Diplomarbeit

OpenGL-basierte interaktive Visualisierung von generalisierten Lichtfeldern

Dirk Wippermüller

Betreuer: Dipl.Ing. Jan-Friso Evers-Senne

Institut für Informatik und Praktische Mathematik der
Christian-Albrechts-Universität zu Kiel
Multimediale Systeme zur Informationsverarbeitung

Prof. Dr.-Ing. Reinhard Koch

13. August 2002

Inhaltsverzeichnis

Kapitel 1

Einleitung

Der traditionelle Ansatz zur Darstellung beliebiger Ansichten einer dreidimensionalen Szene besteht darin, eine Szene mit einem 3D-Graphiksystem von Hand zu erstellen. Dazu werden verschiedene Knoten und Kanten erstellt, meist vereinfacht durch geometrische Primitive, und zusätzlich einige Parameter wie Materialien und einer Menge von Lichtquellen eingestellt. Dann kann man beliebige Ansichten der erstellten Szene rendern.

Es gibt mehrere Möglichkeiten des Renderns. Kommt es nicht auf Geschwindigkeit sondern auf die Darstellungsqualität an, wählt man das sogenannte *Raytracing*, um beliebige Ansichten zu erzeugen. Das Verfahren bietet hohe Qualität und ist unabhängig von der Komplexität der Szene, aber sehr langsam. Eine andere Möglichkeit ist die Benutzung von Hardware-beschleunigten Graphikschnittstellen wie OpenGL. Auch hier muss die Szene per Hand erstellt werden, aber die Rendergeschwinigkeit ist sehr viel höher als die des Raytracings. Es können allerdings nur acht Lichtquellen benutzt werden, dadurch können reale Lichteffekte nur ungenügend modelliert werden. Außerdem ist die Geschwindigkeit der Darstellung abhängig von der Komplexität der Szene.

Ein anderer Ansatz, um neue Ansichten einer Szene zu erzeugen, ist *bildbasiertes Rendering*. Dabei wird eine große Anzahl Ansichten aus vielen verschiedenen Richtungen aufgenommen. Jede Ansicht kann als ein Bündel von Lichtstrahlen betrachtet werden, die von jedem Punkt des Bildes zum Kamerazentrum

führen. Erzeugt man neue Ansichten der Szene, müssen Lichtstrahlen, die in den aufgenommenen Bildern nicht berücksichtigt sind, aus vorhandenen Lichtstrahlen interpoliert werden.

Oft wird von realen Objekten angenommen, dass ein Punkt der Oberfläche den selben Farbwert besitzt, egal aus welcher Richtung auf das Objekt geschaut wird. Treten aber Spiegeleffekte auf, ist dieser Ansatz falsch. Der Farbwert wird sich zusammen mit der Blickrichtung ändern, aber im etwa gleichen Verhältnis, d.h. ändert man die Blickrichtung nur wenig, ändert sich auch der Farbwert nur wenig. Zwei Lichtstrahlen haben also dann ähnliche Farbwerte, wenn ihre Richtung ähnlich ist und ihr Schnittpunkt nahe der Oberfläche des Objektes liegt.

Der bildbasierte Ansatz bietet mehrere Vorteile:

- Keine Handarbeit bei der Erstellung
- Der Darstellungs-Algorithmus verbraucht nur relativ wenige Ressourcen, das System wird echtzeitfähig.
- Die Berechnung einer neuen Ansicht ist unabhängig von der Komplexität einer Szene.
- Es existiert keine Beschränkung bezüglich der Anzahl an Lichtquellen einer Szene.
- Spiegelungen und Reflexionen sowie sämtliche Lichteffekte werden berücksichtigt (ähnlich Raytracing).
- Die vorhandenen Bilder einer Szene können digitalisierte Fotografien oder künstlich erzeugt worden sein.

Andererseits kann die Szene nicht von allen Seiten betrachten werden, sondern nur aus Richtungen, von denen Bilder vorhanden sind.

Es gibt mehrere Ansätze des bildbasierten Renderings. Der ursprüngliche Lichtfeld-Ansatz [LP96] berücksichtigt keine Tiefeninformationen der Szenengeometrie, sondern setzt eine planare Szene voraus. Dadurch kommt es bei der Interpolation von Bildern einer nicht-planaren Szene zu störenden Effekten (*blurring*). Ein

anderer Ansatz, genannt Lumigraph [GGRC96], nutzt Tiefeninformationen, die einzelnen Kameras müssen aber kalibriert werden.

Der hier beschriebene neue Ansatz des bildbasierten Renderings ermöglicht es, Bilder einer handelsüblichen Handkamera zu nutzen, um neue Ansichten einer Szene zu erzeugen. Dazu werden die Kameraparameter der einzelnen Bilder berechnet und für jedes Bild eine Tiefenkarte erstellt. Aus den Tiefenkarten wird eine Szenengeometrie angenähert. Diese wird mit Hilfe der heutigen Techniken texturiert und Echtzeit-fähig.

Kapitel 2 erläutert die Grundlagen der Arbeit. Darin wird zunächst eine allgemeine Lichtfelddarstellung beschrieben, bevor diese Darstellung in einem ersten praktischen Ansatz durch ein regelmäßiges Gitter angenähert wird. Diese Restriktion wird im weiteren Verlauf gelockert.

In Kapitel 3 wird aus mehreren Aufnahmen einer Szene die Tiefe der Szenengeometrie geschätzt und die Kameraparameter ermittelt.

Kapitel 4 behandelt die eigentliche Darstellung neuer virtueller Ansichten der Szene mittels OpenGL.

In Kapitel 5 werden schließlich die Ergebnisse der Arbeit vorgestellt.

Kapitel 2

Grundlagen

2.1 Lichtfelder

Lichtfelder stellen einen Ansatz der bildbasierten Visualisierung dar. Mit diesem Verfahren generierte Ansichten einer Szene bestehen aus vielen unterschiedlichen Teilen der ursprünglichen Bilder und besitzen keinerlei Tiefeninformationen der Szenengeometrie.

2.1.1 Repräsentaion

Die *plenoptische Funktion* von Adelson und Bergen [AB91] definiert die Strahlung aller Punkte x, y, z eines Raumes in alle möglichen 2D-Richtungen, parametrisiert durch zwei Winkel, über die Zeit:

$$f : (\theta, \phi, x, y, z, t) \mapsto i, \tag{2.1}$$

mit $x, y, z, t \in \mathbb{R}$ Ort, $\theta, \phi \in \mathbb{R}$ Richtung, i Farbwert.

Im Folgenden werden nur statische Szenen betrachtet, die 6D-Funktion reduziert sich also auf 5D. Diese 5D-Darstellung lässt sich auf 4D reduzieren, wenn ein freier Raum vorliegt, da die Strahlung eines Punktes entlang einer Linie konstant bleibt, solange der Lichtstrahl nicht blockiert wird.

Im Folgenden wird ein Strahl parametrisiert mittels seiner Schnittpunkte mit zwei beliebig angeordneten parallelen Ebenen mit Koordinaten (u, v) für die erste

und (s, t) für die zweite Ebene. Damit wird ein gerichteter Strahl definiert durch die Verbindung eines Punktes auf der uv-Ebene mit einem Punkt auf der st-Ebene (Abb. 2.1). Es gilt o.B.d.A. $u, v, s, t \in [0, 1]$. Die uv-Ebene repräsentiert die Kameraebene, die st-Ebene die Bildebene.

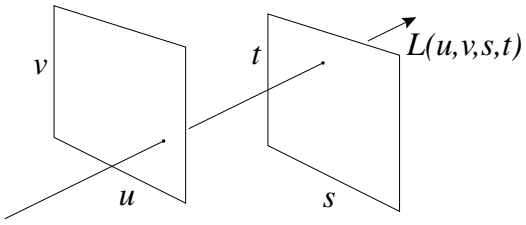


Abbildung 2.1: Lichtstrahl-Parametrisierung.

2.1.2 Bildaquisition

Das theoretische Lichtfeld-Modell besitzt ein unendlich dichtes Kamerafeld, bei der Bildaquisition von natürlichen Bildern wird dieses Modell durch die einzelnen Kamerapositionen auf der uv-Ebene in einem regelmäßigen Gitter angenähert. Sinnvollerweise diskretisiert man das Raster der st-Ebene entsprechenend der Pixelgröße der aquirierten Bilder. Abb. 2.2 zeigt eine Veranschaulichung des entstehenden 4D-Lichtfeldes. Jedes Bild an einer Kameraposition der uv-Ebene repräsentiert alle Strahlen der st-Ebene, die an diesem Punkt eintreffen. Dabei ist zu beachten, dass eine perspektivische Scherung erfolgen muss, um alle gewonnenen Bilder im gleichen Maßstab zu erhalten (Abb. 2.3).

2.1.3 Darstellung

Bei der Generierung einer neuen Ansicht wird durch jeden Pixel der Bildebene ein Strahl geschossen. Schneidet der Strahl die uv-Ebene in einem exakten Kamerapunkt, so kann für den Pixel ein exakter Farbwert ermittelt werden. Anderenfalls werden die Kameras bestimmt, die am nähesten sind und damit am stärksten zu der neuen Ansicht für das aktuelle Pixel beitragen (Abb. 2.4). Der neue Farbwert

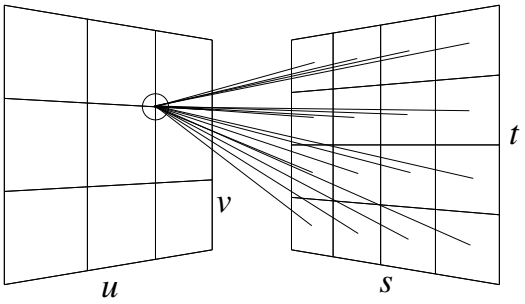


Abbildung 2.2: Lichtfeld-Veranschaulichung.

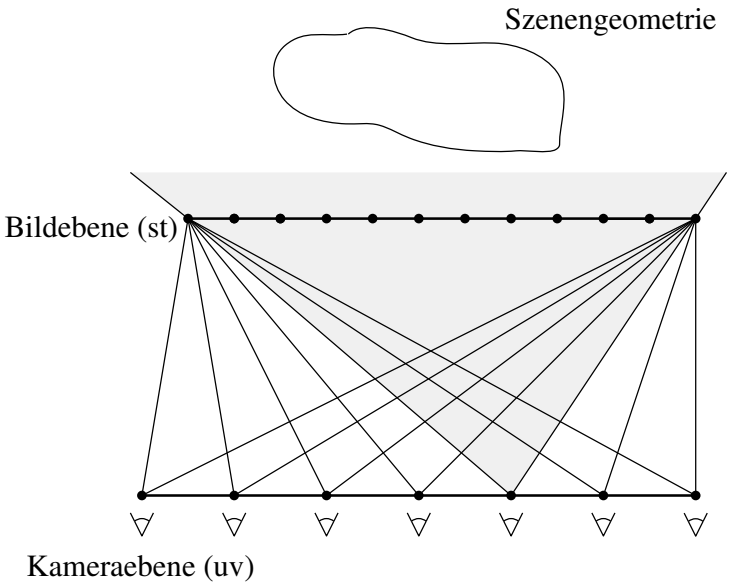


Abbildung 2.3: Perspektivische Scherung.

wird durch Interpolation der korrespondierenden Pixel der bestimmten Kameras gewonnen.

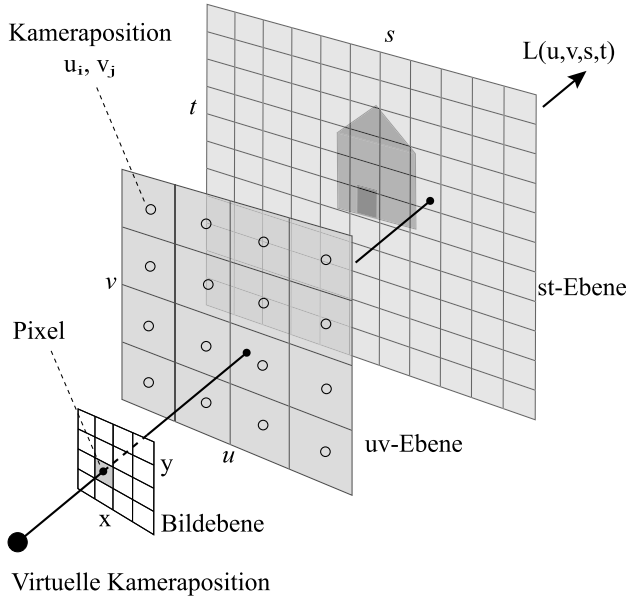


Abbildung 2.4: Rückprojektion.

