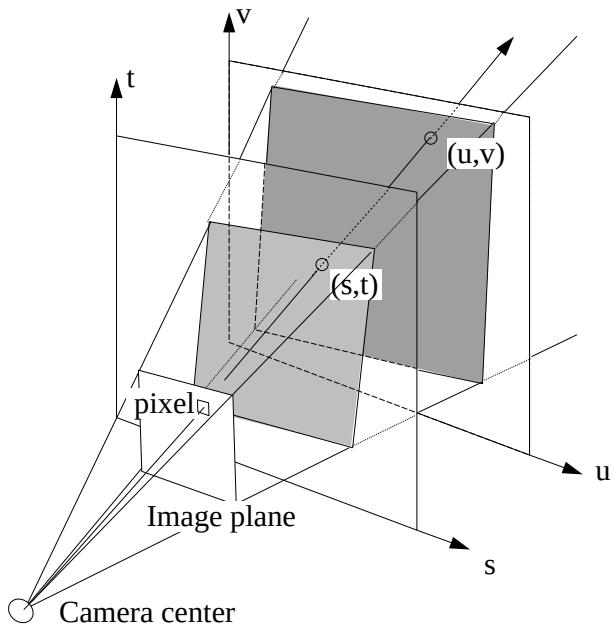


Abbildung 11.3: Füllen der Pixel durch Raytracing

11.2.2 Parametrisierung

Jede Seitenfläche des Würfels liegt in einer Ebene, die durch 2 senkrecht aufeinanderstehende Kantenvektoren aufgespannt wird. Diese Achsen erhalten die Bezeichnung s und t. Im Würfelinneren befindet sich direkt hinter jeder (s,t) $Ebene$ eine (u,v) $Ebene$, welche parallel zu ihr steht. Die Richtung eines austretenden Strahls ist durch den Schnittpunkt mit beiden Ebenen eindeutig festgelegt. Es muss natürlich vorher klassifiziert werden, durch welche Würfelseite der Strahl nach aussen dringt und welches der 6 Ebenenpaare in Frage kommt. Ein Punkt im 4D Lumigraph besitzt also die Koordinaten (s,t,u,v) (Abbildung 11.2).

Die Bildebene befindet sich ausserhalb des Würfels und ihre Pixel werden eingefärbt, indem durch sie von der virtuellen Kamera ein Strahl geschossen wird. Dieser durchdringt danach eine der Würfelseiten und damit ihre beiden Ebenen. Anschliessend verlässt er entweder den Würfel auf der anderen Seite oder trifft ein Szenenobjekt. Der gesuchte Wert ist die spezifische Lichtausstrahlung des Objektpunktes in die Schussrichtung oder aber die Hintergrundfarbe der Szene, falls nichts getroffen wurde. Dieser Wert existiert abhängig von der Würfelseite als Eintrag an der Position (s,t,u,v) im Lumigraph und wird als Pixelfarbe gesetzt (Abbildung 11.3).

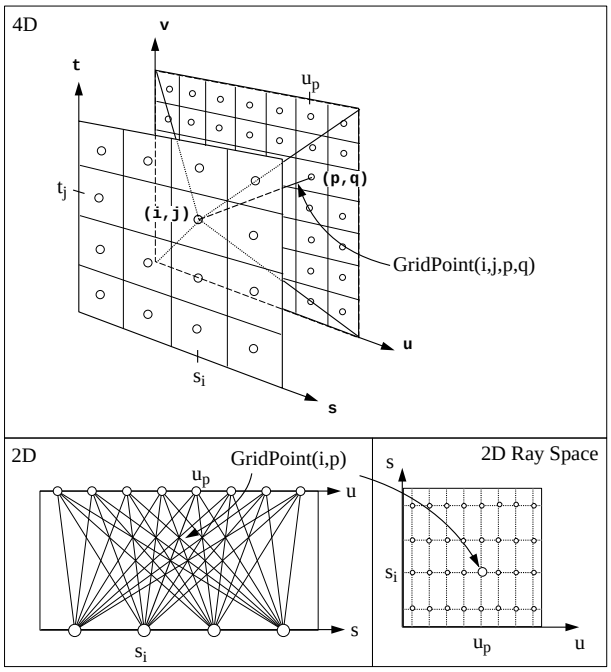


Abbildung 11.4: Diskretisierung des Lumigraph

11.2.3 Diskretisierung des Lumigraph

Für die Implementierung des Lumigraph ist es notwendig eine geeignete Diskretisierung für die kontinuierlichen Werte (s,t,u,v) zu finden. Hierzu wird für die (s,t) $Ebene$ ein Raster mit M Feldern angelegt und analog dazu eines für die (u,v) $Ebene$ mit N Unterteilungen. Jeder Punkt im (4D) Raster wird mit einem Koeffizienten und einer Basisfunktion verknüpft. Der Koeffizient stellt dabei die lichttechnische Grösse dar, die durch den Rekonstruktionskern (d.h. die Basisfunktion) modifiziert wird. Demnach besteht der diskretisierte Lumigraph nicht mehr aus unendlich vielen (s,t,u,v) - Werten, sondern aus $M \times N$ Einträgen der Form (i,j,p,q).

$$\tilde{L}(s,t,u,v) = \sum_{i=0}^{M}\sum_{j=0}^{M}\sum_{p=0}^{N}\sum_{q=0}^{N} x_{i,j,p,q} B_{i,j,p,q}(s,t,u,v) \tag{11.1}$$

Wie man in 11.1 sehen kann, errechnet sich der Wert an einer beliebigen Position (s,t,u,v) aus der Summe aller Koeffizienten an den Rasterpunkten gewichtet mit einer (für den Rasterpunkt) individuellen Basisfunktion B. Im einfachsten Fall wird für B eine konstante Funktion gewählt, die an der Stelle (i,j,p,q) gleich 1 ist und ansonsten 0. Für einen beliebigen Schnittpunkt sucht man also den nächsten Nachbarn im Raster. Allerdings wirken die Ergebnisbilder auf diese Art und Weise ziemlich pixelig.

Bessere Resultate mit weniger Aliasing erreicht man mit einer quadralinearen Basisfunktion, bei der ein zugehöriger Rasterpunkt mit 1 gewichtet wird und die Funktion zu den Zentren der Nachbarn linear auf 0 abfällt. Es handelt sich dabei um eine Interpolation zwischen 16 Punkten, d.h. jeweils 8 für (s,t) und (u,v) $Ebene$.

Um die Werte $x_{i,j,p,q}$ überhaupt erst für den diskretisierten Lumigraph $\tilde{L}$ zu generieren, muss der kontinuirliche Lumigraph L auf $\tilde{L}$ projiziert werden. Der Wert eines Rasterpunktes repräsentiert dann eine Approximation des Lichtes der Region, welche die Fläche dieses Feldes einnimmt. Dabei wird über unendlich viele, eng beieinanderstehende Lichtpunkte integriert. Man kann diese Projektion als Tiefpassfilterung von L interpretieren, an die sich ein Punktesampling anschliesst.

Die Wahl der Auflösung, d.h. die Anzahl der Unterteilungen $M \times M$ und $N \times N$ der Ebenen wird durch die Annahme beeinflusst, dass sichtbare Objekte näher an der (u,v) $Ebene$ liegen, weil sie sich im Zentrum des Würfels befinden. Deshalb sollte $N \times N$ in etwa der Auflösung des Ausgabebildes entsprechen. In Abbildung 11.5 ist zu erkennen, dass die Strahlen teilweise stark streuen und grosse Abstände zwischen den Schnittpunkten in der (s,t) $Ebene$ existieren. Somit kann für sie eine erheblich kleinere Unterteilung gewählt werden. Oft ist $M = \frac{1}{8}N$ völlig ausreichend.

