Seminar

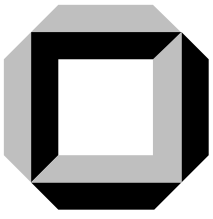
Image-based Rendering

Neue Techniken der photorealistischen Bilderzeugung

Richard Hoffmann, Michaela Karcher, Armin Müller,
Markus Noga, Stefan Preuß, Fred Schättgen,
Thorsten Scheuermann, Erik Sohns, Jörg Tovar

Betreuung
Michael Fautz, Peter Oel, Sven Thüring

Wintersemester 1999/2000



Institut für Betriebs- und Dialogsysteme
Fakultät für Informatik
Universität Karlsruhe

Zusammenfassung

In den letzten Jahren zeigte sich, daß einige Aufgaben im Bereich der Bilderzeugung mit rein geometriebasierten Verfahren nur schlecht oder gar nicht bewältigt werden können. Denkt man zum Beispiel an ein Stofftier, so ist eine exakte geometrische Modellierung, auf deren Basis dann neue Bilder berechnet werden, sicher schwer anzugeben.

Schon in den 60er-Jahren wurde die Idee entwickelt, hinreichend viele Kameraaufnahmen einer Szene zu machen und bei Bedarf die zu einer bestimmten Blickrichtung gehörende Aufnahme aus dem Vorrat zu nehmen (Movie Map). Erst heute reichen Speicher- und Rechenleistung der Computer aus, um Verfahren dieser Art zu verwirklichen. Es wird also versucht, mit Hilfe vorher von einer Szene aufgenommener Bilder neue Ansichten der Szene zu erzeugen. Dabei werden nicht nur auf schon vorhandene Aufnahmen zurückgegriffen und diese unverändert verwendet, sondern man möchte mit möglichst wenig Aufnahmen auskommen und fehlende Bilder aus schon vorhandenen berechnen. Zu diesem Zweck wird keine Geometrie aus den zuvor aufgenommenen Bildern ermittelt. Es wird nur auf den Bilddaten gearbeitet, also neue Bilder aus vorhandenen berechnet: Image-based Rendering (IBR).

Das Seminar *Image-based Rendering – Neue Techniken der photorealistischen Bilderzeugung* wurde im Wintersemester 1999/2000 am Institut für Betriebs- und Dialogsysteme, Fakultät für Informatik der Universität Karlsruhe durchgeführt.

Teilnehmer

Richard Hoffmann
Michaela Karcher
Armin Müller
Markus Noga
Stefan Preuß
Fred Schättgen
Thorsten Scheuermann
Erik Sohns
Jörg Tovar

Betreuung

Michael Fautz
Peter Oel
Sven Thüring

Prof. Dr. Alfred Schmitt
Institut für Betriebs- und Dialogsysteme
Fakultät für Informatik
Universität Karlsruhe

Inhaltsverzeichnis

1 Light Field Rendering

Thorsten Scheuermann

Dieser Text beschreibt das von Marc Levoy und Pat Hanrahan in [Marc Levoy '96] vorgestellte Light Field Rendering-Verfahren. Es handelt sich dabei um einen Image-based Rendering Ansatz, bei dem aus vorhandenen Aufnahmen eines realen oder synthetischen Objekts neue Ansichten von beliebiger Betrachterposition erzeugt werden können.
Die vorhandenen Aufnahmen werden dazu als Abtastung der plenoptischen Funktion, die den Lichtfluß durch den Raum beschreibt, interpretiert.

1.1 Repräsentation

1.1.1 Die plenoptische Funktion

Die plenoptische Funktion ist ein Modell zur Szenenbeschreibung ohne Verwendung von Geometrieinformationen und Oberflächeneigenschaften der in der Szene enthaltenen Objekte, die von Adelson und Bergen in [Adelson E.H. '91] vorgestellt wurde. Ausgangspunkt für die Modellbildung ist das menschliche Auge, mit dem die visuellen Eigenschaften eines Objekts nur indirekt über eintreffende Lichtstrahlen wahrgenommen werden. In dieser Sichtweise definiert eine Szene also die Eigenschaften einer Vielzahl von Lichtstrahlen.
Die Funktion, die alle diese Lichtstrahlen beschreibt, heißt *plenoptische Funktion* (von lat. plenus, vollständig), da sie alle visuellen Eigenschaften einer Szene definiert. Sie ordnet jedem orientierten Strahl im kartesischen Raum eine Farbe zu. Der Strahl wird definiert durch den Augenpunkt (P_x, P_y, P_z) des Betrachters und einer Blickrichtung (θ, ϕ), gegeben als Winkelpaar:

$$Plen\colon (P_x, P_y, P_z, \theta, \phi) \rightarrow Farbe$$

. Hanrahan und Levoy nennen diese Funktion *Light Field*.