Abb. 5.7 zeigt einen 2D-Ausschnitt der st- und uv-Ebene. Ist die Szenengeometrie planar und liegt nahe der Bildebene wie bei $L_1(u, s)$, was der Annahme entspricht, wird korrekt interpoliert ziwschen den Farbwerten $L(u_1, s_5)$ und $L(u_2, s_5)$. Bei einer Abweichung von der Annahme wie in $L_2(u, s)$ werden die beiden Farbwerte $L(u_1, s_{10})$ und $L(u_2, s_{10})$ interpoliert. Diese stellen aber nicht den gleichen Punkt innerhalb der Szene dar, der zweite Punkt ist verschoben. Es kommt zu einer verschwommenen Interpolation mit einem überlagerten verschobenen Bild (*blurring*, *ghosting*). Soll die Interpolation der Bilder genauer sein, muss die Tiefe der Szenengeometrie berücksichtigt werden.

Die folgenden Abschnitte der Grundlagen wurden von [Woe01] übernommen. Sie führen zu einer grundlegenden Definition der P-Matrix und der Epipolargeometrie.

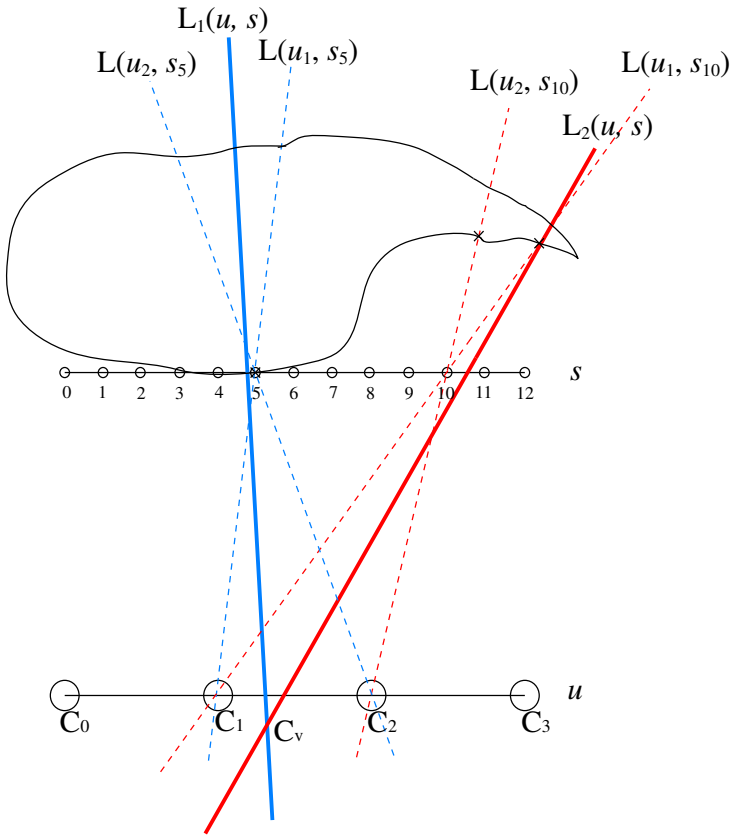


Abbildung 2.5: Nichtplanare Szenengeometrie.

2.2 Projektive Geometrie

Im Raum der projektiven Geometrie bieten sich elegante Möglichkeiten der Darstellung von *perspektivischen Projektionen*. Der Vorteil der projektiven Geometrie gegenüber den anschaulichen euklidischen Koordinaten besteht darin, dass sich die Projektion als eine lineare Matrixoperation in homogenen Koordinaten ausdrücken lässt.

2.2.1 Der projektive Raum

Der n-dimensionale *projektive Raum* $\mathbb{P}^n$ kann mathematisch als die Menge aller eindimensionalen Unterräume eines $(n+1)$-dimensionalen Vektorraums $V^{(n+1)}$ verstanden werden. Diese Unterräume nennt man auch *Hyperebenen*. Anschaulich kann man sich dies bezüglich $\mathbb{P}^2$ als den Raum der Geraden durch den Ursprung $\mathbb{R}^3$ vorstellen.

Obwohl die Elemente eines projektiven Raumes mathematisch formal Hyperebenen sind, nennt man sie Punkte. Ein Punkt x ist in einem n-dimensionalen projektiven Raum $\mathbb{P}^n$ somit durch einen Vektor der Länge $(n+1)$ gegeben.

$$x \in \mathbb{P}^n : x = [x_1, \ldots, x_{n+1}] \tag{2.2}$$

Der Nullvektor sei nicht Element von $\mathbb{P}^n$, es dürfen also nicht alle Elemente x_i gleichzeitig null sein. Die Koordinaten des Punktes x nennt man *projektive Koordinaten* oder für $x_{n+1} = 1$ auch *homogene Koordinaten*.

Da die Punkte eines euklidischen Raumes $\mathbb{R}^n$ in einen projektiven Raum $\mathbb{P}^n$ höherer Dimension eingebettet sind, ergibt sich eine Besonderheit für die Gleichheit von Punkten.

Definition 1 (Gleichheit projektiver Koordinaten) *Zwei Punkte $x, y \in \mathbb{P}^n$ nennt man genau dann 'gleich', wenn es ein Skalar $\lambda \neq 0$ gibt, so dass für alle $1 \leq i \leq (n+1)$ gilt:*

$$x_i = \lambda y_i \tag{2.3}$$

Die projektiven Koordinaten eines Punktes sind also nur bis auf einen skalaren Faktor festgelegt, so dass ein euklidischer Punkt in projektiven Koordinaten nicht

eindeutig festgelegt ist, sondern sich mit verschiedenen Koordinaten darstellen lässt. Die *Gleichheit* bezieht sich also nicht auf die Werte seiner Koordinaten sondern vielmehr auf den mit ihm im euklidischen Raum korrespondierenden Punkt (sofern dieser existiert) bzw. die Richtung des Vektors.

Bei mit einer Tilde (z.B. $\tilde{p}$) gekennzeichneten Buchstaben handelt es sich im Folgenden um die Darstellung in homogenen Koordinaten.

2.2.2 Zusammenhang zwischen euklidischen und homogenen Koordinaten

Für jeden Punkt eines euklidischen Raumes gibt es also eine Menge von *gleichen* projektiven Koordinaten, die sich nur durch einen skalaren Faktor unterscheiden. In späteren Betrachtungen werden nur $\mathbb{P}^2$ und $\mathbb{P}^3$ benutzt. Für diese Räume wird der Zusammenhang zwischen euklidischen und homogenen Koordinaten im Folgenden dargestellt.

2.2.2.1 Von euklidischen zu homogenen Koordinaten

Ein Raumpunkt in euklidischen Koordinaten $P_e \in \mathbb{R}^3$ lässt sich als ein Punkt im projektiven Raum $P_h \in \mathbb{P}^3$ darstellen.

$$P_e = \begin{pmatrix} x_e \\ y_e \\ z_e \end{pmatrix} \tag{2.4}$$

$$P_h = \lambda \begin{pmatrix} x_e \\ y_e \\ z_e \\ 1 \end{pmatrix} \tag{2.5}$$

Als beliebigen skalaren Faktor $\lambda \neq 0$ kann man z.B. $\lambda = 1$ setzen, wodurch man die homogenen Koordinaten erhält, indem alle Elemente x, y, z übernommen werden und das zusätzliche Element der projektiven Koordinate auf 1 setzt.

Analog wird ein Raumpunkt $P_e \in \mathbb{R}^2$ in homogenen Koordinaten wie folgt dargestellt.

$$P'_e = \begin{pmatrix} x_e \\ y_e \end{pmatrix} \tag{2.6}$$

$$P'_h = \lambda \begin{pmatrix} x_e \\ y_e \\ 1 \end{pmatrix} \tag{2.7}$$

Man beachte, dass für jedes $\lambda \neq 0$ der gleiche Punkt P dargestellt wird, die Umwandlung also unabhängig von der Wahl von λ ist. Für $\lambda = 0$ liegt der Spezialfall des Nullvektors $(0, 0, 0)^T$ vor. Dieser ist nach Definition in Abschnitt 2.2.1 jedoch nicht im projektiven Raum enthalten.

2.2.2.2 Von projektiven zu euklidischen Koordinaten

Bei der Umwandlung von projektive in euklidische Koordinaten muss man unterscheiden, ob der Wert des letzten Elements w einer projektiven Koordinate P_h null ist oder nicht.

Sei $\mathbb{P}^n_{he}$ die Menge der projektiven Koordinaten mit $w \neq 0$ aus $\mathbb{P}^n$.

$$\mathbb{P}^n_{he} = \{p \in \mathbb{P}^n | p = (x_1, \ldots, x_n, w)^T \wedge w \neq 0\} \tag{2.8}$$

Dann ergibt sich die Abbildung eines Punktes $P_h \in \mathbb{P}^3_{he}$ in euklidische Koordinaten wie folgt:

$$P_h = (x_h, y_h, z_h, w)^T \wedge P_h \in \mathbb{P}^3_{he} \Leftrightarrow P_e = \begin{pmatrix} \frac{x_h}{w} \\ \frac{y_h}{w} \\ \frac{z_h}{w} \end{pmatrix} \tag{2.9}$$

Und für $P_h \in \mathbb{P}^2_{he}$ ergibt sich analog:

$$P'_h = (x'_h, y'_h, w)^T \wedge P_h \in \mathbb{P}^2_{he} \Leftrightarrow P'_e = \begin{pmatrix} \frac{x'_h}{w'} \\ \frac{y'_h}{w'} \end{pmatrix} \tag{2.10}$$

Die obigen Formeln ergeben sich aus der Gleichheit der projektiven Koordinaten $(x, y, z, w)^T$ und $(x/w, y/w, z/w, 1)^T$ (bzw. $(x', y', w')^T$ und $(x'/w', y'/w', 1)^T$)

nach Definition 1 zusammen mit der Umkehrung der Gleichungen in Abschnitt 2.2.2.1.

Im Fall $w = 0$ lässt sich der Punkt nicht direkt in euklidischen Koordinaten darstellen.

2.3 Kameramodelle

2.3.1 Lineares Kameramodell

In diesem Abschnitt wird die Abbildung der Szene auf den Sensor der Kamera quantitativ beschrieben. Es wird davon ausgegangen, dass jeder Punkt der Szene in genau einen Bildpunkt abgebildet wird, es also z.B. keine Spiegelungen im Bild gibt.

Zunächst wird der Begriff *kamerazentriert* eingeführt.

Definition 2 (kamerazentriert) *Ein orthonormales Weltkoordinatensystem sei genau dann kamerazentriert, wenn es parallel zu seiner (X, Y)-Ebene bei $Z = 1$ das Kamerasystem mit gleichem Maßstab aufspannt.*

Das Weltkoordinatensystem ist also in seiner Lage auf die Kamerakoordinaten normiert, man spricht auch von einer *normierten* Kamera.

Für ein kamerazentriertes Koordinatensystem ergibt sich eine besonders einfache Abbildungsgleichung, die im Folgenden beschrieben wird.

In einem kamerazentrierten Koordinatensystem ergibt sich eine einfache Beziehung zwischen den dreidimensionalen Weltkoordinaten eines Punktes $P = (x, y, z)$ der Szene und den Bildkoordinaten seiner Abbildung $p = (u, v)$ in euklidischen Koordinaten.

$$\frac{f}{z} = \frac{u}{x} = \frac{v}{y} \tag{2.11}$$

$$u = \frac{x}{z} \quad \text{sowie} \quad v = \frac{y}{z} \tag{2.12}$$

Man beachte, dass das Tripel $(u, v, 1) \in \mathbb{R}^3$ sowohl die Koordinaten von p sind, als auch den Richtungsvektor des Strahls s vom Projektionszentrum C zu

P darstellt, welcher (unter Vernachlässigung der entgegengesetzten Orientierung) den Lichtstrahl in der Kamera repräsentiert.

2.3.2 Reales Kameramodell

Eine reale Kamera besteht aus einer Optik und einem lichtempfindlichen Sensor. Der Sensor (z.B. CCD-Chip) liegt in der Bildebene und tastet sie diskret und üblicherweise mit einem regelmäßigen Gitter ab. Er liefert ein Bild der Szene in Pixelkoordinaten.

Das Linsensystem der Optik verursacht lediglich eine Skalierung des Bildkoordinatensystems. Das *Pixelkoordinatensystem* ist somit gegenüber dem Kamerakoordinatensystem im Allgemeinen skaliert. Außerdem ist das Kamerakoordinatensystem gegenüber dem Weltkoordinatensystem meist translatiert und rotiert.