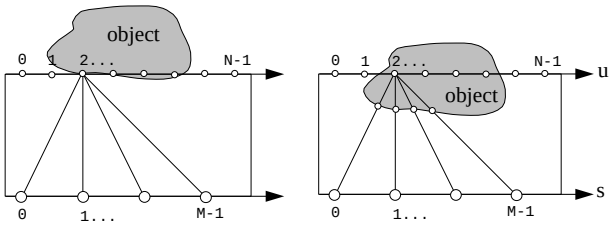


Abbildung 11.5: Diskretisierung des Lumigraph

QuadralinearDepthCorrect(s,t,u,v)

Result = 0
$h_{st} = s_1 - s_0$ / $*$ $gridspacing$ $*$ /
$h_{uv} = u_1 - u_0$
for each of the four (s_i, t_j) surrounding (s, t)

$u' = u + (s - s_i) * z/(1 - z)$
$v' = v + (t - t_j) * z/(1 - z)$
temp = 0
for each of the four (u_p, v_q) surrounding (u', v')

interpWeight$_{uv} = (h_{uv} - |u_p - u'|) * (h_{uv} - |v_q - v'|)/h_{uv}^2$
temp += interpWeight$_{uv}$ $* L(s_i, t_j, u_p, v_q)$

interpWeight$_{st} = (h_{st} - |s_i - s|) * (h_{st} - |t_j - t|)/h_{st}^2$
Result += interpWeight$_{st}$ $*$ temp

return Result

Abbildung 11.6: Algorithmus zur Tiefenkorrektur

11.2.4 Verwendung geometrischer Information

Ein unschöner Effekt der Diskretisierung liegt darin, dass Strahlen zwar ein Objekt schneiden, jedoch trotzdem einem Feld zugewiesen werden können, welches eher die spezifische Lichtausstrahlung des Nachbarobjektes oder die Hintergrundfarbe enthält. Dabei existiert durchaus ein besserer Wert in einem der Nachbarfelder. Je niedriger die Auflösung, um so häufiger ist dies der Fall. Ein Strahl (s, t, u, v), dem ein Rasterelement (i, j, p, q) zugeordnet wird, schneidet ein Szenenobjekt in einer Tiefe z. Ist z bekannt, so können weitere Strahlen ermittelt werden, die das Objekt ebenfalls im selben Punkt schneiden. Werden Strahlen von der virtuellen Kamera aus verschossen, so unterzieht man sie einer Korrektur, damit sie sich nach der Diskretisierung möglichst im selben Punkt schneiden wie vorher. Somit bleibt der optische Fluss gleichmässig, wenn die Position verändert wird und Sprünge wie auch Ungleichmässigkeiten werden reduziert. Der Korrekturprozess wird in die Basisfunktion integriert, wie man in Gleichung 11.2 sieht.

$$B'_{i,j,p,q}(s, t, u, v) = B_{i,j,p,q}(s, t, u', v') \tag{11.2}$$

u' und v' werden im Zuge des Korrekturalgorithmus 11.6 berechnet, dessen Grundlage die Verhältnisgleichung 11.3 darstellt, welche sich an Abbildung 11.7 nachvollziehen lässt. Abbildung 11.7 zeigt das Ergebnis der Korrektur. Auf die Gewinnung der geometrischen Information aus der realen Szene wird später eingegangen.

$$\frac{u' - u}{z} = \frac{s - s_i}{1 - z} \tag{11.3}$$

$$u' = u + (s - s_i)\frac{z}{1 - z} \tag{11.4}$$

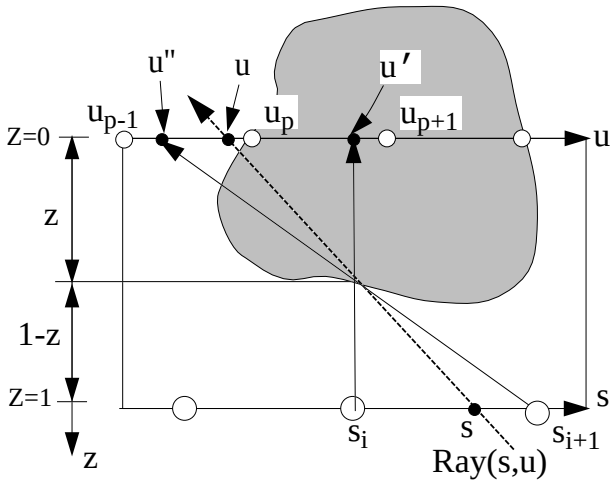


Abbildung 11.7: Tiefenkorrektur der Strahlen

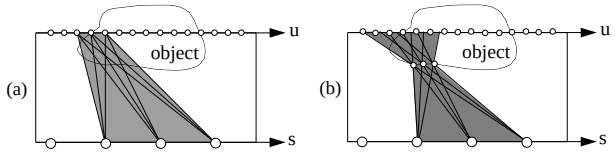


Abbildung 11.8: Tiefenkorrektur der Strahlen

11.2.5 Das Lumigraph System

Bei der Erstellung des Ausgangsmaterials für den Lumigraph unterscheidet man zwischen **computergenerierten** und **realen** Szenen. Erstere stellen dabei den einfacheren Fall dar, weil problemlos beliebig viele Bilder von exakt berechneten Positionen aus berechnet werden können. Dazu wird die virtuelle Kamera in der jeweiligen Graphiksoftware an die Stelle (i, j) im gedachten Raster platziert. Die Blickrichtung ist entlang der z-Achse und für jedes imaginäre Feld (p, q) wird das Frustum entsprechend angepasst. Um die beim Thema Diskretisierung angesprochene Tiefpassfilterung zu berücksichtigen, werden pro Pixel (oder pro Feld) mehrere Strahlen verschossen und der mit B gewichtete Mittelwert gebildet. Auf diese Art erhält man für jede Auflösung die gewünschten Koeffizienten für den diskretisierten Lumigraph.

Während bei synthetischen Szenen Skripte die korrekte Positionierung und Ausrichtung der Kamera übernehmen, um geeignete Bilder zu generieren, sieht das in der Realität anders aus. Für gleichmässige Abstände und Winkel sind Motion Control Plattformen nötig, obwohl oft die Benutzung einer einfachen Videokamera angestrebt wird. In den weiteren Ausführungen wird die Vorgehensweise bei der Verwendung eines solch “handelsüblichen“ Modells erläutert. Die Bestimmung der intrinsischen Parameter wie Linsenverzerrung, Pixelgrösse usw. erfolgt durch eine einmalige Kalibrierung mit Hilfe eines Testbildes. Die extrinsischen Parameter Rotation und Position ändern sich ständig und werden durch Marker berechnet, die vorher in der Szene verteilt werden.