1.1.2 Reduktion auf vier Dimensionen

Diese 5D-Repräsentation lässt sich in einem hindernisfreien Raum auf vier Dimensionen reduzieren. Der Grund dafür ist, dass sich die Farbe auf einem Strahl nicht ändert, es sei denn, er wird von einem Hindernis blockiert. Damit muss man nicht jedem Punkt des Raums für jede Richtung eine Farbe zuordnen, sondern

nur jedem Strahl, der durch den hindernisfreien Raum verläuft. Ein solcher lässt sich mit vier Parametern eindeutig beschreiben.
Die Einschränkung auf einen freien Raum ohne Hindernisse scheint zwar gravierend, ist in zwei üblichen Fällen aber nützlich. Bei einem Objekt mit einer begrenzten Geometrie ist der Raum außerhalb dessen konvexer Hülle hindernisfrei. Wenn man Ansichten einer Szene, die den Betrachter umgibt, erzeugen will und die Kamera sich innerhalb eines konvexen leeren Gebiets bewegt, ist dieses ebenfalls frei von Hindernissen, da in das Gebiet eintretende Lichtstrahlen darin nicht mehr blockiert werden können. In diesem Fall kann also ebenfalls eine 4D-Repräsentation gewählt werden.

1.1.3 Parametrisierung

Es stellt sich die Frage, wie der Raum der orientierten Strahlen parametrisiert werden soll. Die Parametrisierung sollte dabei einige Eigenschaften erfüllen:

Effiziente Berechnungen: Für die Darstellung neuer Ansichten ist es wichtig, dass die zugehörigen Strahlparameter schnell aus einem Augenpunkt und einer Pixelposition auf der Bildebene ermittelt werden können.

Kontrolle über die Strahlenmenge: Aus dem unendlichen Strahlenraum wird immer nur eine endliche Teilmenge benötigt (z.B. nur die Strahlen, welche die konvexe Hülle des darzustellenden Objekts schneiden). Deshalb wäre es hilfreich, wenn ein intuitiver Zusammenhang zwischen den Strahlen im kartesischen Raum und deren Parameter bestünde.

Gleichmäßige Abtastung: Bei gleichmäßig abgetasteten Samples im Parameterraum, bei dem jeder Parameter eine Dimension darstellt, sollte das Muster der Geraden im kartesischen Raum (dem dreidimensionalen Anschauungsraum mit den Koordinaten (x, y, z)) ebenfalls gleichmäßig sein.

Die vorgeschlagene Lösung ist, die Strahlen durch ihre Schnittpunkte mit zwei Ebenen in allgemeiner Lage zu parametrisieren. Das Koordinatensystem auf der ersten Ebene ist dabei (u, v), das auf der zweiten (s, t). Weiterhin soll $u, v, s, t \in [0, 1]$ gelten, d.h. die Schnittpunkte auf den Ebenen liegen jeweils in einem konvexen Viereck.
Diese Repräsentation wird *Light Slab* genannt (von engl. slab, Platte). Ein Light Slab repräsentiert alle Lichtstrahlen, die durch ein Viereck eintreten und durch das Viereck auf der anderen Ebene wieder austreten. Eine der Ebenen darf im Unendlichen liegen. Dies ermöglicht die Verwendung von Bildern, die Parallelprojektionen sind, oder alle mit demselben konstanten Sichtfeld aufgenommen wurden, zur Erzeugung eines Light Field. Bei durchgehender Verwendung von homogenen Koordinaten können diese Fälle ohne zusätzlichen Aufwand behandelt werden.