2.3.3 Allgemeines projektives Kameramodell

Mit der Kalibriermatrix K der Kamera wird die Transformation vom Kamerakoordinatensystem in das Pixelkoordinatensystem beschrieben. Allgemein kann die Kamera gegenüber dem Weltkoordinatensystem translatiert, rotiert und skaliert sein. Mit den *äußeren Kameraparametern* werden Translation und Rotation beschrieben. Die Skalierung entspricht einer Änderung des Fokus der Kamera und gehört zu den *inneren Kameraparametern*.

2.3.3.1 Äußere Kameraparameter R und T

Jede der Kameras nimmt die Szene aus einer unterschiedlichen Perspektive auf. Sie ist eindeutig bestimmt durch

- ihre **Lage**
  mit dem Translationsvektor T,
- ihre **Orientierung**
  durch die 3×3 Rotationsmatrix R.

Die Translation ist der Vektor vom Ursprung des Weltkoordinatensystems zum Kamerazentrum $C = (C_1, C_2, C_3)$.

$$T = \begin{pmatrix} C_1 \\ C_2 \\ C_3 \end{pmatrix} \tag{2.13}$$

Die 3×3-Rotationsmatrix R kennzeichnet die Orientierung des Kamerakoordinatensystems. Man beachte, dass die Einheitsvektoren $\vec{r}_1, \vec{r}_2, \vec{r}_3$ des Kamerakoordinatensystems dargestellt im Weltkoordinatensystem gerade die Spalten der Rotationsmatrix R sind.

$$R = R_\phi * R_\vartheta * R_\psi \tag{2.14}$$

$$= (\vec{r}_1, \vec{r}_2, \vec{r}_3) \tag{2.15}$$

$$= \begin{pmatrix} r_{1,1} & r_{1,2} & r_{1,3} \\ r_{2,1} & r_{2,2} & r_{2,3} \\ r_{3,1} & r_{3,2} & r_{3,3} \end{pmatrix} \tag{2.16}$$

Die Rotationsmatrix R hat neun Elemente, jedoch nur drei Freiheitsgrade, da sie z.B. über die drei Raumwinkel parametrisierbar ist.

Durch Hintereinanderausführung der Rotation und Translation erhält man eine bijektive Abbildung, mit der zwischen Kamera- und Bildkoordinaten transformiert werden kann. Die Bildkoordinaten $q = (u, v)$ eines Punktes $Q = (x, y, z)$ der Szene erhält man z.B. als

$$q = R^T(Q - C) \tag{2.17}$$

Die Projektion in die Bildebene ergibt mit (2.12) die Koordinaten

$$u = \frac{r_{1,1}(x - C_1) + r_{2,1}(y - C_2) + r_{3,1}(z - C_3)}{r_{1,3}(x - C_1) + r_{2,3}(y - C_2) + r_{3,3}(z - C_3)} \tag{2.18}$$

und

$$v = \frac{r_{1,2}(x - C_1) + r_{2,2}(y - C_2) + r_{3,2}(z - C_3)}{r_{1,3}(x - C_1) + r_{2,3}(y - C_2) + r_{3,3}(z - C_3)} \tag{2.19}$$

Sind die äußeren Kameraparameter Translation und Rotation der Kamera bekannt, so spricht man davon, dass die Kamera *extern kalibriert* ist.

2.3.3.2 Innere Kameraparameter - Die K-Matrix

Der Sensor liefert normalerweise Daten in seinem eigenen diskreten *Pixelkoordinatensystem*, dessen Transformation in Kamerakoordinaten nur von den inneren Parametern der Kamera abhängig ist. Diese sind:

- Das **Kamerazentrum**
  mit den Pixelkoordinaten $(center_x, center_y)$, das im Allgemeinen in der Bildebene in der Mitte des Sensors oder dessen Nähe liegt.

- Die **Brennweiten** in x- und y-Richtung f_x, f_y
  geben für x- und y-Richtung getrennt die Skalierung an. Sie sind also ein Maß für die Größe eines Pixels in Weltkoordinaten.

- Die **Pixelscherung** s
  gibt den Winkel der Pixel zueinander an. Es ist $s = 0$ genau dann, wenn die Pixel orthogonal zueinander stehen, was üblicherweise der Fall ist.

Die Transformation von dem geometrischen Kamerakoordinaten (u, v) in Pixelkoordinaten (x, y) wird durch folgende affine Abbildung modelliert.

$$x = f_x u + sv + center_x \tag{2.20}$$

$$y = f_y v + center_y \tag{2.21}$$

Eleganter lässt sich diese affine Abbildung als Matrix schreiben, die nur von den inneren Parametern abhängig ist und *Kamera-Kalibriermatrix* oder K-Matrix genannt wird.

Mit der 3×3 K-Matrix

$$K = \begin{pmatrix} f_x w & s & center_x \\ 0 & f_y h & center_y \\ 0 & 0 & 1 \end{pmatrix} \tag{2.22}$$

ergeben sich die Kamerakoordinaten (x, y) aus den Pixelkoordinaten $(u, v, 1)$ in homogenen Koordinaten als

$$p = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{2.23}$$

$$= \begin{pmatrix} f_x w & s & center_x \\ 0 & f_y h & center_y \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \tag{2.24}$$

Das Verhältinis $\alpha = f_x/f_y$ wird *Seitenverhältnis* (engl. aspect ratio) gegannt. Da meistens quadratische Pixel vorliegen, ist $f_x \approx f_y$, und es folgt, dass das Seitenverhältnis $\alpha \approx 1$ ist. Dies ist aber von der Form der Pixel abhängig.

Bei den verwendeten Kameras kann die Pixelscherung s als 0 angenommen werden, da sie kein geschertes Bild liefern.

Die Betrachtung der inneren Kameraparameter ist idealisiert, weil z.B. keine optischen Verzerrungen betrachtet werden und auch von einem regulären orthogonalen Abtastgitter ausgegangen wird.

Weil die Abweichungen vom obigen Modell klein sind, können sie vernachlässigt werden, so dass die Benutzung einer affinen Abbildung zwischen Pixelkoordinaten und Bildkoordinaten gerechtfertigt ist.

Sind die inneren Kameraparameter, also die K-Matrix, bekannt, so nennt man die Kamera *intern kalibriert*. Man beachte, dass die K-Matrix also insgesamt fünf freie Parameter besitzt.

2.4 Die P-Matrix

Die Formeln (2.18) und (2.19) können mit

$$p = r_{1,3}(x - C_1) + r_{2,3}(y - C_2) + r_{3,3}(z - C_3) \tag{2.25}$$

einfacher ausgedrückt werden als

$$p \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = R \begin{pmatrix} x - C_1 \\ y - C_2 \\ z - C_3 \end{pmatrix} \tag{2.26}$$

Kombiniert man diese mit (2.23), so ergibt sich für ein $\lambda \neq 0$ und $Q = (x, y, z)$ die *Projektionsgleichung*

$$\lambda q = KR^T(Q - C) \tag{2.27}$$

Stellt man auch noch den Punkt der Szene Q in homogenen Koordinaten als $\tilde{Q} = (x, y, z, 1)^T$ dar, so kann man die Gleichung (2.27) mit $\tilde{q} = (x, y, w)^T$ auch darstellen als

$$\lambda \tilde{q} = \underbrace{(KR^T | - KR^T C)}_{P} \tilde{Q} \tag{2.28}$$

Die 3×4-Matrix P nennt man die *Projektionsmatrix* der Kamera.

$$P = (KR^T | - KR^T C) = KM \text{ mit } M = (R^T | - R^T C) \tag{2.29}$$

Dabei stellt $\lambda \in \mathbb{R}$ nur einen Proportionalitätsfaktor dar. Der Vorteil der Darstellung in projektiven Koordinaten ist an dieser Stelle deutlich erkennbar, denn die Abbildung eines Szenenpunktes von seinen homogenen Weltkoordinaten in seine projektiven Pixelkoordinaten lässt sich mit *einer linearen Matrixoperation* nur in projektiver Geometrie beschreiben.

Unter Benutzung der Projektionsgleichung (2.29) reicht allein die Kenntnis einer 3×4-Matrix P aus, um die Projektion von Weltkoordinaten in Pixelkoordinaten zu beschreiben.

Die P-Matrix hat 11 freie Parameter, da zu den 5 Freiheitsgraden von K noch jeweils drei für Rotation und Translation dazu kommen.

2.5 Epipolargeometrie

Bisher wurden nur einzelne Bilder betrachtet, genauer die Entstehung eines Bildes als Projektion einer Szene in die Bildebene einer Kamera. Bei Aufnahme von

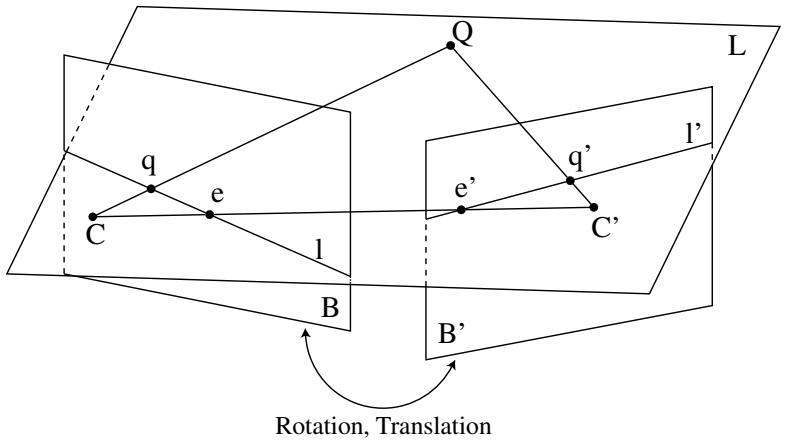


Abbildung 2.6: Korrespondenzen zweier Ansichten in epipolarer Geometrie.

mehreren Bildern bestehen jedoch Beziehungen, die durch die Epipolargeometrie beschrieben werden können. Dabei wird eine Szene aus unterschiedlicher Perspektive aufgenommen, um die geometrischen Beziehungen zwischen den einzelnen Kameras betrachten zu können. Diese werden dazu benutzt, um

- die *Korrespondenzen* zwischen den Bildpunkten aus beiden Bildern zu finden, die zum selben Szenenpunkt gehören,
- die *Kamerabewegung* im Raum zu schätzen,
- eine *3D-Struktur der Szene* zu erhalten.

Eine *epipolare Geometrie* besteht zwischen zwei beliebigen Kamarasystemen und ist in Abbildung 2.6 skizziert.

Dabei bezeichnen C und C' die Projektionszentren der ersten und zweiten Kamera. Der Punkt Q der Szene wird mit der ersten Kamera auf den Bildpunkt q und mit der zweiten Kamera auf den Bildpunkt q' abgebildet. Da q bzw. q' Projektionen desselben Raumpunktes Q sind, werden sie *korrespondierende* oder *homologe Punkte* genannt. Die *Epipolarebene* L sei die Ebene, die durch die Punkte C, C' und Q gebildet wird. Jeder auf dem Strahl $\overline{CQ}$ liegende Punkt wird in q abgebildet.

Eine Gerade $\overline{CC'}$ durch die Kamerazentren schneidet die Bildebenen B und B' in den Punkten e und e'. Diese Punkte werden *Epipol* genannt. Die Projektion des Kamerazentrums eines Bildes in das jeweils andere Bild ist somit gerade der Epipol.

Die Linien l bzw. l' sind der Schnitt der Epipolarebene L mit den Bildebenen B bzw. B' und werden *Epipolarlinien* genannt. Die Raumpunkte auf dem Sichtstrahl $\overline{CQ}$ bilden sich alle auf die Epipolarlinie l' als Projektion des Strahls in die andere Kamera ab. Durch das Wissen von C' und B' allein ist bekannt, dass der zu q korrespondierende Punkt auf der Epiploarlinie l' liegen muss.

Man beachte, dass obwohl die genaue Lage von q' nicht bekannt ist, allein durch die Kenntnis von C, C' sowie B und B' der mögliche Ort auf die Epipolarlinie l' im zweiten Bild eingeschränkt ist.

Im Folgenden wird diese Eigenschaft mathematisch beschrieben. Dazu wird zunächst eine Schreibweise eingeführt.