Das Objekt wird auf eine Bühne gestellt, die aus einem offenen Würfel besteht, bei dem die Seitenflächen beliebig montierbar sind. Auf ihnen sind die Kalibrierungsmarker verteilt (Abbildung 11.10). Die Farbe der Wände und Symbole sind so gewählt, dass sie später gut vom Objekt separiert werden können. Mit Hilfe dieser BlueBox - Technik gewinnt man auch die geometrischen Informationen für die Tiefenkorrektur der Strahlen. Dazu wird ein Bild binarisiert, wobei alle Pixel des Objektes auf weiss gesetzt werden. Ein Octree wird angelegt und Strahlentests gegen die Silhouette des Objektes durchgeführt. Bei Erfolg wird das jeweilige Octree - Element unterteilt und in dieser Region weiter getestet, ansonsten

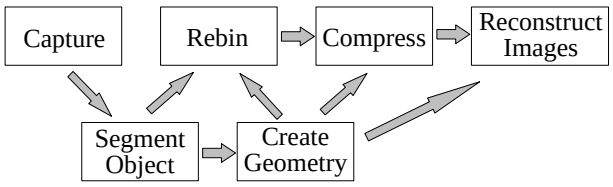


Abbildung 11.9: Das Lumigraph System

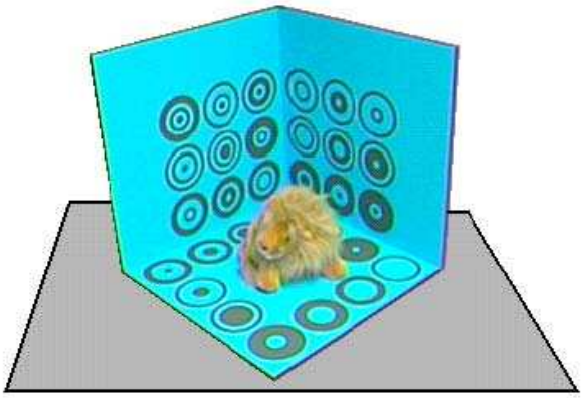


Abbildung 11.10: Die capture stage

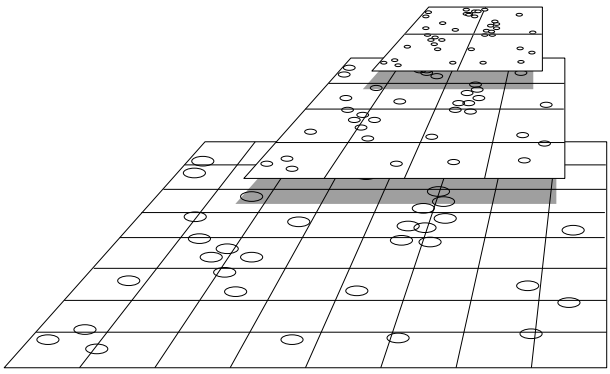


Abbildung 11.11: Reduzierung der Auflösung

erfolgt seine Löschung. Durch die Abarbeitung einer Bildsequenz entsteht so ein Volumenmodell aus Voxeln. Die Position des Voxels wird als RGB Wert abgespeichert. Je nach betrachteter Würfelseite wird eine andere Farbkomponente als Tiefenwert z interpretiert.

Damit wären die Schritte “Capture“, “Segment Object“ und “Create Geometry“ aus Abbildung 11.9 abgearbeitet. Nun erfolgt die Verwendung der gewonnenen Daten beim Aufbau des Lumigraph.

Rebinning - Aufbau des Lumigraph

$$x_{i,j,p,q} = \int L(s,t,u,v)\tilde{B}_{i,j,p,q}(s,t,u,v)dsdtdudv \tag{11.5}$$

11.5 sagt aus, dass die Werte an allen Positionen (s,t,u,v) mit $\tilde{B}$ gewichtet und dann aufaddiert werden, um den Eintrag an der Rasterposition (i,j,p,q) zu bestimmen. In der Praxis wird das Integral durch eine endliche Menge von Samples der Funktion L ausgewertet. Dabei repräsentiert jedes Pixel der Videobilder eines der Samples. Bei dem Datensatz ergeben sich jedoch einige Probleme. Die Samples sind ungleichmässig gestreut und auch innerhalb einer Sampleregion variiert ihr Abstand. Dadurch entstehen grosse Löcher zwischen den gesampelten Bereichen. Ausserdem ist ihre Anzahl relativ hoch, so dass herkömmliche Methoden zum Lösen von Gleichungssystemen nicht in Frage kommen.

Ein Lösungsansatz liegt darin, dass die Auflösung der Daten in Verbindung mit einem Filter reduziert wird, so dass die Löcher kleiner werden. Die einzelnen Samples jedoch sind bereits die minimale Bildeinheit und können nicht kleiner werden. Ihre Intensität nimmt dafür ab und um sie herum steigt sie etwas an - ein typischer Weichzeichnungseffekt. Somit werden die leeren Flächen teilweise mit interpolierten Daten gefüllt (Abbildung 11.11). Damit übermässig stark abgetastete, aber kleine Regionen nicht zu grossen Einfluss auf das Gesamtergebnis haben, werden sie teilweise zu einem einzigen Sample zusammengefasst und ihr Mittelwert verwendet.

Diese Ideen werden kombiniert in einem hierarchischen Algorithmus, der aus 3 Phasen besteht:

1. Splat
2. Pull
3. Push

Beim **Splatting** werden die Sampledaten als Annährung des Integrals verwendet und die Koeffizienten x (die Werte an den Rasterpunkten) sowie die Gewichte w (der Basisfunktion B) ermittelt.

$$w_i^0 = \sum_k \tilde{B}_i(s_k) \tag{11.6}$$

$$x_i^0 = \frac{1}{w_i^0} \sum_k \tilde{B}_i(s_k) L(s_k) \tag{11.7}$$

Bezogen auf 11.6 und 11.7 stellt s_k die räumliche Position eines Samples dar. Index i gibt die aktuelle Iteration an und der Exponent die jeweilige Auflösungsstufe. Zu Anfang sind die Gewichte an den Stellen, wo Samples existieren gleich 1 und leere Bereiche 0. In späteren Iterationen wird sich das ändern.

Die **Pullphase** dient zum Anlegen einer niedriger aufgelösten Kopie der Daten. Wieder werden x und w berechnet, wobei der Weichzeichnungseffekt dazu führt, dass einige leere stellen aus der vorherigen Phase mit neuen Werten belegt werden.

$$w_i^{r+1} = \sum_k \tilde{h}_{k-2i} min(w_k^r, 1) \tag{11.8}$$

$$x_i^{r+1} = \frac{1}{w_i^{r+1}} \sum_k \tilde{h}_{k-2i} w_k^r x_k^r \tag{11.9}$$

Der min - operator in 11.8 setzt dabei eine obere Grenze, die festlegt, inwieweit ein Koeffizient aus einer stark gesampelten Region Einfluss auf die Gesamtsumme erhält. Der Kern $\tilde{h}_{k-2i}$ dient dazu, um die Koeffizienten der höheren Auflösung in der nächstniedrigeren zu kombinieren, d.h. mit ihm wird die eigentliche Reduzierung durchgeführt.