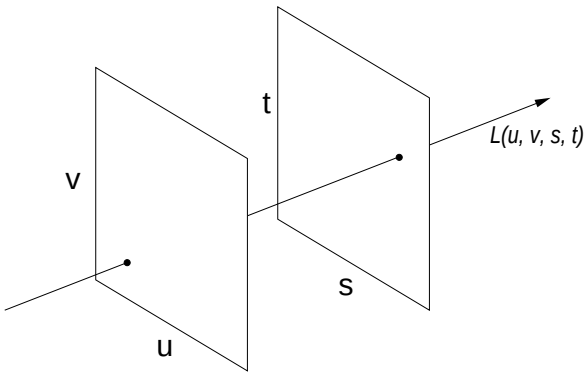


Abbildung 1.1: Ein Light Slab.

1.2 Sampling

1.2.1 Abtastung des Strahlenraums

Es stellt sich die Frage nach der genauen Gestalt und Lage der Light Slabs, also wie der Strahlenraum abgetastet werden soll. Unter der Annahme, dass alle möglichen Ansichten mit gleicher Wahrscheinlichkeit erzeugt werden, wird jeder Strahl mit gleicher Wahrscheinlichkeit benötigt. Also sollte die Sampledichte überall gleich sein. Dabei ist keine Ebenenanordnung perfekt, aber bei Light Slabs mit parallelen Ebenen ergibt sich ein relativ gleichmäßiges Muster, das sich noch weiter verbessert, wenn eine Ebene im Unendlichen liegt. Diese Geometrie der parallelen Ebenen wird praktisch ausschließlich verwendet. Darüber hinaus sollte aus Symmetriegründen der Abstand der Samples auf der uv- und st-Ebene gleich sein. Wenn sich der Beobachter bei den zu erzeugenden Ansichten jedoch nahe bei der uv-Ebene aufhält, Kann es akzeptabel sein, diese Ebene mit kleinerer Auflösung abzutasten. Dies wird insbesondere bei Light Fields aus realen Szenen genutzt, da eine höhere Sampledichte auf der uv-Ebene mehr aufzunehmende Bilder und damit eine längere Aufnahmezeit bedeutet.
Mit nur einem Light Slab kann man aber nicht alle verschiedenen Richtungen erfassen. Deshalb werden bei Light Fields, die von allen Seiten betrachtet werden sollen, mehrere Light Slabs aus unterschiedlichen Richtungen aufgenommen, so dass sich eine geschlossene Anordnung ergibt (siehe Abbildung 1.2).

1.2.2 Antialiasing

Wie jeder Abtastungsprozeß kann auch die Abtastung eines Light Fields zu Aliasing führen, wenn im Light Field hohe Frequenzen auftreten. Die negativen Auswirkungen des Aliasing können gemildert werden, indem man vor der Abtastung filtert. Dabei muss ein 4D-Tiefpassfilter im Strahlenraum angewendet werden. Wenn man von einem Boxfilter ausgeht, muss der Mittelwert der Farben aller Strahlen, die ein Samplequadrat auf der uv- und st-Ebene schneiden, berechnet

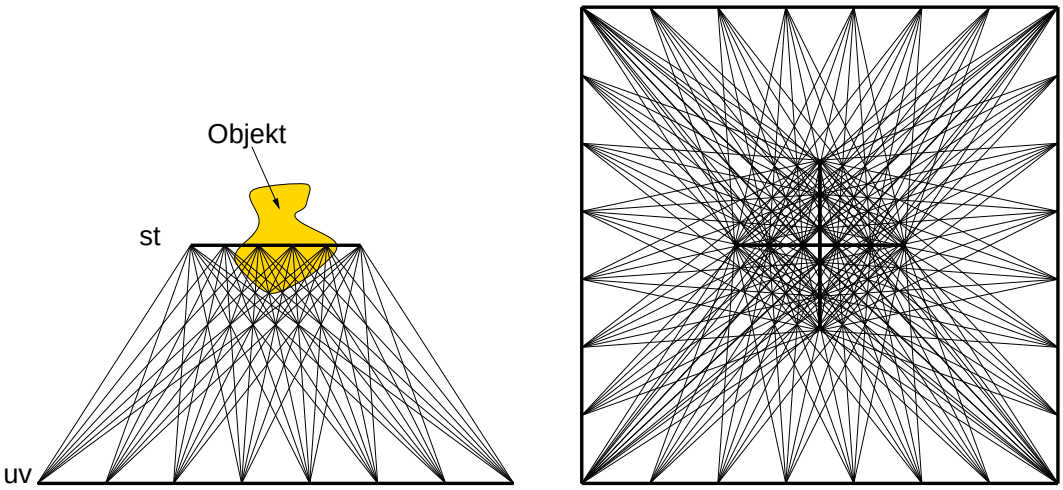


Abbildung 1.2: Links: Light Slab mit parallelen Ebenen, Rechts: vier rotierte Kopien decken alle Blickrichtungen ab.