Definition 3 (Vektorprodukt-Matrix) *Für einen Vektor* $a = (a_1, a_2, a_3)^T \in \mathbb{R}^3$ *sei* $[a]_\times \in \mathbb{R}^{3\times 3}$ *die* Vektorprodukt-Matrix

$$[a]_\times = \begin{pmatrix} 0 & -a_3 & a_2 \\ a_3 & 0 & -a_1 \\ -a_2 & a_1 & 0 \end{pmatrix}. \tag{2.30}$$

Man beachte, dass dies gerade die Schreibweise des Kreuzproduktes als Matrixmultiplikation ist, es gilt also

$$\forall v \in \mathbb{R}^3 : a \times v = [a]_\times \cdot v. \tag{2.31}$$

Wegen der schiefen Symmetrie der Matrix gilt

$$[a]_\times = -[a]_\times^T. \tag{2.32}$$

Für folgende Betrachtung seien K, R und C die den Kameraparametern entsprechenden Matrizen.

Der Bildpunkt q liegt auf der Strecke $\overline{CQ}$. Der Richtungsvektor $\vec{Cq}$ wird in Weltkoordinaten beschrieben durch

$$\vec{Cq} = RK^{-1}q. \tag{2.33}$$

Dann gibt es ein $\lambda \in \mathbb{R}$ so, dass mit Kamerazentrum C

$$Q = C + \lambda R K^{-1} q. \tag{2.34}$$

Dies stellt nur eine andere Schreibweise für die Projektionsgleichung (2.28) dar. Dann gibt es ein $\mu \in \mathbb{R}$ so, dass mit den Kameraparametern der zweiten Kamera K', R' und C'

$$\mu q' = K' R'^T (Q - C'). \tag{2.35}$$

Der Punkt Q wird auch im zweiten Bild abgebildet. Man kann also Gleichung (2.34) in (2.35) einsetzen und erhält

$$\begin{aligned}
\vartheta q' &= K' R'^T (Q - C') && (2.36)\\
&\overset{(2.34)}{=} K' R'^T (C + \lambda R K^{-1} q - C') && (2.37)\\
&= K' R'^T (\lambda R K^{-1} q + C - C') && (2.38)\\
&= \underbrace{\lambda K' R'^T R K^{-1} q}_{\text{erster Term}} + \underbrace{K' R'^T (C - C')}_{\text{zweiter Term}}. && (2.39)
\end{aligned}$$

Der zweite Term entspricht der Projektion des Kamerazentrums C in die Bildebene der zweiten Kamera, ist also gerade e'. Dazu setzt man den Epipol e', welcher der Projektion von C in die Bildebene von C' entspricht, in Gleichung (2.35) ein, also

$$\mu e' = K' R'^T (C - C'). \tag{2.40}$$

Die Gleichung (2.39) besagt also, dass q' auf der Linie e' und $K' R'^T R K^{-1}$ liegt.

2.5.1 Die Homographie-Matrix H

Im Folgenden sei

$$H = K' R'^T R K^{-1} \tag{2.41}$$

die *Homographie-Matrix* oder auch *Kollineationsmatrix*. In projektiver Geometrie ausgedrückt, ist H eine Matrix, die die erste Bildebene über die Ebene in der Unendlichkeit in die zweite Bildebene abbildet. Damit kann Gleichung (2.39) vereinfacht werden zu

$$\vartheta q' = \lambda H q + \mu e'. \tag{2.42}$$

Im Folgenden werden zwei Bedingungen vorgestellt, die in der Epipolargeometrie gelten und die für die spätere Definition der *Fundamentalmatrix* von Bedeutung sind.

Definition 4 (Koplanaritätsbedingung) *Seien q und q' korrespondierende Punkte. Dann müssen q, q', C, C' und Q auf einer Ebene liegen.*

Definition 5 (Epipolarbedingung) *Zu einem Punkt q muss der korrespondierende Punkt q' auf der Epipolarlinie liegen.*

2.5.2 Die Fundamentalmatrix F

Die Fundamentalmatrix F stellt eine Zusammenführung der Koplanaritäts- und Epipolarbedingung dar. Dazu seien B und B' zwei Ansichten einer statischen Szene und $q \in B, q' \in B'$ korrespondierende Bildpunkte. Dann gibt es eine Matrix $F \in \mathbb{R}^{3\times 3}$ mit Rang 2 so, dass

$$q'^T F q = 0. \tag{2.43}$$

Setzt man $q' = (u', v', 1)^T$ und $q = (u, v, 1)^T$ in Gleichung (2.43) ein, so gilt

$$(u' \quad v' \quad 1) F \begin{pmatrix} u \\ v \\ 1 \end{pmatrix} = 0. \tag{2.44}$$

Diese Bedingung wird *Fundamentalrelation* genannt. Sie berücksichtigt sowohl Koplanaritäts- als auch Epipolarbedingung.

Die Fundamentalmatrix F kann berechnet werden durch

$$F = [e']_\times H \tag{2.45}$$

und hat genau 7 Freiheitsgrade.

Kapitel 3

Rekonstruktion einer Szenengeometrie

Mit Hilfe der Epipolargeometrie ist es möglich, die Szenengeometrie in gewissem Rahmen zu rekonstruieren. Dazu wird zunächst eine *Korrespondenzsuche* durchgeführt, um die F-Matrix bestimmen zu können, und die Kamerapositionen geschätzt. Mittels der Korrespondenzsuche, insbesondere durch das *Matching*, kann eine flächenbasierte *Disparitätssuche* genutzt werden, um die Tiefenkarten zu erstellen.

Dazu wird in den folgenden Abschnitten zunächst die Bestimmung der F-Matrix beschrieben, bevor die eigentliche Korrespondenzsuche beginnt, da für eine verlässliche Suche abschnittsweise schon eine Fundamental-Matrix benötigt wird.

3.1 Bestimmung der F-Matrix

Wie in Kapitel 2.5.2 beschrieben, hat die Fundamentalmatrix F genau 7 Freiheitsgrade. Ermittelt man also sieben verschiedene Korrespondenzen zwischen den Bildern, kann F bereits geschätzt werden. Dazu wird im folgenden Abschnitt ein nichtlinearer Algorithmus vorgestellt, der genau 7 Korrespondenzen benötigt.

Darauf aufbauend wird ein robuster Algorithmus zur verlässlichen Bestim-

mung der F-Matrix erläutert. Dieser stellt eine Kombination aus Näherungsverfahren und exaktem Verfahrem dar und benötigt mehr als 7 Korrespondenzen, ist aber rauschunempfindlich und auch bei fehlerhaften Korrespondenzen noch zuverlässig verglichen mit dem Sieben-Punkt- oder Acht-Punkt-Algorithmus.

3.1.1 Sieben-Punkt-Algorithmus

Seien $q_i = (x, y, 1)^T$ und $q_i' = (x', y', 1)^T$ korrespondierende Punkte mit $i \in \{0, \ldots, 6\}$. Dann gilt mit Gleichung (2.43)

$$0 = q_i^T F q_i' \tag{3.1}$$

$$= (x \quad y \quad 1) \begin{pmatrix} F_{1,1} & F_{1,2} & F_{1,3} \\ F_{2,1} & F_{2,2} & F_{2,3} \\ F_{3,1} & F_{3,2} & F_{3,3} \end{pmatrix} \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} \tag{3.2}$$

$$= \underbrace{(xx'yx'x'xy'yy'y'xy1)}_{u_i^T} \underbrace{(F_{1,1}F_{1,2}F_{1,3}F_{2,1}F_{2,2}F_{2,3}F_{3,1}F_{3,2}F_{3,3})^T}_{f} \tag{3.3}$$

$$= u_i^T f \text{ mit } i \in \{0, \ldots, 6\} \tag{3.4}$$

mit

$$u_i = (xx' \quad yx' \quad x' \quad xy' \quad yy' \quad y' \quad x \quad y \quad 1)^T \tag{3.5}$$

$$f = (F_{1,1} \quad F_{1,2} \quad F_{1,3} \quad F_{2,1} \quad F_{2,2} \quad F_{2,3} \quad F_{3,1} \quad F_{3,2} \quad F_{3,3})^T. \tag{3.6}$$

Aus n gegebenen Korrespondenzen erhält man ein Gleichungssystem A_n aus n Gleichungen

$$A_n = (u_1, \ldots, u_n)^T. \tag{3.7}$$

Dann gilt

$$A_n f = 0. \tag{3.8}$$

Da F genau 7 Freiheitsgrade hat, ist das Gleichungssystem bei 7 Korrespondenzen lösbar.

Eine Eigenwertzerlegung von A_n berechnet die Zerlegung

$$A_n = USV^T \tag{3.9}$$

mit U und V Orthonormalmatrizen und S Diagonalmatrix.

Seien Vektoren v_1, v_2 die letzten beiden Spalten von V. Der so aufgespannte Nullraum kann parametrisiert werden als

$$\lambda v_1 + (1 - \lambda)v_2. \tag{3.10}$$

Seien F_1, F_2 die zu $v1$ und v_2 korrespondierendnen Matrizen. Dann gilt

$$F = \lambda F_1 + (1 - \lambda)F_2. \tag{3.11}$$

Mit der Einschränkung der F-Matrix auf Rang 2 gilt

$$\begin{aligned} 0 &= det(F) && (3.12)\\ &= det(\lambda F_1 + (1 - \lambda)F_2) && (3.13)\\ &= a_3\lambda^3 + a_2\lambda^2 + a_0. && (3.14) \end{aligned}$$

Als Lösung dieses kubischen Polynoms erhält man entweder eine oder drei reelle Lösungen, von denen nur eine zu den untersuchten Korrespondenzen geführt haben kann.

3.1.2 RANSAC-Algorithmus

Fischler und Bolles haben bereits 1981 einen robusten Algorithmus vorgestellt, der auch auf die Bestimmung der F-Matrix angewandt werden kann. Der sogenannte *Random Sampling Consensus* Algorithmus arbeitet wie folgt.

Sei eine Menge $\mathbb{M}$ von $n > 7$ Korrespondenzen gegeben. Aus diesen wähle man *zufällig* eine Menge von 7 Korrespondenzen aus, die ausreichend sind, eine exakte Lösung F_i berechnen zu können.

Man teste F_i gegenüber allen n Korrespondenzen auf *korrekte* Korrespondenzen und sogenannte *Ausreißer* und zähle diese und deren Abstandsmaß. Die Ausreißer werden von F_i nicht als Korrespondenzen bestätigt, haben also einen zu großen Abstand z.B. von der Epipolarlinie.

Man wiederhole das Verfahren für weitere zufällig ausgewählte Korrespondenzen und ermittle das F_i mit den wenigsten Ausreißern bzw. bei Gleichheit das mit dem besseren Abstandsmaß. Dieses F_i wird $F_{unbereinigt}$ genannt.

Sei $\mathbb{M}_{bereinigt} \subset \mathbb{M}$ die Menge von Korrespondenzen, aus denen alle Ausreißer bezüglich $F_{unbereinigt}$ entfernt wurden.

Man wende den Algorithmus erneut auf $\mathbb{M}_{bereinigt}$ an, um ein verlässliches $F_{bereinigt}$ berechnen zu können.

3.2 Bestimmung von Korrespondenzen

Im vorherigen Abschnitt wurde ein Verfahren erläutert, um eine verlässliche F-Matrix bestimmen zu können. Der folgende Abschnitt erläutert die Bestimmung der einzelnen Korrespondenzen zwischen den Bildern. Dazu werden zunächst Merkmale, also charakteristische Punkte der Szene, bestimmt.

3.2.1 Merkmalsdetektion

Die Merkmale einer Szene werden mit dem *Harris-Eckendetektor* ermittelt. Dieser definiert ein Merkmal bzw. eine Ecke dadurch, dass eine kleine Bewegung in beliebiger Richtung bezüglich des Merkmals eine große Änderung in der Intensität der Bildpunkte bedeutet.

Auf die genauen Einzelheiten der Merkmalsbestimmung sei auf [Woe01] verwiesen.

3.2.2 Normalisierte Kreuzkorrelation

Der vorherige Abschnitt beschrieb die Bestimmung einer genügend großen Menge von Merkmalen der Szene. Im Folgenden soll das Finden von Paaren von korrespondierenden Punkten erläutert werden. Um überhaupt Beziehungen zwischen den Merkmalen finden zu können, folgt eine Definition eines Ähnlichkeitsmaßes mittels der *normalisierten Kreuzkorrelation*.

Diese Metrik basiert auf der Ähnlichkeit der *Nachbarschaften* zweier Punkte. Sei hierzu eine Nachbarschaft N definiert als ein quadratisches Fenster aus $(2n+1) \times (2n+1)$ Bildpunkten mit zentralem Aufpunkt und $I(x, y)$ die Intensität eines Bildpunktes an Position (x, y).