Daran schliesst sich die **Pushphase** an. Hier werden die Ergebnisse aus der Pullphase mit denen des Splattings zusammengefügt. Allerdings wird darauf geachtet, dass "gute" Werte nicht einfach überschrieben werden. Das wird durch eine Überblendung gewährleistet, die sich auf die Gewichte w stützt.

$$tw_i^r = \sum_k h_{i-2k} min(w_k^{r+1}, 1) \tag{11.10}$$

$$tx_i^r = \frac{1}{tw_i^r} \sum_k h_{i-2k} min(w_k^{r+1}, 1) x_k^{r+1} \tag{11.11}$$

$$w_i^r = tw_i^r (1 - w_i^r) + w_i^r \tag{11.12}$$

$$x_i^r = tx_i^r (1 - w_i^r) + w_i^r x_i^r \tag{11.13}$$

tw_i^r und tx_i^r sind Werte aus der geringer aufgelösten Version, die wieder auf ihre ursprüngliche Grösse hochskaliert werden (11.10 und 11.11). Ihre Ausdehnung gleicht danach wieder derjenigen aus der Splattingphase. Danach erfolgt die Überblendung durch die Koeffizienten $(1-w_i^r)$ und w_i^r. Dadurch werden wie gewünscht bei grossen w_i^r die Koeffizienten der hochauflösenden Version stärker berücksichtigt und nur die Löcher mit den Daten der Pullphase gefüllt.

Speicherbedarf und Kompression

Die benötigte Datenmenge ist im Vergleich zu polygonalen Repräsentationen sehr hoch. Z.B. wird für die (s,t) $Ebene$ eine Auflösung von 32×32 und 256×256 für die (u,v) $Ebene$ gewählt. Bei 6 Würfelseiten und einer Farbtiefe von 24 bit ergibt das $32^2 * 256^2 * 6 * 3 = 1,125GB$. Schon mittels verlustfreier Kompressionsverfahren lässt sich diese Menge um einiges reduzieren, da viele Elemente des Lumigraph teilweise identische Pixeldaten haben. In Verbindung mit JPEG und ähnlichen Algorithmen ist ein Kompressionsfaktor von 200:1 in guter Qualität möglich.

Rekonstruktion von Bildern aus dem Lumigraph

Die einfachste Methode zum Generieren von Bildern aus dem Lumigraph ist Raytracing. Dazu wird ein Strahl von der Kamera durch das Pixel und die beiden Ebenen geschossen. Dieser Strahl wird mit Hilfe des Voxelmodells korrigiert und der Rasterpunkt (i, j, p, q) für die Position (s, t, u', v') ermittelt. Der Koeffizient des Rasterpunktes und seiner Nachbarn wird mit der Basisfunktion B gewichtet und als Pixelfarbe gesetzt.

Die Raytracingoperationen sind ziemlich teuer, so dass sich die Verwendung von hardwarebeschleunigten Verfahren anbietet. Texturemapping wird von den meisten aktuellen Graphikkarten unterstützt und kann dazu verwendet werden, um (u, v) Koordinaten auf ein Flächenelement (i, j) im Raster abzubilden. Dies ist für konstante und auch quadralineare Basisfunktionen möglich.

11.2.6 alternative Verfahren und Fazit

Es existieren eine Reihe weitere Verfahren, um photorealistische Szenen mit Explorationsmöglichkeiten zu präsentieren. Das **Plenoptic Modeling** verwendet dazu mehrere zylindrische Projektionen zwischen denen die Betrachterposition verändert werden kann. Die **Lightfields** gehen in eine ähnliche Richtung wie der Lumigraph, wobei auf zusätzliche geometrische Information verzichtet wird und eine Motion Control Plattform notwendig ist.

All diese Verfahren haben die Gemeinsamkeit, dass der rechnerische Aufwand unabhängig von der Komplexität der Szene und ihrer Lichtverhältnisse ist. Die Qualität des Resultats hängt besonders vom Ausgangsmaterial und der gewählten Auflösung der Ausgabe ab. Der Aufwand steigt mit zunehmender Auflösung leider nichtlinear und stellt einen der grössten Nachteile dieser Visualisierungsmethoden dar. Für eine Leinwandpräsentation in einer annehmbaren Auflösung sind diese Verfahren (noch) nicht geeignet. Zudem sind die Bewegungsmöglichkeiten kaum mit denen einer echten VR Simulation vergleichbar. Im Bildmaterial ist die Entfernung zum Objekt relativ gross und eine Kamerafahrt durch die Szene wäre nur durch Interpolation zwischen vorhandenen Pixeln möglich. Die Folge sind starke Treppen- und Unschärfeeffekte. Interaktionsmöglichkeiten können nur im geringen Masse implementiert werden, da kein komplettes geometrisches Modell vorliegt und Objekte innerhalb der Szene nicht separierbar sind. Typisch sind auch die statischen Lichtverhältnisse. Die Verwendung von Animationen würden im Falle einer naiven Implementierung einen kompletten Lumigraph für jede Animationsphase, bzw. frame bedeuten. Trotz dieser Beschränkungen zeigen diese Techniken besonders dann ihre Stärken, wenn detailreiche, statische Objekte mit nicht - diffusen Oberflächen aus unterschiedlichen Perspektiven präsentiert werden sollen. In etlichen Fällen stellen sie sogar die einzige Möglichkeit dar, einen wirklich photorealistischen Eindruck zu vermitteln.

Literaturverzeichnis

[1] Steven J. Gortler, Radek Grzeszczuk, Richard Szeliski, and Michael F. Cohen. The lumigraph. *Computer Graphics*, 30(Annual Conference Series):43–54, 1996.

[2] Marc Levoy and Pat Hanrahan. Light field rendering. *Computer Graphics*, 30(Annual Conference Series):31–42, 1996.

[3] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics*, 29(Annual Conference Series):39–46, 1995.

Vortrag 12

Bild-Normierung

Andrea Müller

Februar 2004

Institut für Computer Visualistik
Universität Koblenz-Landau
Universitätsstraße. 1, 56070 Koblenz
vorstel@uni-koblenz.de
http://www.uni-koblenz.de/∼vorstel

Abbildung 12.1: Wechselnde Lichtverhältnisse und Bildinformationen an aufeinanderfolgenden Tagen

12.1 Einleitung

In diesem Kapitel, werden unterschiedliche Normierungsverfahren vorgestellt. Alle im weiteren erwähnten Verfahren beziehen sich dabei speziell auf die Normierung von Farben.

Die Farbnormierung spielt in der Bildverarbeitung eine wichtige Rolle. Probleme bei aufgenommenen Bildern können unterschiedlich Ursachen haben. Hierzu gehören unteranderem auch Farb- und Helligkeitsdifferenzen, welche noch näher untersucht werden. Um nun aber weiterhin mit diesen Bildern zu arbeiten, müssen alle auf ein einheitlichen Standard gebracht werden. Dieses erreicht man durch unterschiedliche Normierungsverfahren. Die wichtigstens Einsatzgebiete solcher Normierungsverfahren sind in der industriellen Prüfung, bei der Verarbeitung von Luftbildaufnahmen und ganz konkret in der Medizin.

12.2 Farbnormierung

Als erstes soll für das Verständnis wichtige Begriff der Normierung erläutert werden. Das Verfahren der Normierung dient zur Vereinheitlichung von interessanten Größen, in dieser Ausarbeitung die Normierung der Farben.

Wie schon in der Einleitung erwähnt, werden speziell Farb-und Helligkeitsschwankungen untersucht und gegebenfalls eliminiert. Einwirkung auf die Farbdarstellung können, zum Beispiel, Tageszeiten haben. Dafür werden Verfahren entwickelt, die diese Farbdifferenzen zwischen zwei Bildern beheben können. In Abbildung 12.1 sehen sie ein Beispiel, welches inhaltlich völlig identisch ist, aber durch die unterschiedliche Lichtwirkung unterschiedliche Wahrnehmungen hervorrufen.