werden. Durch Grenzübergang ergibt sich ein mehrdimensionales Integral (hier der zweidimensionale Fall):

$$\lim_{\substack{n\to\infty \\ m\to\infty}} \frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} color(\overline{p_i q_j}) = \int_{x=0}^{1} \int_{y=0}^{1} color(\overline{xy})\, dy\, dx$$

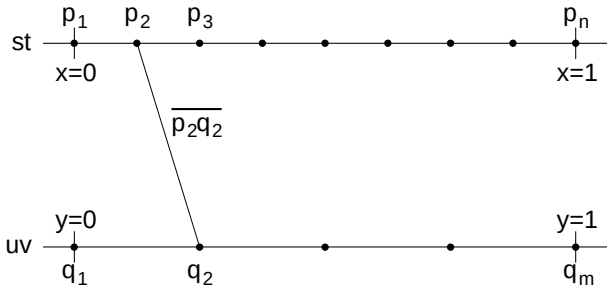


Abbildung 1.3: Zur Herleitung des Tiefpassfilters.

Dabei sind die Summenbildung und Integration komponentenweise zu verstehen, denn $color(\overline{ab})$ bezeichnet die Farbe, die ja ein Vektor ist, des Strahls, der die Punkte a und b verbindet. Der Übergang auf vier Dimensionen ist einfach.
Diese Filterung kann in einen Pixelfilter und einen Blendenfilter zerlegt werden (Abbildung 1.4). Die Pixelfilterung kann zum Beispiel näherungsweise durch Aufnahme der Bilder mit höherer Auflösung und anschließender geeigneter Verkleinerung erreicht werden. Bei Aufnahmen mit einem realen optischen System wird die Blendenfilterung von der Linse durchgeführt, weil jeder Strahl von einem Punkt auf der st-Ebene durch die Linse auf denselben Punkt der Bildebene abgebildet wird, wobei die Mittelung durchgeführt wird. Dies geschieht jedoch nur dann

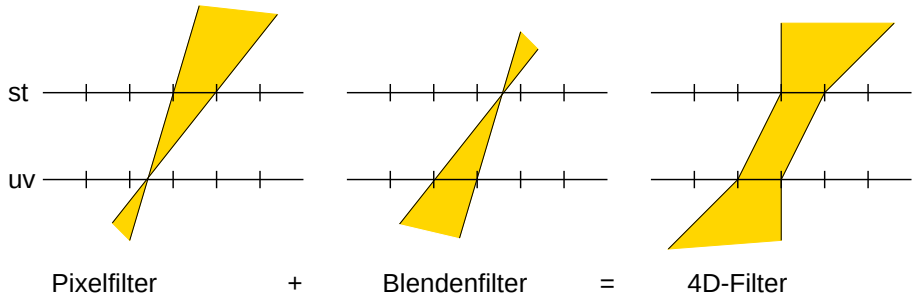


Abbildung 1.4: Zerlegung des Filters.

vollständig, wenn die Abstände der Abtastpunkte auf der uv-Ebene nicht größer als die Blendenöffnung sind.

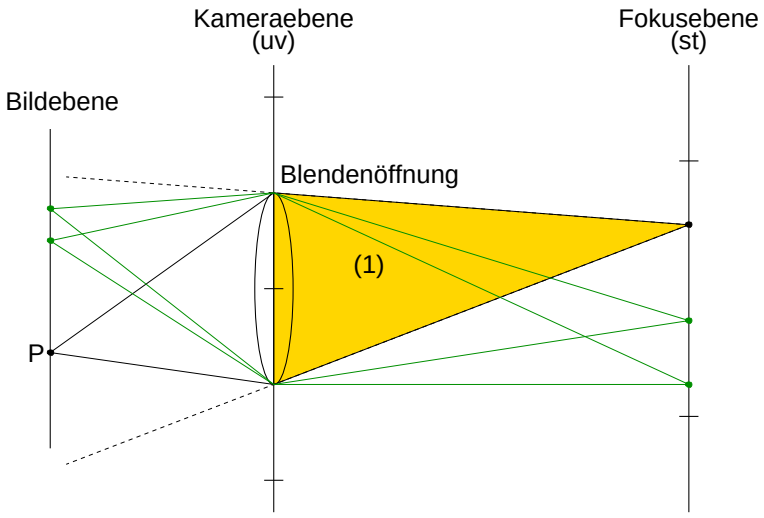


Abbildung 1.5: Blendenfilterung bei optischen Systemen. Das ganze Strahlenbündel (1) wird auf denselben Punkt P abgebildet.