Seien $\bar{I}, \bar{I}'$ die mittleren Intensitäten über die Nachbarschaften, also

$$\bar{I} = \frac{1}{(2n+1)^2} \sum_{i=-n}^{n} \sum_{j=-n}^{n} (I(x+i, y+j)) \tag{3.15}$$

$$\bar{I}' = \frac{1}{(2n+1)^2} \sum_{i=-n}^{n} \sum_{j=-n}^{n} (I'(x+i, y+j)) \tag{3.16}$$

Seien $q = (x, y)$ und $q' = (x', y')$ Bildpunkte. Dann ist die normalisierte Kreuzkorrelation NCC definiert als

$$NCC(q, q') = \sum_{i=-n}^{n} \sum_{j=-n}^{n} ((I(x-i, y-j) - \bar{I})(I'(x'-i, y'-j) - \bar{I}')). \tag{3.17}$$

Je größer die NCC zweier Punkte, desto *ähnlicher* sind sie sich. Typischerweise wird für die Größe der Nachbarschaft ein Wert von $n = 3$ gewählt, es entsteht also ein 7×7 Bildpunkte großes Fenster.

3.2.3 Das Matching

Aufbauend auf einer Menge von Merkmalen einer Szene kann nun versucht werden, diese Merkmale mit Hilfe des im vorigen Abschnittes vorgestellten Ähnlichkeitsmaßes zu Paaren von korrespondierenden Punkten zusammenzuführen, also die Punkte zu *matchen*.

Es Wird davon ausgegangen, dass bei der gesamten Bildfolge von Bild zu Bild nur relativ kleine Kamerabewegungen vorliegen. Der korrespondierende Punkt eines Merkmalpunktes q lässt sich also auf einen bestimmten Bereich um q im zweiten Bild einschränken, der sogenannten *maximalen Disparität*.

Definition 6 (Disparität) *Seien C, C' Kameras und Q ein Punkt, der in zwei Bildern in q bzw. q' abgebildet wird. Dann nennt man $d = \|q - q'\|$ Disparität.*

Ermittelt man zu einem Merkmalspunkt die Ähnlichkeiten zu anderen potentiellen Korrespondenzen innerhalb des eingeschränkten Bereiches, lässt sich relativ schnell eine Grundmenge an Korrespondenzen finden.

Auf diese wird der RANSAC-Algorithmus aus Kapitel 3.1.2 angewandt, um eine vorläufige F-Matrix zu finden. Nun können mutmaßliche Korrespondenzen für Punkte des ersten Bildes entlang der zugehörigen Epipolarlinie im zweiten Bild gefunden werden. Mit dieser erweiterten Menge an Korrespondenzen kann die F-Matrix weiter verfeinert werden.

3.3 Erstellung von Tiefenkarten

Durch einfache Triangulation können bereits Tiefenwerte für alle gefundenen Korrespondenzen bestimmt werden. Die erzeugte Tiefenkarte ist jedoch schwach besetzt mit größeren Lücken. Um eine dichtere Tiefenkarte erstellen zu können, muss der bisherige Ansatz erweitert werden.

Dazu ermittle man für *jeden* Bildpunkt den maximalen NCC-Wert entlang der Epipolarlinie des zweiten Bildes in einem 7×7 Fenster. Nun kann für jeden Pixel die Disparität berechnet und dadurch die Tiefe abgeleitet werden. Dabei werden nicht immer korrekte Korrespondenzen gefunden, die Tiefenkarte stellt nur eine Näherung dar.

Treten Verdeckungen von einem Bild zum anderen auf, kann kein korrekter Tiefenwert gefunden werden. Der entsprechende Bereich bleibt leer.

3.4 Bestimmung der P-Matrizen

In der gewonnenen Bildsequenz wird von einer starren Szene ausgegangen. Aufbauend auf dieser Annahme kann ein Bezugssystem mittels *structure from motion*, also der *Geometrie durch Kamerabewegung*, iterativ geschätzt werden.

Dazu wird zunächst anhand der ersten beiden Bilder ein initiales Bezugssystem geschätzt, welches iterativ verbessert wird.

3.4.1 Bestimmung eines initialen Bezugssystems

Im Folgenden wird o.B.d.A. ein kamerazentriertes Weltkoordinatensystem angenommen.

Sei $F_{1,2}$ eine bereits berechnete F-Matrix zwischen den ersten beiden Bildern. Dann gilt für die initiale P-Matrix

$$P_1 = K \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{3.18}$$

$$= K(I_{3\times 3}|0) \tag{3.19}$$

Mit $F_{1,2}$ kann die Projektionsmatrix der zweiten Kamera geschrieben werden als

$$P_2 = K([e_2]_\times F_{1,2} + e_2 a^T|\sigma e) \tag{3.20}$$

mit σ globaler Skalierungsfaktor, a^T Referenzebene.

P_2 wird so gewählt, dass die auftretenden Verzerrungen möglichst gering sind. Die P-Matrix wird damit 'fast' euklidisch. Nimmt man ferner eine üblicherweise kleine Rotation an, muss nur noch der Vektor a^T der Referenzebene bestimmt werden, denn es gilt nach (2.29)

$$P_2 = K[R^T| - R^T T] \tag{3.21}$$

mit T Translation zum zweiten Kamerazentrum.

Die P-Matrizen können weiter verfeinert werden, indem eine *Selbstkalibrierung* der Kameras durchgeführt wird. Verwiesen sei hier auf [BZM97] und [Pol99].

Kapitel 4

Darstellung neuer Ansichten

Die Darstellung neuer Ansichten in Kapitel 2.1.3 setzte ein regelmäßiges Gitter von Kamerapositionen voraus, und es wurde eine fehlerhafte Interpolation wegen fehlender Tiefeninformationen durchgeführt. Aufgrund von Kapitel 3 können nun die Kamerapositionen und Kameraparameter geschätzt werden, so dass die Annahme eines Gitters gelockert werden kann. Zusätzlich wurden Tiefenkarten eingeführt und für jedes aufgenommene Bild berechnet.

Aus den gewonnenen Tiefenkarten und den geschätzten Kameramatrizen können neue Ansichten aus den vorhandenen Bildern erstellt werden. Dazu wird zunächst eine Regressionsebene über ein Raster von Tiefenwerten jeder Kamera erzeugt, um eine gemittelte Szenenebene für jede Kamera zu erhalten. Diese Ebenen werden dann benötigt, wenn für bestimmte Punkte keine Tiefenwerte vorliegen.

Für die spätere Darstellung mittels OpenGL muss ein dreidimensionales Dreiecksnetz vorliegen. Dazu wird eine 2D-Triangulierung vorgenommen, die jede reale Kameraposition in die Bildebene der virtuellen Kamera projiziert. Aus den gewonnenen 2D-Punkten wird ein optimiertes Dreiecksnetz erzeugt. Dieses Dreiecksnetz wird mittels OpenGL texturiert und die neue Ansicht dargestellt.

4.1 Erzeugung von Regressionsebenen

Um bei der Darstellung auch Punkte berücksichtigen zu können, von denen keine Tiefeninformationen vorliegen, benötigt man für jede reale Kamera eine Ausgleichsebene.

Dazu wird die Bildebene der Kameras mit Breite w und Höhe h in einem Raster abgetastet. Diese 2D-Punkte werden in die virtuelle Szene projiziert. Dann ist

$$\{0..w\text{–}1\} \times \{0..h\text{–}1\} \longrightarrow \mathbb{R}^3 : \quad p_B = (p_{B_1}, p_{B_2}) \mapsto \frac{RK^{-1}\begin{pmatrix} p_{B_1} \\ p_{B_2} \\ 1 \end{pmatrix}}{\left\| RK^{-1}\begin{pmatrix} p_{B_1} \\ p_{B_2} \\ 1 \end{pmatrix} \right\|} d + C \tag{4.1}$$

$$=: p_R$$

eine Abbildung von Bildkoordinaten p_B in Raumkoordinaten p_R.

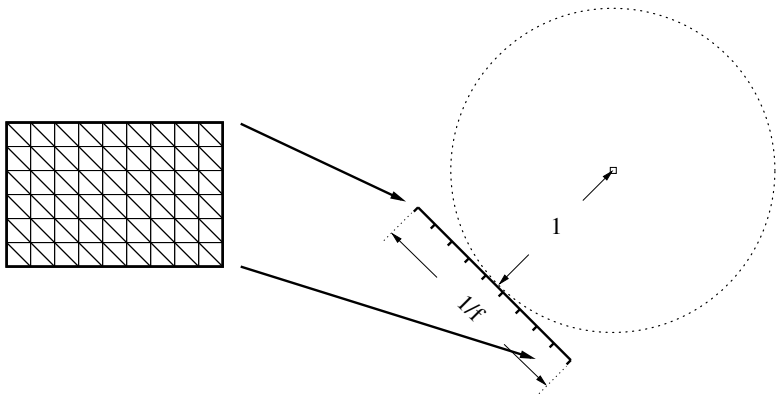


Abbildung 4.1: Aufbau eines Rasters und Transformation in eine virtuelle Szene

Aus den gewonnenen 3D-Punkten kann nun eine Ausgleichsebene erzeugt werden. Es wird versucht, eine Ebene zu finden, deren Abstand zu den einzelnen Punkten minimal ist. Dazu wird die Hesse'sche Normalenform der Ebene benutzt:

$$\vec{n}\vec{p} - d = 0 \tag{4.2}$$

Die zu lösende Gleichung ist also

$$A\vec{n_L} \underset{min}{=} \vec{0} \quad \text{mit } A = \begin{pmatrix} -1 & x_1 & y_1 & z_1 \\ \cdots & \cdots & \cdots & \cdots \\ \cdots & \cdots & \cdots & \cdots \\ -1 & x_i & y_i & z_i \end{pmatrix} \text{ und } \vec{n_L} = \begin{pmatrix} d \\ x_n \\ y_n \\ z_n \end{pmatrix} \text{ Lösungsvektor.} \tag{4.3}$$

Diese Gleichung wird mit Hilfe einer Singulärwert-Zerlegung gelöst.

4.2 Erzeugung eines Dreiecksnetzes

4.2.1 2D-Triangulierung

Voraussetzung, um überhaupt eine Triangulierung vornehmen zu können, ist eine Beschränkung der virtuellen Kamera auf Positionen *hinter* den realen Kamerapositionen in Bezug zur Szene. Damit ist bereits eine grundlegende zweidimensionale Triangulierung möglich.

Hierfür werden die realen Kamerapositionen C_i in die Bildebene der virtuellen Kamera projiziert. Also

$$\forall i \in \{1, \ldots, anz\} : \begin{pmatrix} p_{i_x} w \\ p_{i_y} w \\ w \end{pmatrix} = P_v C_i = (K_v R_v^T | - K_v R_v^T C_v) C_i \tag{4.4}$$

Aus den gewonnenen 2D-Punkten kann nun ein optimiertes Dreiecksnetz erzeugt werden. Dazu wird ein mittels *divide-and-conquer* inkrementell arbeitender Delaunay-Triangulations-Algorithmus von [She96] benutzt. Dieser Algorithmus stellt unter anderem sicher, dass aus den vorhandenen Punkten Dreiecke erstellt werden, deren Winkel nicht zu klein für eine gute Darstellung sind. In Abbildung 4.2 ist ein solches 2D-Dreiecksnetz zu sehen.

4.2.2 3D-Triangulierung

Als Basis für eine weitere Unterteilung wird das ursprüngliche 2D-Dreiecksnetz mit Hilfe der Tiefenkarten bzw. der Ausgleichsebenen in die Szene projiziert. Da-

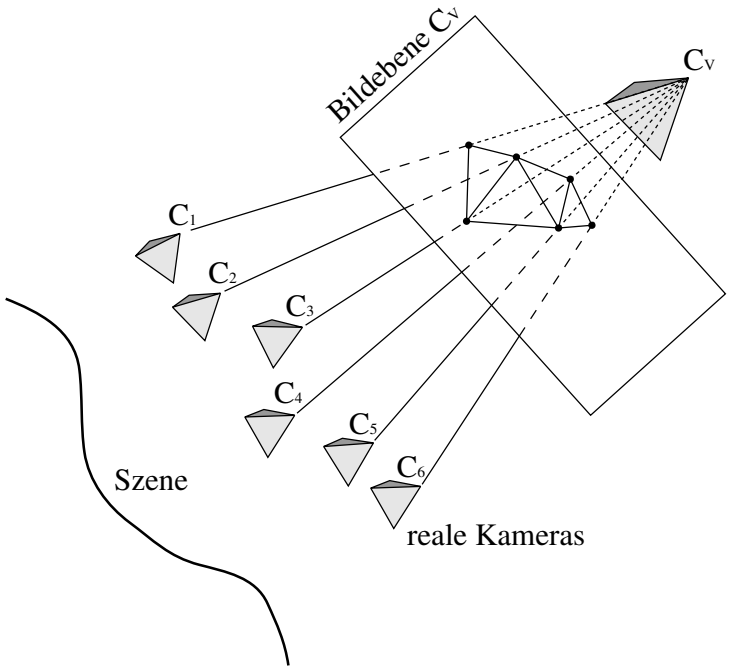


Abbildung 4.2: 2D-Triangulierung.

zu benötigt man einen Punkt in Richtung Szene auf dem Sichtstrahl von der virtuellen Kamera zur aktuellen realen Kamera, also derjenigen, dessen korrespondierender 2D-Punkt in die virtuelle Szene projiziert werden soll (siehe Abbildung 4.3).