Anhand dieses Beispiel erkennt man auch gleich die Notwendigkeit einer Normierung. In der Bildverarbeitung werden zudem nicht nur Bilder miteinander verglichen. Eine Normierung ist auch für die Identifikation und Lokalisierung von Objekten bzw. für die Merkmalsisolierung vonnöten. Aus diesen Gründen muß auch eine Normierung vor einer Analyse durchgeführt werden. Die Normalisierung ansich zählt daher zu den Vorverarbeitungsschritten.

Als Eingabeformat aller betrachteten Normierungsverfahren wird jeweils ein Farbbild genommen. Die Eigenschaften eines solchen Bildes sind in den Formeln 12.1 und 12.2 verdeutlicht. Formel 12.1 beschreibt den Aufbau eines Farbbildes, wobei die zweite Formel dann den eigentlichen Farbvektor für jedes Pixel definiert. Zusätzlich sei noch erwähnt, das alle Berechnungen ausschließlich auf der RGB-Farbraum beziehen. Jeder einzelne dieser Kanäle besteht wiederum aus einem Wertebereich von 0...255 Intensitäten.

$$f = [f_{ij}]_{i=1...M,j=1..N} \tag{12.1}$$

$$f_{ij} = (r_{ij}, g_{ij}, b_{ij})^T \tag{12.2}$$

Um nun mit Bilden zu arbeiten, sollte man nicht nur die Farben eines Objektes betrachten, sondern auch die Umgebungsbeleuchtung und deren Auswirkungen auf das Objekt in die Analyse mit eingeziehen. Wichtige Einflußgrößen sind Beleuchtungsgeometrie und Beleuchtungsfarben. Um diese Faktoren nun besser zu verstehen, werden hier die Wirkungen näher erläutert.

12.2.1 Beleuchtungsgeometrie

Zur Beleuchtungsgeometrie gehören Eigenschaften, die das Verhältnis zur Lichtquelle beeinflussen. Unterschiedliche Farbwahrnehmungen können durch Schattenwurf, veränderte Winkeleinstellung in Bezug auf die Kamera oder die Entfernung von der Kamara entstehen. Daher sollten diverse Berechnungen im Vorfeld mit berücksichtigt werden.

$$\begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} \rightarrow \begin{pmatrix} R_i & S_i \\ G_i & S_i \\ B_i & S_i \end{pmatrix}$$

Diese Größen haben eine Veränderung der Helligkeit zur Folge. Jedes Pixel muß daher mit dem Faktor S_i verrechnet werden. Die Veränderung der Beleuchtungsgeometie wird auch als ein lokales Ereignis beschrieben.

12.2.2 Beleuchtungsfarbe

Im Gegenzug zur Beleuchtungsgeometrie beschreibt die Beleuchtungsfarbe ein globales Ereignis. Die Auswirkung einer Veränderung dieser hat daher nicht nur Einfluss auf eine Region von Pixel, sondern auf das ganze Bild. Die Unterschiede können durch unterschiedliche Größen entstehen. Hierzu können farbige Beleuchtung oder eine natürliche Farbveränderung durch andere Tageszeiten entstehen (Abbildung 12.1). Eine Entfernung solcher Differenzen erfolgt daher auch für jedes Pixel und für jeden Farbkanal durch einen eigenen Faktor α, β, γ.

$$\begin{pmatrix} R_i \\ G_i \\ B_i \end{pmatrix} \rightarrow \begin{pmatrix} R_i & \alpha \\ G_i & \beta \\ B_i & \gamma \end{pmatrix}$$

12.3 Normierungverfahren

Wenn sich die Beleuchtung einer Szene durch unterschiedliche Jahres- bzw.Tageszeiten ändert, sind wir trotzdem in der Lage, Farbe eines bestimmten Objekts als ein und die gleiche zu identifizieren. Das ist auch das Ziel einer maschinellen Bildverarbeitung. Hierzu wird ein Farbkonstanz-Algorithmus verwendet, der die spektrale Farbe $S(x, \lambda)$ unabhängig von der Beleuchtung ermittelt.

Die nun im weiteren vorgestellten Verfahren sind weniger auf die Konsistenz von Farben ausgelegt. Hierbei handelt es sich ausschließlich um datengetrieben Verfahren, die zur Bildvorverarbeitung dienen. Um eine optimale Normierung der Farben zu bekommen, ist es bei einigen der nun folgenden Verfahren notwendig mehreren Durchläufen sicherzustellen. Das erste Verfahren besteht aus einer Normierung auf der lokalen und globalen Ebene, bis zur Konvergenz, das zweite Verfahren ist eine geometrische Interpretation von Farbclustern.

12.3.1 Comprehensive Color Image Normalization (CCN)

Bei diesem Verfahren wird der Farbkanal sowohl lokal als auch global normiert. Das bedeutet, das eine Änderung der Beleuchtungsrichtungen und Beleuchtungsfarben durch die Normalsierung durchgeführt wird. Das Vefahren wird dann solange wiederholt bis es letztendlich konvergiert. Eine Einschränkung wird durch die Oberfläche aller Objekte beschrieben. Die Oberflächen dürfen nur durch die Oberflächeneigenschaften und durch die Parameter der Lichtquelle beschrieben sein. Eine Oberfläche besteht deshalb nur den Lambert'sche Eigenschaften. Das bedeutet, dass keine Reflektion vorhanden sein darf. Im Realen ist diese Vorraussetzung kaum erfüllt, aber trotzdem erreicht man durch dieses Verfahren auch bei natürlichen Szenen gute Ergebnisse. Fehler entstehen durch Streulicht, das von einem anderen Objekt kommen kann. Außerdem noch in Schattenbereichen, da hier verstärkt Interreflektionen auftreten. Die Anforderungen an ein Bild können mit Filtern in der Vorverarbeitung erfüllt werden.

Problem: Beleuchtung ist maßgeblich an der Farbgebung beteiligt
Ziel: Normierung des Bildes
Lösung: Algorithmus, die Beleuchtungsabhängigkeiten beseitigt

Die Idee des Algorithmus, welcher iterativ und zweistufig aufgebaut ist, liegt darin zuerst jedes Pixel isoliert zu normieren (Formel 12.4). Hierdurch werden Intensitätsschwankungen beseitigt. Wichtig ist zu dem noch, daß jedes Pixel in der Summe zu eins normiert wird. Dieser Teil des Algorithmus beschäftigt sich mit dem lokalen Aspekt. Im zweiten Teil des Algorithmus wird der globale Teil normiert, was wiederum die Farbdifferenzen eliminiert. Jeder Farbkanal wird deswegen so normiert, dass die Summe der Farbkomponenten im Knanal sich zu einem Drittel der Pixelanzahl addiert (Formel 12.5). Dieser Algorithmus wird nun solange angewendet bis keine Veränderungen mehr in den Ergebnissen vorgenommen wird oder ein vordefinierter Schwellwert unterschritten wurde (Formel 12.7). Bei mehreren Testläufen wurde zudem noch festgestellt, dass dieser Algorithmus in weniger als fünf Schritten terminierten kann.