1.2.3 Aufnahme von Light Fields

Um ein Light Field zu erzeugen, muss nach Auswahl eines Abtastmusters für eine Vielzahl von Strahlen deren Farbe ermittelt werden. Dies ist in virtuellen Umgebungen durch Einsatz eines Raytracers machbar, bei realen Szenen ist die Erfassung einzelner Lichtstrahlen jedoch nicht praktikabel. Ein durchführbarer Ansatz besteht darin, eine Menge von Bildern aufzunehmen und daraus die Farben der einzelnen Strahlen zu bestimmen.
In virtuellen Umgebungen kann ein Light Slab einfach durch Berechnung eines 2D-Feldes von Bildern erzeugt werden. Bei jedem Bild liegt dabei das Projektionszentrum der Kamera auf einem Samplepunkt der uv-Ebene. Außerden müssen die xy-Pixel des Bildes genau mit dem Raster auf der st-Ebene zusammenfallen. Dies

kann durch eine gescherte Perspektivprojektion erreicht werden (siehe Abbildung 1.6).

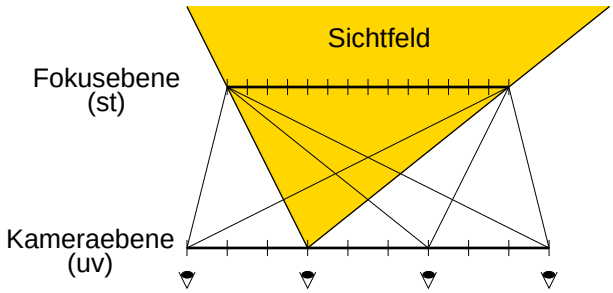


Abbildung 1.6: Gescherte Perspektivprojektionen bei der Aufnahme eines Light Slab.

Das sich daraus ergebende Light Slab kann entweder als uv-Feld von st-Bildern oder als st-Feld von uv-Bildern visualisiert werden, wie dies in Abbildung 1.7 zu sehen ist.