Dann kann mit einer einfachen Multiplikation mit der P-Matrix der realen Kamera der Eintrag in der Tiefenkarte gefunden werden. Existiert kein gültiger Tiefenwert, wird der aktuelle Punkt mit Hilfe der Regressionsebene der Kamera in die virtuelle Szene projiziert.

Ausgehend von dieser grundlegenden Einteilung der virtuellen Szene wird eine weitere Unterteilung in Unterdreiecke rekursiv vorgenommen. In Abbildung 4.4 sieht man das Verfahren zur Unterteilung einer Strecke p_1, p_2 eines Dreiecks p_1, p_2, p_3. Nach Berechnung des Mittelpunktes der Strecke wird die Tiefe des Punktes ermittelt. Dazu projiziert man den Punkt in die Bildebene einer der beiden beteiligten Kameras und erhält den zugehörigen Tiefenwert. Die Auswahl der Kameras ist bei feinerer Unterteilung nicht mehr trivial. Man ordnet hier jedem

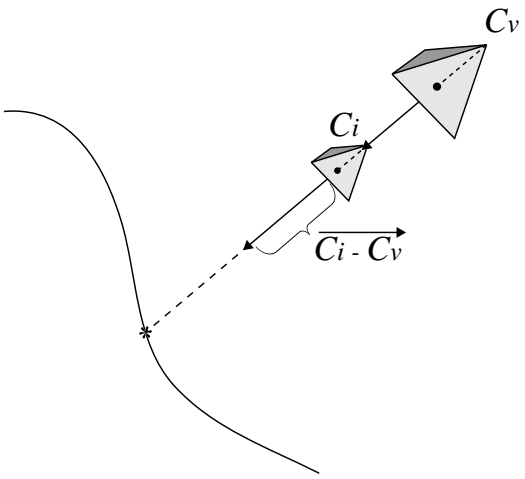


Abbildung 4.3: Projektion der Kamerazentren.

Punkt jedes Dreiecks, in Kapitel 4.3 genauer beschriebener Weise, ein Gewicht und Kameranummer zu, die Kamera mit dem größten Gewicht wird ausgewählt.

Die Qualität der Darstellung ist abhängig von der Rekursionstiefe. Je feiner die Unterteilung ist, desto besser wird die Szenengeometrie approximiert, allerdings zu Lasten der Rechengeschwindigkeit.

4.3 Multitexturing und Gewichtung

Das erzeugte 3D-Dreiecksnetz stellt eine approximierte Szenengeometrie aus einer virtuellen neuen Ansicht dar. Es fehlt allerdings noch die eigentliche farbliche Darstellung der Szene, es existiert nur ein Drahtgittermodell. Um zu einer wirklichkeitsnahen Darstellung zu kommen, muss eine *Texturierung* erfolgen.

Zu allen vorhanden Dreiecken tragen mehrere Kameras zur Darstellung bei. Hätte man drei Aufnahmen einer realen Szene von drei Kameras C_1, C_2, C_3, ergäbe eine erste Triangulierung und Projektion in die virtuelle Szene *ein* Dreieck. Zu diesem Dreieck tragen alle drei Kameras zu gleichen Teilen bei. Nun wird angenommen, dass ein Punkt, der in dem Dreieck dichter an der projizierten Position von Kamera C_1 liegt als an C_2 und C_3, seinen Farbwert am stärksten von C_1 bzw.

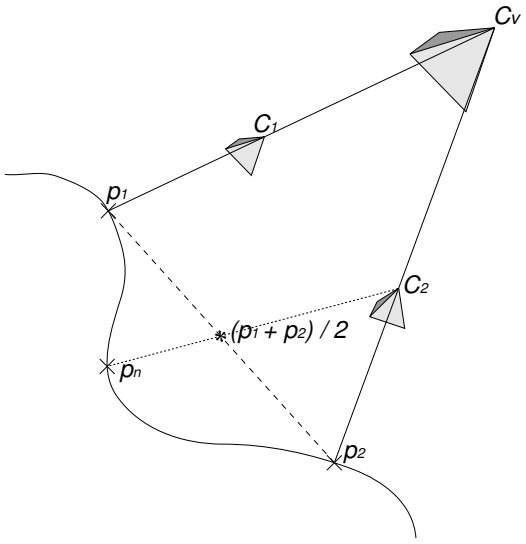


Abbildung 4.4: Unterteilung der Dreiecke.

von deren Aufnahme bezieht. Die Aufnahmen von C_2 und C_3 beeinflussen den Punkt in schwächerer Weise. Um diese Annahme nachzubilden, werden die drei Aufnahmen in geeigneter Art und Weise im Dreieck *gemischt* (engl. *blending*). Dazu weist man jeder Ecke des Dreiecks die entsprechende Aufnahme der jeweiligen Kamera zu und gewichtet diese Ecke mit 1 und die beiden anderen mit 0. Die inneren Punkte werden aus den Eckwerten linear interpoliert. Dabei beträgt die Summe aller Gewichte jeden Punktes innerhalb des Dreiecks 1.

Werden die Dreiecke unterteilt, verändern sich die Gewichte der einzelnen Bilder. Dazu betrachte man das Dreieck p_4, p_5, p_6 in Abbildung 4.5. Der Punkt p_4 ist Bild 1, p_5 Bild 2 und p_6 Bild 3 zugeordnet. Die Gewichte betragen nun nicht mehr 1, denn das Dreieck ist näher an p_1 als an p_2 oder p_3. Da p_4 auf $\frac{1}{4}$ der Strecke von p_1, p_2 liegt, erhält der Punkt ein Gewicht von 0.75, p_5 wird mit 0.5 und p_6 mit 0.25 gewichtet. An jeweils den anderen beiden Punkten werden die Gewichte auf 1 summiert.

Das geschilderte Verfahren ist dann exakt, wenn das Dreieck aus den drei Aufnahmen in einem Schritt texturiert wird, z.B mittels einer Software-Texturierung. Bei Verwendung von OpenGL ergibt sich jedoch die Notwendigkeit, das Dreieck

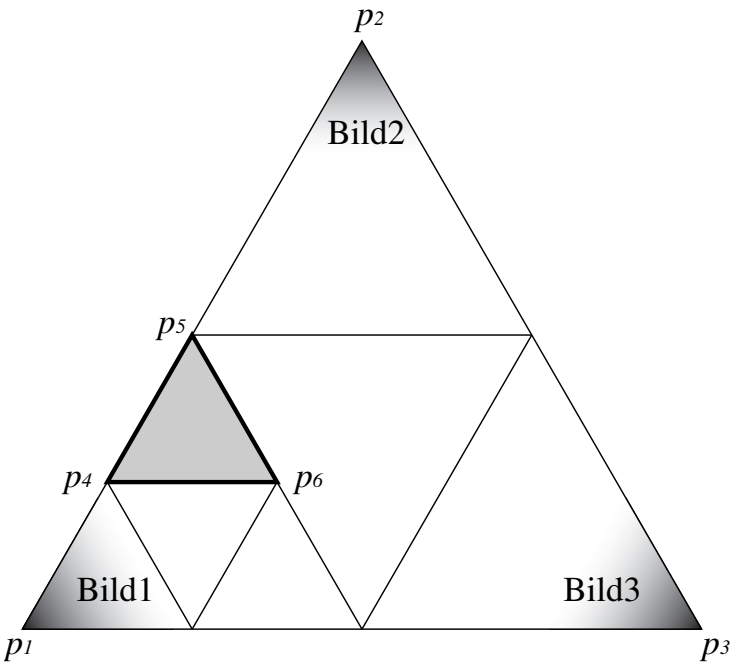


Abbildung 4.5: Unterteilung in Unterdreiecke.

in drei Schritten zu texturieren, jeweils ein Schritt pro Aufnahme. Dazu wird im ersten Schritt ohne Blending gearbeitet, erst im zweiten und dritten wird gemischt.

Das verwendete Blending-Verfahren arbeitet wie folgt:

$$D_{neu} = \alpha_s S + (1 - \alpha_s) D_{alt} \tag{4.5}$$

mit α Gewichtung, D_{alt} bisher berechneter Farbwert, D_{neu} aktualisierter Farbwert und S Quell-Farbwert.

Wie anhand der Gleichung zu sehen ist, können die Farbwertanteile in jedem Blending-Schritt höchstens gleichbleiben, aber nie steigen. Um die Farbwertanteile des ersten gezeichneten Bildes nicht zu klein werden zu lassen, muss die erste Textur ohne Blending gezeichnet werden.

Wird beispielsweise ein gleichseitiges Dreieck in beschriebener Weise aus drei Bildern gemischt, beträgt das Gewicht aller drei Bilder im Schnittpunkt der Winkelhalbierenden genau $\frac{1}{3}$. Wendet man Gleichung (4.5) auf das dreistufige Misch-

Verfahren an, ergeben sich folgende Bildanteile im Schnittpunkt:

$$D = F_1 \qquad \text{Schritt1} \tag{4.6}$$
$$= \frac{1}{3}F_2 + \frac{2}{3}F_1 \qquad \text{Schritt2} \tag{4.7}$$
$$= \frac{1}{3}F_3 + \frac{2}{3}(\frac{1}{3}F_2 + \frac{2}{3}F_1) \qquad \text{Schritt3} \tag{4.8}$$
$$= \frac{1}{3}F_3 + \frac{2}{9}F_2 + \frac{4}{9}F_1 \tag{4.9}$$

mit F_i Farbwert von Bild i an Schnittposition.

Man erkennt eine leichte Unterbewertung von Bild 2 und eine Überbewertung von Bild 1 an der Schnittposition. Das Verfahren ist also nicht exakt, es lassen sich aber mit bloßem Auge kaum Unterschiede zu einem exakten Verfahren feststellen.

Kapitel 5

Ergebnisse

In diesem Kapitel werden einige Ergebnisse und Beispiele präsentiert, die mit Hilfe der in dieser Arbeit entwickelten Software erzeugt wurden.

5.1 Darstellungsergebnisse

Dieser Abschnitt zeigt einige Darstellungsergebnisse einer Szene, die aus acht Kamerapositionen aufgenommen wurde.

Die entstandenen Bildsequenzen basieren auf relativ dichten Tiefenkarten. Eine solche ist in Abbildung 5.1 zu sehen. Sie ist nicht lückenlos, es sind einige schwarze Flecken sichtbar, die dann entstehen, wenn keine verlässlichen Tiefeninformationen für diesen Bereich vorliegen. Diese Bereiche müssen mit Hilfe der Regressionsebenen ausgeglichen werden.

Nach der Triangulierung und Unterteilung in Unterdreiecke kommt es zu Dreiecken, die nur in der Bildebene einer oder zwei Kameras liegen. Ein korrektes Blending benötigt jedoch alle drei Kameras.

Die Abbildungen B.2 und B.3 zeigen eine Darstellung, in der nur Dreiecke gezeichnet werden, die von allen drei Kameras sichtbar sind. In Abbildung B.1 sind auch Dreiecke sichtbar, die von weniger als drei Kameras sichtbar sind. Dabei kommt es zu Unregelmäßigkeiten, die besonders gut im oberen Teil des Bildes sichtbar sind. Die Bildübergänge sind nicht mehr nahtlos, da die beteiligten Ka-

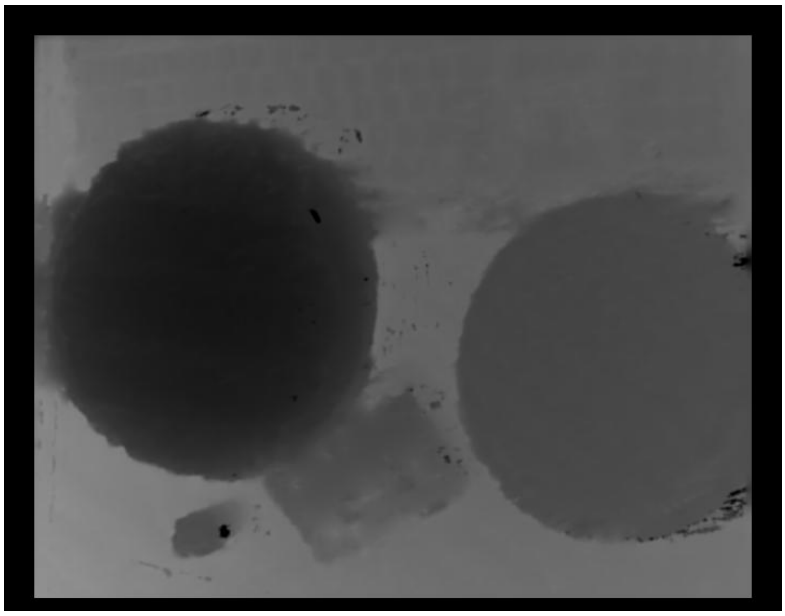


Abbildung 5.1: Tiefenkarte.

meras gewechselt haben.