Die Eingabe ist ein Farbbild (Definition 12.1) bestehend aus Farbvektoren (Dfinition 12.2) auf das der Algorithmus angewandt wird. Als erstes wird die Summe aller Farbkanäle berechnet.

$$S_{ij} := r_{ij}^{(t)} + g_{ij}^{(t)} + b_{ij}^{(t)} \tag{12.3}$$

Mit dem daraus errechneten Wert S_{ij} wird nun jedes Pixel normiert.

$$r_{ij}^{(t+1)} = r_{ij}^{(t)} / S_{ij} \qquad g_{ij}^{(t+1)} = g_{ij}^{(t)} / S_{ij} \qquad b_{ij}^{(t+1)} = b_{ij}^{(t)} / S_{ij} \tag{12.4}$$

Das Ergebnis nach diesen Schritten sieht man in Abbildung 12.4

Im zweiten Teil, der globalen Bearbeitung, wird ein gewichteter Mittelwert für jeden Kanal berechnet. (Berechnung nur für de Rotkanal beschrieben, analog Grün und Blau)

$$\hat{r} = \frac{3}{MN} \sum_{i=1}^{M} \sum_{j=1}^{N} r_{ij}^{(t)} \tag{12.5}$$

Als nächstes werden dann die einzelnen Komponeten ebenfalls normiert. Die Anwendung erfolgt dann aber schon auf das bereits veränderte Bild, also eine Iterationstufe höher.

$$r_{ij}^{(t+2)} = r_{ij}^{(t+1)} / \hat{r} \tag{12.6}$$

Dann wird der Algoritmus so lange wiederholt bis keine weitere Veränderung der Werte mehr eintritt, oder der Schwellwert unterschritten wird:

$$\epsilon = \sum_{i=1}^{M} \sum_{j=1}^{N} \left(\left(r_{ij}^{(t+2)} - r_{ij}^{(t)} \right)^2 + \left(g_{ij}^{(t+2)} - g_{ij}^{(t)} \right)^2 + \left(b_{ij}^{(t+2)} - b_{ij}^{(t)} \right)^2 \right) \tag{12.7}$$

Das Ergebnis nach diesen Schritten sieht man in Abbildung 12.5

Ein weiterer Effekt dieser Normalisierung ist die Verringerung der Anzahl der verwendeten Farben.

Beispiel

In Abbildung 12.6 kann man sehen, dass sowohl ein Schatten als auch farbige Beleuchtung durch die Verwendung des CCN-Algorithmus eliminiert werden.

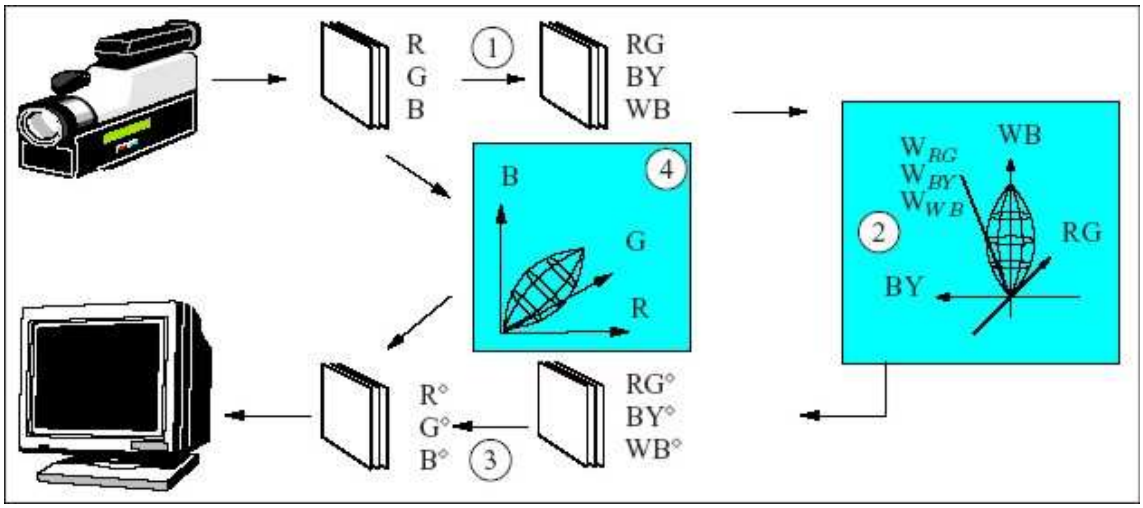


Abbildung 12.2: Farbnormierung

12.3.2 Farbnormierung durch Rotation

Bei der Farbnormierung durch Rotation kann man zwei verschieden Methoden verwenden. Im weiteren Verlauf dieser Arbeit wird noch genauer darauf eingegangen. Zum einem gibt es die Rotation in RG,BY,WB nach Pomierski bzw. Rotation im RGB nach Paulus und zum anderen gibt es die Whitening Transformation nach Fukunaga.

Im Allgemeinen sollte man bei der Normierung durch Rotation folgende Ausgangspunkte wissen, die bei beiden Verfahren von Bedeutung sind.

Bei diesen Verfahren erfolgt eine Analyse über die Clustereigenschaften eines Farbraums. Bei der Analyse wird angenommen, daß sich die Farbwerte des Bildes im Mittel vektoriell zu Grau addieren lassen - wird in der Literatur oft auch als 'graue Welt' bezeichnet. Wenn das bei einem Bild aber nicht der Fall ist, dann kann man eine farbige Beleuchtung annehmen. Bei diesen Methoden will man die Bilder wieder in die 'graue Welt' normieren. Hierfür benötigt man die Berechnung der Hauptachse des Farbclusters die mit Hilfe eines neuronalen Netze erfolgt (oder alternativ mit der Eigenvektorbestimmung der Kovarianzmatrix). Der daraus entstehend Richtungsvektor wird benötigt um das Farbcluster zu rotieren, so dass die Hauptachse des Clusters auf der WB-Achse des (RG, BY,WB)-Raumes fällt. Anschließend wird dann das Farbcluster auf dieser Achse so skaliert, dass es dann letztendlich wieder den ganzen Farbraum abdeckt.

Die Transformation eines Farbvektors erfolgt durch eine Transformationsmatrix A, die durch empirische Test aufgestellt wurde.

$$\boldsymbol{A} = \begin{pmatrix} 0.5 & -1 & 0.5 \\ -0.875 & 0 & 0.875 \\ 1.5 & 1.5 & 1.5 \end{pmatrix}$$

$$\tilde{\boldsymbol{f}} = \begin{pmatrix} RG \\ BY \\ WB \end{pmatrix} = \begin{pmatrix} 0.5 & -1 & 0.5 \\ -0.875 & 0 & 0.875 \\ 1.5 & 1.5 & 1.5 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Den Ablauf des Verfahrens kann man anhand der Abbildung 12.2 verdeutlichen.

Die Schritte die mit 1 und 3 gekennzeichnet sind, beinhalten die Farbtransformationen durch die Matrix $\boldsymbol{A}$ und $\boldsymbol{A}^{-1}$. Schritt 2 beschreibt die Suche nach der Hauptachse und dann die anschließende Rotation auf die WB-Achse. Im weitern kann man dieses Prinzip auch in anderen Farbräumen anwenden, welches zur Folge hat, daß man zusätzlich Umrechnungen in einen neuen Farbraum einspart. Hat aber dann auch zur Folge, dass andere Normierungsergebnisse entstehen können.

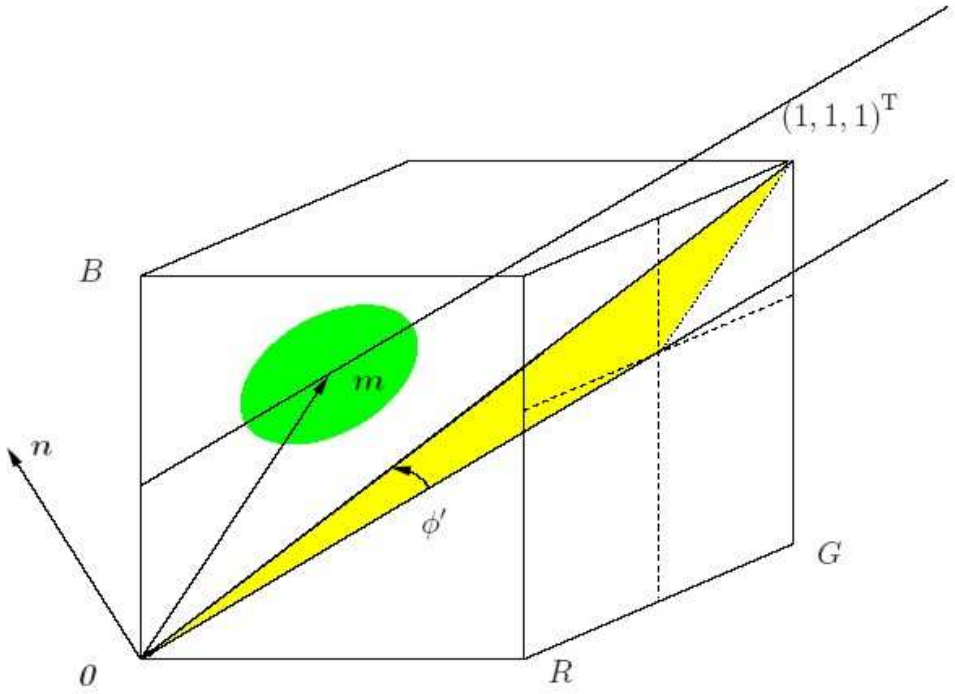


Abbildung 12.3: RGB-Würfel

Rotation im RGB-Farbraum

Die Abbildung 12.3 soll zunächst das Verfahren der Rotation verdeutlichen. In der Grafik erkennt man die Unbunt-Achse. Diese verläuft in Richting (1,1,1). Im RGB-Würfel grün dargestellt ist das Farbcluster des Bildes, welches nun auf die Unbunt-Achse verschoben werden soll. Zusätzlich ist die Hauptachse des Clusters eingezeichnet.

Das Verfahren der Rotation im RGB-Raum besteht nun darin, das gewonnene Farbcluster auf die Hauptachse zu projizieren. Dazu wird eine Normale $\boldsymbol{n}$ durch den Ursprung der Ebene, die durch die Hauptdiagonale des Würfels und die Hauptachse des Clusters aufgespannt wird, berechnet. Der Vektor $\boldsymbol{m}$ beschreibt die Verschiebung des Clusters in den Ursprung der Ebene. Als nächstes wird das Cluster nun um diese Normale rotiert (Winkel ϕ')so dass, die Hauptachse des Würfels und die des Clusters zusammenfallen. Danach wird der Mittelpunkt des Clusters auf der unbunten Achse verschoben bis dieser mit dem Mittelpunkt dieser Achse identisch sind. Als letztes wird dann das Cluster mit einem Skalierungsfaktor multipliziert um somit den RGB-Würfel bestmöglichst auszufüllen. Alle überstehenden Farben werden dann abgeschnitten. Der Mittelwert des Cluster liegt somit auf der unbunten Achse des Würfels. Als Ergebnis erhält man dann ein farblich verändertes Bild (heller bzw.dunkeler) als das Ausgangsbild und Farbstiche (z.B. gelb) verschwinden.

Whitening-Transformation

Die zweite Methode einer Farbnormierung durch Rotation, ist die Whitening-Transformation. Diese Methode ist eine orthogonale Transformation der Hauptkomponenten des Farbclusters auf Eigenvektoren. Die Hauptkomponeneten sind dabei Linarkombinationen der ursprünglichen Werte. Um die Berechnung korrekt durchführen zu können, sei noch erwähnt, dass zur Rechnung der größte Eigenvektor verwendet werden soll. Eigenvektoren sind die Hauptkomponenten des Farbclusters. Dabei entspricht der Eigenvektor dem Eigenwert. Ein Grund für die Verwendung des größte Eigenvektor, ist die Tatsache, das damit diesem die Verzerrung der Farben umgangen werden kann.

Der Anfang der Berechnung erfolgt, auch bei dieser Rotation, analog der Rotation im RGB-Farbraum. Bei dieser Methode wird aber die Hauptdiagonale nicht um eine Winkel ϕ' rotiert, sondern verfolgt vordefinierte Rotationsschamatas. Bei der Whitening-Transformation wird das Cluster um 45 Grad zur positiven Rot-Richtung und um 45 um die Blau-Achse rotiert. Als letztes wird jeder Vektor um $\frac{1}{2}\left(N_R, N_G, N_B\right)^T$ verschoben. N beschreibt dabei die Anzahl von möglichen Werten eines Farbkanals. Über- und Unterläufer werde werden dann auf die Kanten des RGB-Würfels skaliert.

Als Ergebnis erhält man ein farbnormalisiertes Bild, dessen Mittelwert ebenfalls auf der Unbunten Achse zu finden ist.

Abbildung 12.4: Ergebnis nach der lokalen Normierung

Abbildung 12.7 zeigt zwei Ergebnisbilder der beiden Transformationen. Abbildung 12.8 zeigt ebenfalls Ergebnisbilder dieser Normalisierungen.

12.4 Vergleich

Abbildung 12.9 verdeutlicht die unterschiedlichen Ergebnisse. Zusätzlich kann man dort noch die unterschiedlichen Farbräume erkennen. Bei dem CCN-Algorithmus muß man den Schwellwert geeignet wählen, sonst geht ein Großteil durch die Normierung verloren. Am Ende hat man dann nur noch ein fast graues Bild. Am besten eignet sich die Methode der Rotation im RG,BY,WB-Raum. Hier werden dann noch die besten Ergebnisse erzielt. Ideal ist diese Methode deshalb für die Betrachtung der Bilder. Die Whitening-Transformation eignet sich am besten für eine Weiterverarbeitung der Bilder. Hier können leicht verfälschte Farben entstehen.

12.5 Grenzen der Methoden und Ausblick

Die meisten Methoden eliminieren entweder nur Farbabhängigkeiten durch Beleuchtungsgeometrie oder Abhängigkeiten durch Beleuchtungsfarbe.
Der CCN-Algoritmus dagegen kann zwar beide Farbverfälschungen beheben, doch ist dieser durch sein iteratives Verfahren sehr komplex und zeitaufwendig.
Neue Methoden werden deshalb zur Zeit noch erforscht.

12.6 Beispiele

Literaturverzeichnis

[1] Dietrich Paulus Detlev Droege, Vinh Hong. Farbnormierung auf langen bildfolgen. 2003.

[2] Graham D. Finlayson, Bernt Schiele, and James L. Crowley. Comprehensive colour image normalization. *Lecture Notes in Computer Science*, 1406:475–490, 1998.

[3] D. Paulus, L. Csink, and H. Niemann. Color cluster rotation. 1998.