5.2 Szenengeometrie bei steigender Dreiecksdichte

Die Rekursionstiefe der Dreiecksunterteilung, also die Dreiecksdichte, ist je nach Leistungsmöglichkeiten des verwendeten Rechners einstellbar. Damit werden die Qualität und die Geschwindigkeit der Darstellung skalierbar, bei geeigneter Tiefe wird das System echtzeitfähig. Abbildungen 5.7(a) bis (e) zeigen, wie gut die Szenengeometrie angenähert wird, wenn eine bestimmmte Dreiecksdichte benutzt wird.

5.3 Kameraaufbau

Die gesamte Szene wurde aus achte Kamerapositionen aufgenommen. Abbildung 5.7(f) zeigt die Positionen der realen Kameras und darüber die virtuelle Kamera. Die Szenengeometrie wurde mit Rekursionstiefe 6 erstellt.

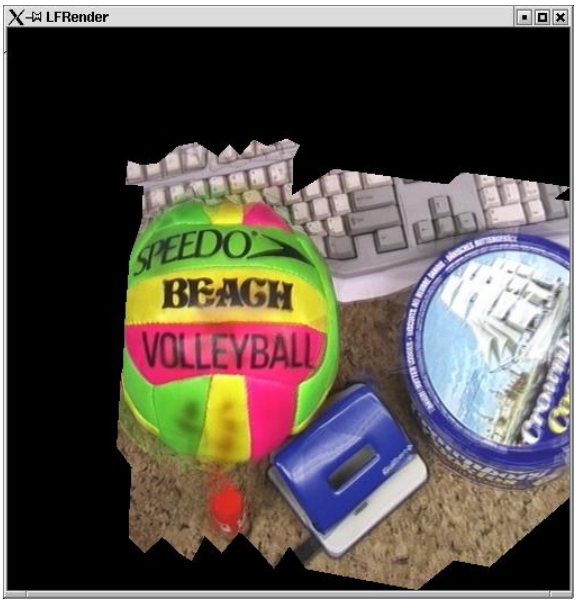


Abbildung 5.2: Erste Ansicht einer Szene, nur Dreiecke gezeichnet, die von mindestens drei Kameras gesehen werden

Abbildung 5.3: Zweite Ansicht einer Szene, nur Dreiecke gezeichnet, die von mindestens drei Kameras gesehen werden

Abbildung 5.4: Dritte Ansicht einer Szene, auch Dreiecke gezeichnet, die nur von einer oder zwei Kameras gesehen werden

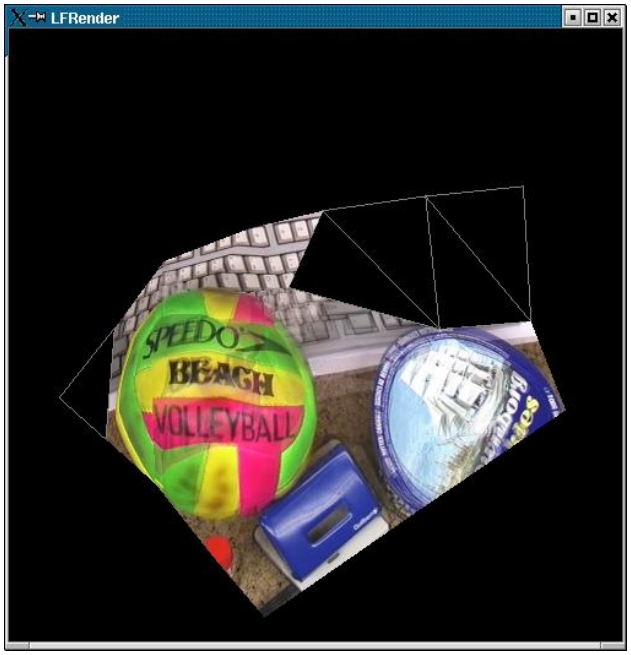
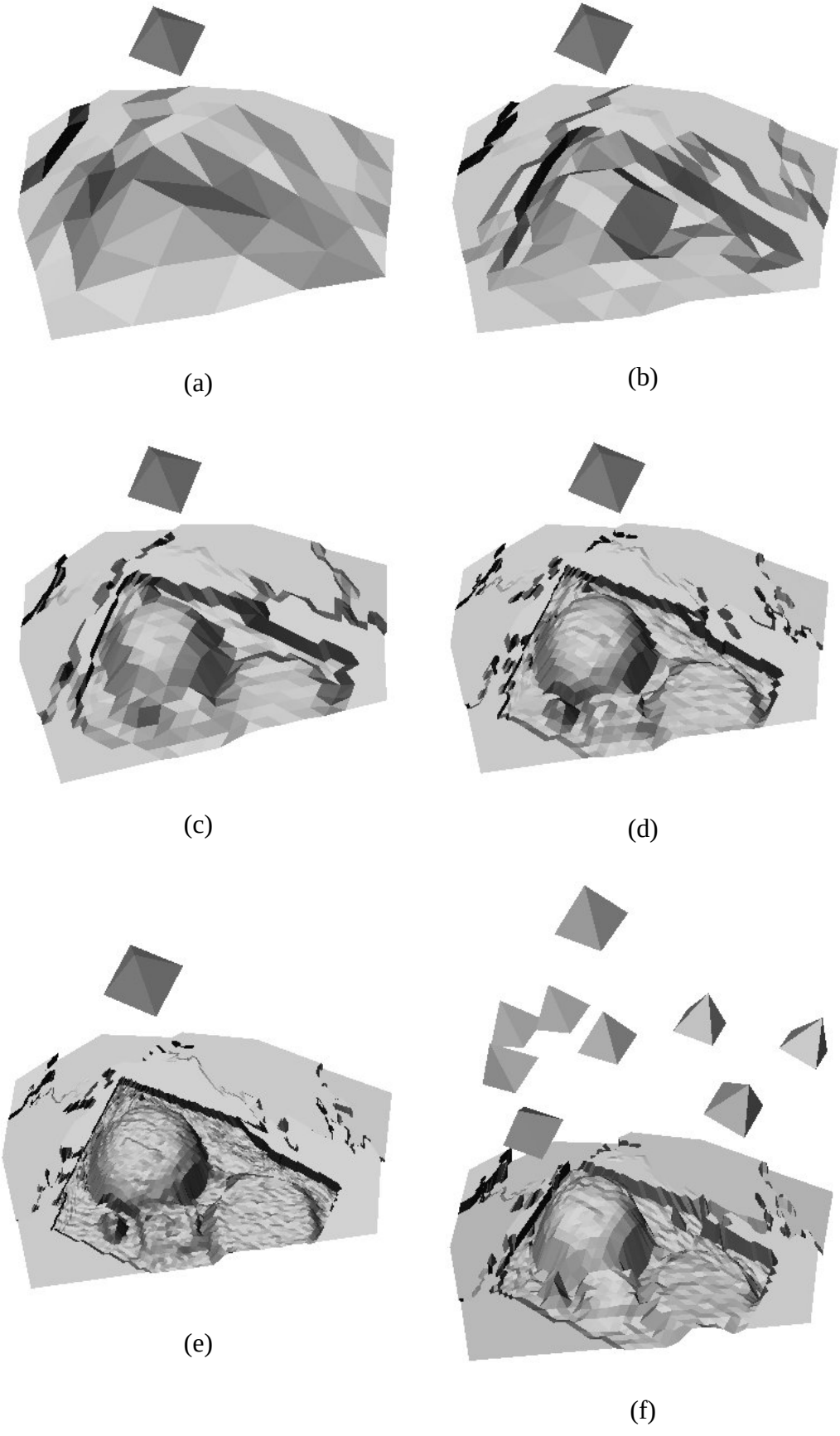


Abbildung 5.5: Ansicht einer Szene mit einer groben Gitterstruktur

Abbildung 5.6: Ansicht einer Szene mit einer feinen Gitterstruktur



(a) (b) (c) (d) (e) (f)

Abbildung 5.7: Szenengeometrien bei steigender Rekursionstiefe: (a) Tiefe 3 - (e) Tiefe 7, (f) Kameraaufbau

Kapitel 6

Diskussion

Die folgenden Abschnitte sollen einen Überblick über die geleistete Arbeit bieten sowie diese Arbeit bewerten.

6.1 Vor- und Nachteile

Die vorgestellte Implementierung in OpenGL bietet sowohl Vorteile als auch Nachteile verglichen mit einer reinen Software-Lösung, die bereits im Vorfeld der Erarbeitung der Diplomarbeit vorlag. Sie basiert auf dem Paper von [HKP+99]. Die Nachteile der reinen Software-Lösung sind wie folgt.

- Die Software-Lösung ist *langsam*. Für eine mittlere Rekursionstiefe, z.B. Tiefe 4, und einer Auflösung von 800 x 600 Pixel benötigt das Programm auf einem Pentium III 500 Mhz Rechner pro Frame schon knapp 10 Sekunden. Der zeitaufwändigste Schritt ist hier die Texturierung. Für jeden Pixel müssen drei Texturen gemischt werden.

- Es gab keine Regressionsebenen der Kameras. Für jede Szene musste außerhalb des Programms eine *globale* Ausgleichsebene berechnet und eingelesen werden.

- Es gab keinerlei Interaktionsmöglichkeiten mit dem Programm, d.h. es wurde ein statischer Kameraflug dargestellt.

Die Vorteile der hier implementierten OpenGL-basierten Variante sind wie folgt.

- Es werden bei mittlerer Rekursionstiefe, z.B. Tiefe 4, schon bei älteren Grafikkarten, z.B. NVidia Geforce 2 GTS, und älteren Rechnern, hier Pentium III 500 Mhz, flüssige Bildfolgen erreicht. Die Auflösung spielt hier keine Rolle, da die Texturierung in Hardware erfolgt.

- Für jede Kamera wird eine seperate Regressionsebene berechnet. Damit wird auf die Szenengeometrie stärker eingegangen als mit einer globalen Ausgleichsebene.

- Es besteht die Möglichkeit der freien Navigation durch die virtuelle Szene. Dies erfolgt bei geeigneter Einstellung der Rekursionstiefe in Echtzeit.

Andererseits bestehen auch Nachteile der OpenGL-basierten Version gegenüber der reinen Software-Lösung.

- Das verwendete Bildmisch-Verfahren ist ungenau. Beträgt der Anteil der ersten Textur insgesamt 0, so kann, da die erste Textur unmodifiziert gezeichnet wird, der Anteil nach Mischen der letzten beiden Texturen nicht mehr auf 0 gedrückt werden. Die erste Textur wird also zu deutlich sichtbar.

- Der Speicherbedarf ist relativ hoch. Die Breite und Höhe der Texturen müssen in zweier Potenzen vorliegen. Dies lässt sich allerdings durch Nutzung von Texturkompression oder geeignetem Textur-Management umgehen, wenn viel Speicher und eine schnelle Bus-Anbindung vorliegt.

6.2 Grenzen

Die Darstellungsqualität bei der verwendeten Szene ist visuell ansprechend. Man kann anhand der Ergebnisbilder gut sehen, wie deutlich auch kleinere Gegenstände, wie der rote Klebestift im vorderen Bereich der Szene, sichtbar bleiben und aus dem neuen Blickwinkel wirklichkeitsgetreu dargestellt wird. Dieses Verhalten

kann man auch sehr deutlich an den Lichtreflexen auf dem Locher in den Ergebnisbildern beobachten, die erwartungsgemäß gut sichtbar sind.

Das verwendete theoretische Verfahren ist dennoch nicht optimal. Der Ansatz zur anfänglichen Triangulierung erzeugt ein Dreiecksnetz, deren Knoten den Kameras nicht eindeutig zugeordnet werden können. Ausgehend von einer virtuellen Kamera werden Sichtstrahlen durch die Zentren der realen Kameras in die virtuelle Szene geworfen und den realen Kameras als Ursprungsort zugeordnet. Da aber der Sichtbereich dieser Kameras den Sichtstrahl nicht immer abdeckt, kann das Verfahren nur als grobe Näherung bezeichnet werden.