[4] Dietrich Paulus. *Aktives Bildverstehen*. Der andere Verlag, Osnabrück, 2001. Habilitationsschrift in der Praktischen Informatik, Universität Erlangen-Nürnberg, Mai 2000.

[5] Dietrich Paulus. Bildverarbeitung. pages 51–56, 2001/2002.

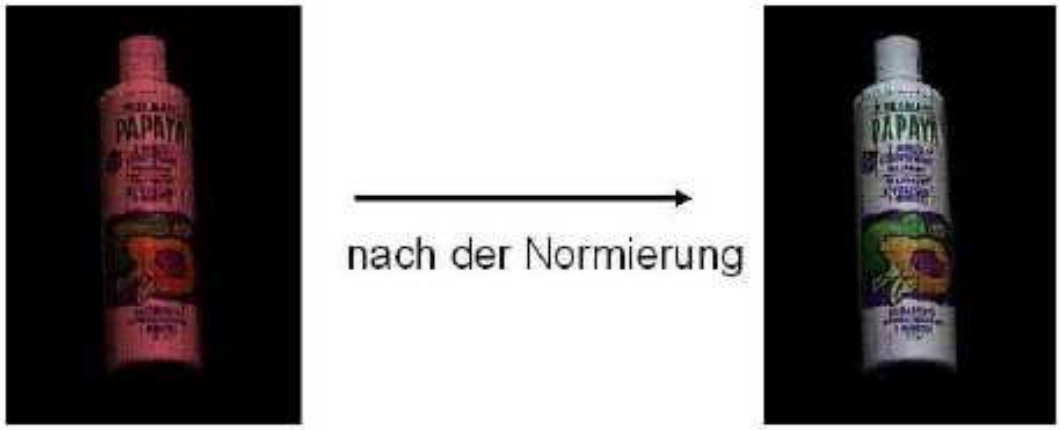


Abbildung 12.5: Ergebnis nach der globalen Normierung

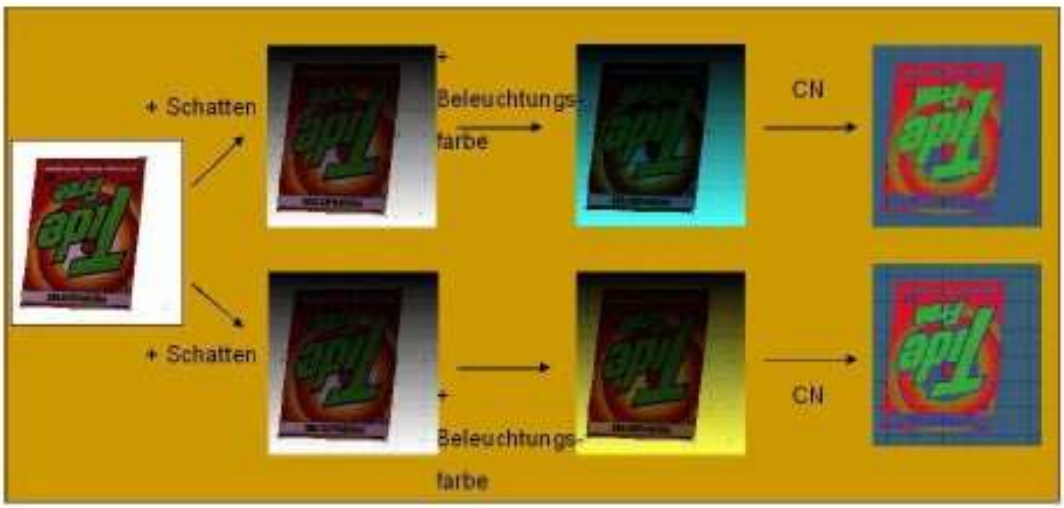


Abbildung 12.6: Beispielnormierung

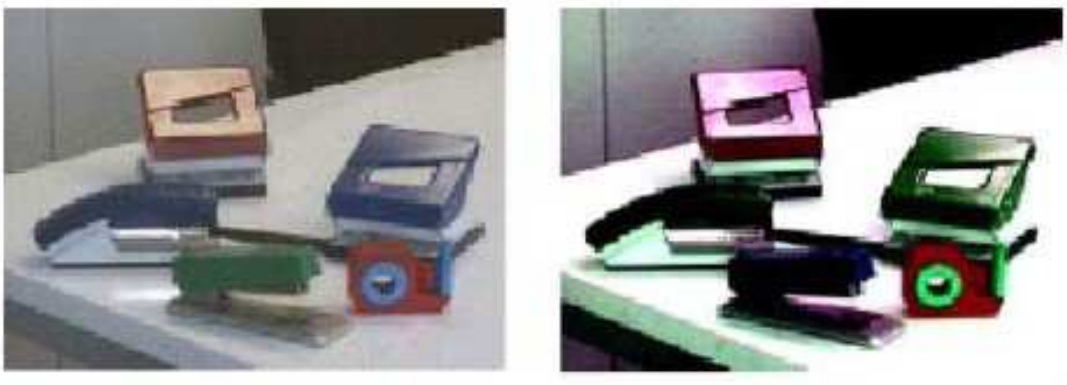


Abbildung 12.7: Ergebnis einer Farbrotationim RGB + Whitening-Transformation

Abbildung 12.8: Ausgangsbild + Rotation in RGB + Whitening-Transf. + Rotation n.Pomierski

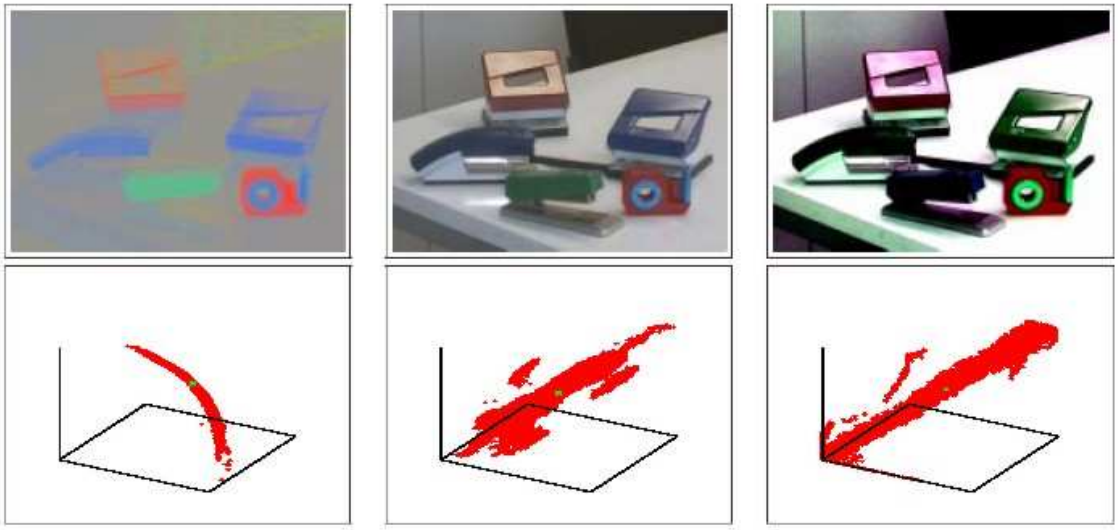


Abbildung 12.9: CCN + Farbrotation + Whitening-Transf.

[6] Bozena Zdunczyk. Bildnormierung. In *Methoden der Bildverarbeitung für (Uni-)Webcam-Bilder*, Koblenz, November 2003.