Weiterhin ist bei ungünstiger Triangulierung nur jeweils ein kleiner Szenenausschnitt sichtbar, obwohl die aufgenommene Szene deutlich größer ist. Werden mit der interaktiven virtuellen Kamera die realen Kameras innerhalb der virtuellen Szene passiert, um den Szenenausschnitt zu vergrößern, kann jedoch eine Triangulierung nicht mehr durchgeführt werden. Dazu betrachte man Abbildung 6.1. Die Sichtstrahlen der virtuellen Kamera durch die realen Kamerazentren treffen nicht die Szenengeometrie.

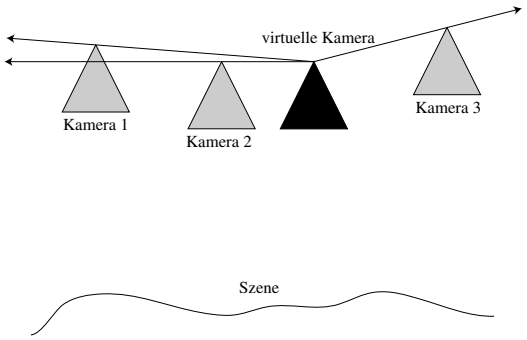


Abbildung 6.1: Triangulierung nicht möglich, wenn die virtuelle Kamera die realen Kameras passiert hat.

6.3 Ausblick

Das geschilderte Verfahren lässt sich an obigen Stellen verbessern. Es ist denkbar, statt wie bisher nur einem Sichtstrahl pro Kamera vier für den gesamten Sichtbereich der Kamera für die Triangulierung zu nutzen. Betrachtet man zusätzlich den Sichtbereich der virtuellen Kamera, kann ein Clippingbereich gefunden werden.

Weiterhin sollten für eine anfängliche Triangulierung nur Sichtstrahlen betrachtet werden, die von der jeweiligen realen Kamera ausgehen, nicht von der virtuellen. In der hier verwendeten Szene führt dies jedoch zu einem sehr kleinen Bildausschnitt.

Kapitel 7

Zusammenfassung

Im Zuge dieser Arbeit wurde ein neuartiges Verfahren beschrieben, um neue virtuelle Ansichten einer aufgenommenen Szene zu visualisieren.

Dazu wurde zunächst behandelt, wie aus mehreren mit einer Handkamera aufgenommenen Bildern deren Tiefenwerte extrahiert werden und die zugehörigen Kameraparameter berechnet werden können. Dabei wurde keinerlei Vorwissen bezüglich der verwendeten Kameras oder der betrachteten Szene benötigt.

Darauf aufbauend wurde skizziert, wie mit diesen Ergebnissen für jede reale Kamera eine Regressionsebene erstellt werden kann. Mit diesen Ebenen und den Tiefenkarten ist es möglich, mit Hilfe heutiger Hardware eine virtuelle, neue Ansicht der aufgenommenen Szene zu erstellen.

Dazu wurde ein spezielles Triangulierungs- und Gewichtungsverfahren vorgestellt. Mit Hilfe des Mischens dreier Texturen innerhalb eines Dreiecks wurde eine realitätsnahe Darstellung möglich, wobei der Schwerpunkt auf der Nutzung aktueller Graphik-Hardware zur Beschleunigung der Darstellung liegt. Damit kann eine Echtzeit-Fähigkeit erreicht werden.

Unter Verwendung von C++ und OpenGL sind die Referenzimplementierungen `LightFieldRenderGL` und das zugehörige Datenmodell `LightFieldData` entstanden, siehe hierzu Anhang A.

Anhang A

Implementierung

Zur Umsetzung des beschriebenen Verfahrens wurden zwei zentrale C++-Klassen und einige Hilfsklassen entwickelt. Diese werden im Folgenden näher erläutert.

A.1 LFData

Die Klasse `LFData` dient der Bereitstellung der benötigten Daten für die Klasse `LFRender`. Sie führt folgende Schritte durch.

- Einlesen aller Tiefenkarten
- Extrahieren der Kameraparameter aus den Tiefenkarten
- Einlesen der Bilddaten
- Vergrößern der Dimensionen der Bilddaten auf zweier Potenzen
- Einlesen der Bilddaten in OpenGL
- Erstellen der Regressionsebenen anhand der Tiefenkarten
- Bereitstellung von Zugriffsfunktionen auf alle Daten

A.2 LFRender

Die Klasse `LFRender` führt die eigentliche Render-Arbeit aus.

- Initialisierung aller benötigter Daten mittels `LFData`
- Einlesen der Kommandozeilen-Parameter
- *2D-Triangulierung*, also Projektion aller realen Kamerazentren in die Bildebene der virtuellen Kamera und anschließende Delaunay-Triangulierung
- *3D-Triangulierung*, also anfängliche Triangulierung und je nach gewünschter Dreiecksdichte weitere Rekursion mit Gewichtung und Dreiecksunterteilung
- *Darstellung neuer Ansichten*, also Übergeben des Dreiecksnetzes an OpenGL und Texturierung mit Hilfe von Blending

Anhang B

Farbbilder

Abbildung B.1: Dritte Ansicht einer Szene, auch Dreiecke gezeichnet, die nur von einer oder zwei Kameras gesehen werden

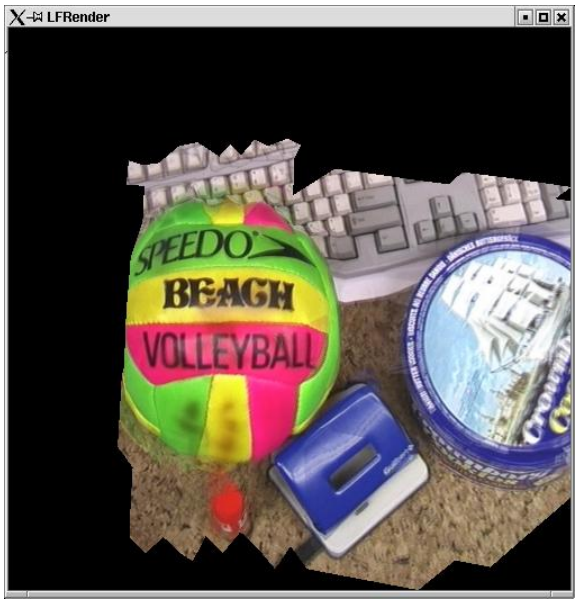


Abbildung B.2: Erste Ansicht einer Szene, nur Dreiecke gezeichnet, die von mindestens drei Kameras gesehen werden

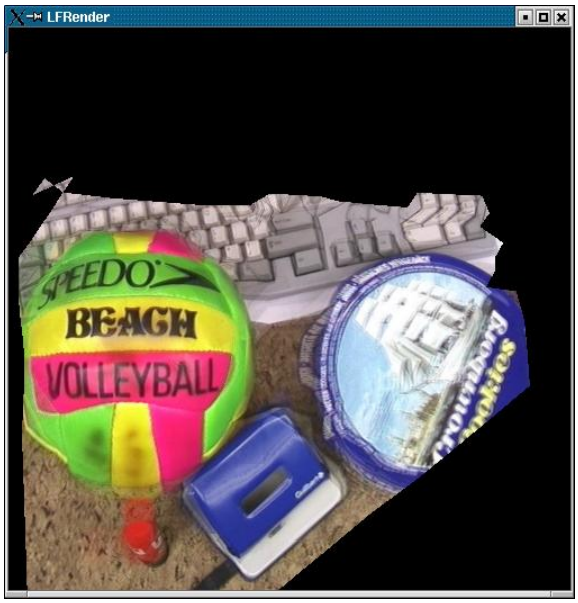


Abbildung B.3: Zweite Ansicht einer Szene, nur Dreiecke gezeichnet, die von mindestens drei Kameras gesehen werden

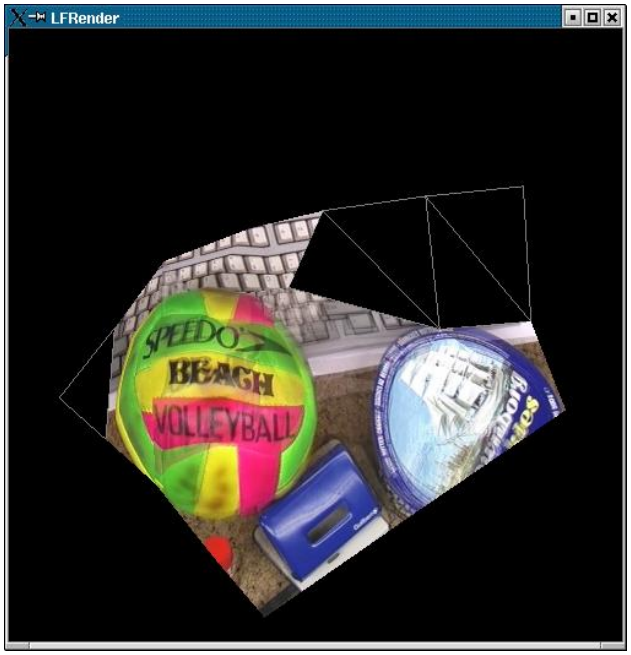


Abbildung B.4: Ansicht einer Szene mit einer groben Gitterstruktur

Abbildung B.5: Ansicht einer Szene mit einer feinen Gitterstruktur

Literaturverzeichnis

[AB91] E.H. Adelson and J. R. Bergen. The plenoptic function and the elements of early vision. In *Computational*, pages 3–20. Cambridge, Massachusetts: M.I.P Press, 1991.

[BZM97] P. Beardsley, A. Zisserman, and D. Murray. Sequential update of projective and affine structure from motion. In *International Journal of Computer Vision*, volume 23, pages 235–259. Cambridge, Massachusetts: M.I.P Press, 1997.

[GGRC96] S. Gortler, R. Grzeszczuk, R.Szeliski, and M. F. Cohen. The lumigraph. In *Proceedings SIGGRAPH '96*, pages 43–54. ACM Press, New York, 1996.

[HKP+99] B. Heigl, R. Koch, M. Pollefeys, J. Denzler, and L. Van Gool. Plenoptic modeling and rendering from image sequences taken by a handheld camera. In *Proceedings 21. Symposium für Mustererkennung (DAGM '99)*, pages 94–101. Bonn, September 1999.

[LP96] M. Levoy and P.Hanrahan. Lightfield rendering. In *Proceedings SIGGRAPH '96*, pages 31–42. ACM Press, New York, 1996.

[Pol99] M. Pollefeys. *Self-calibration And Metric 3D Reconstruction From Uncalibrated Image Sequences*. PhD thesis, Department ESAT, Khatolike Universiteit, Leuven, 1999.

[She96] Jonathan Richard Shewchuk. Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator. In Ming C. Lin and Di-

nesh Manocha, editors, *Applied Computational Geometry: Towards Geometric Engineering*, volume 1148 of *Lecture Notes in Computer Science*, pages 203–222. Springer-Verlag, May 1996. From the First ACM Workshop on Applied Computational Geometry.

[Woe01] Jan Woetzel. Projektive 3D-Rekonstruktion durch Bildfolgenanalyse monokularer Kameras mit adaptierbarer Korrespondenzsuche. Master's thesis, Christian-Albrechts-Universität zu Kiel, 2001.

Danksagungen

Diese Diplomarbeit wurde am Lehrstuhl Multimediale Systeme zur Informationsverarbeitung der Christian-Albrechts-Universität zu Kiel angefertigt. Ich danke allen Mitarbeitern des Lehrstuhls für die Hilfsbereitschaft und das gute Arbeitsklima.

Bei meinem Betreuer Herrn Dipl.-Ing. Jan-Friso Evers-Senne bedanke ich mich besonders für die gute fachliche und menschliche Betreuung und die hervorragende Unterstützung. Ich danke Herrn Prof. Dr.-Ing. Reinhard Koch für das Angebot des interessanten Themas und die vielen Anregungen zu dieser Arbeit.

Weiterhin bedanke ich mich bei meiner Freundin Stefanie für ihr Verständnis und ihre Motivierung und Aufmunterung, sowie bei meinem Bruder Frank, der mit seiner spritzigen Heiterkeit für so manche Ablenkung sorgte. Ein großer Dank gilt meinen Eltern, ohne deren Motivierung manches nicht geklappt hätte.

Erklärung

Ich versichere, dass ich die vorliegende Arbeit selbstständig verfasst und ausschließlich die angegebenen Hilfsmittel und Quellen verwendet habe.

Kiel, im August 2002