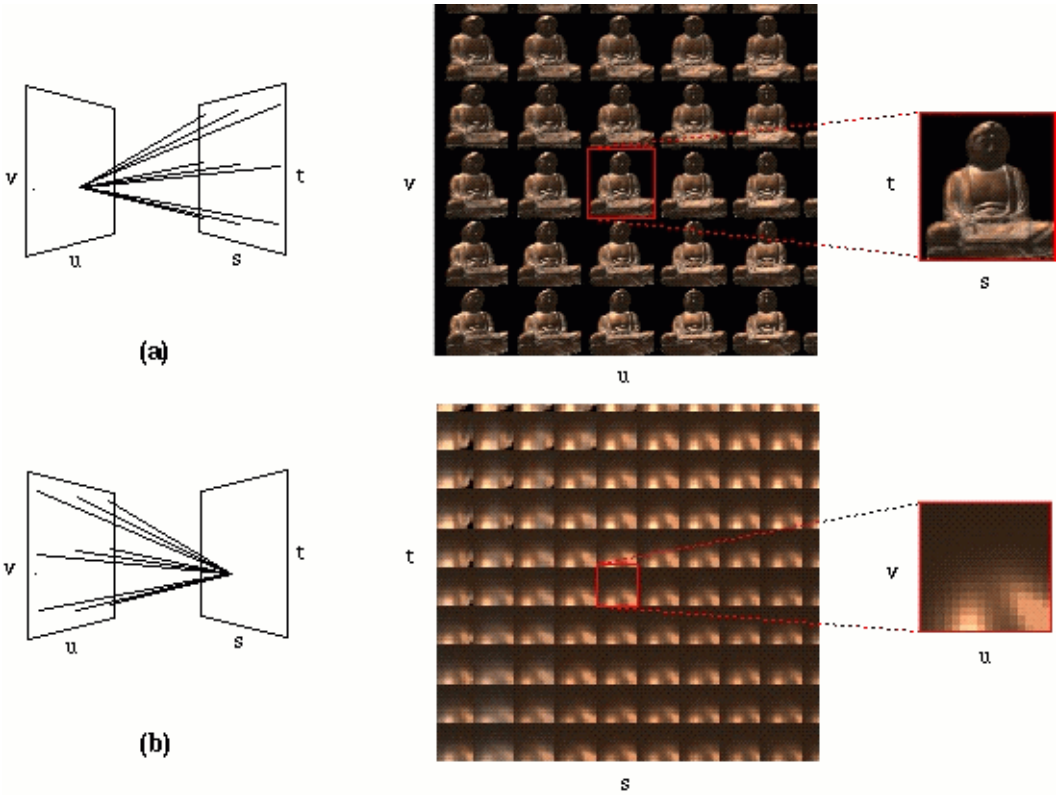


Abbildung 1.7: Zwei Visualisierungen eines Light Fields.

1.2.4 Aufnahme von Light Fields realer Szenen

Bei der Erzeugung von Light Fields von realen Szenen und Objekten werden eine Reihe von Bildern von Position auf der uv-Ebene aufgenommen. Folgende Punkte müssen dabei berücksichtigt werden:

- Die große Anzahl benötigter Bilder bei der Aufnahme eines Light Field (hunderte bis tausende) und die genaue Positionierung der Kamera erfordert Automatisierung, also eine maschinell geführte Kamera.
- Die Eigenschaften der bei der Bildaufnahme verwendeten optischen Systeme müssen berücksichtigt werden. Dazu gehören insbesondere Tiefenschärfe, Blendeneinstellung und Kamerasichtfeld. Aus diesem Grund wird zur Aufnahme eine kalibrierte Kamera eingesetzt.

Im Folgenden sollen die Designentscheidungen für die in Stanford verwendete Aufnahmevorrichtung erläutert werden:

Computer- oder Handsteuerung: Ein günstiger Ansatz, Light Fields zu erzeugen, besteht darin, eine Handkamera durch die Szene zu bewegen und darin die Bilder aufzunehmen. Dieses Verfahren bringt einige Schwierigkeiten mit sich und wird im Kapitel über den Lumigraph ausführlich behandelt. Der Einfachheit halber wurde deshalb eine computergeführte Aufnahmevorrichtung verwendet, welche die Bilder auf einem gleichmäßigen Gitter aufnimmt.

Kamerabewegung auf Kugel oder Ebene: Bei Quick-Time VR wird für die Aufnahme von Object Movies eine Vorrichtung verwendet, bei der sich die Kamera auf einer Kugeloberfläche bewegt und ein Objekt im Mittelpunkt fotographiert. Leider ist dabei die Lichtquelle an der sich bewegenden Kamera montiert, weshalb die Vorrichtung ungeeignet ist, ein statisches Light Field aufzunehmen. Stattdessen wurde trotz einiger Nachteile eine Vorrichtung verwendet, bei der die Kamera auf einer Ebene bewegt wird. Zum einen war diese recht einfach zu konstruieren, außerdem passt das besser zum Light Slab-Modell.

Beleuchtung: Die viermalige Drehung des Objekts bei der Aufnahme erfordert mehr Aufwand bei der Beleuchtung. Um statisch zu erscheinen, muss sie sich entweder mit dem Objekt mitdrehen, oder vierfach symmetrisch ausgelegt sein, so dass sie von jeder der vier Seiten gleich aussieht. Darüber hinaus darf die sich bewegende Kamera keine Schatten werfen, die in der Aufnahme zu sehen wären.

Sichtfeld: Ein Ziel war es, Light Fields aufzunehmen, bei denen man ein Objekt komplett um die eigene Achse drehen kann. Um das mit der ebenen Aufnahmevorrichtung zu erreichen, müssen vier um 90 Grad gedrehte Light Slabs

aufgenommen werden. Um einen Light Slab aufzunehmen könnte eine Kamera verwenden, die sich nur auf der Ebene bewegen, jedoch nicht um das Projektionszentrum rotieren kann, indem man ein Weitwinkelobjektiv verwendet und dann immer einen entsprechenden Bildausschnitt verwendet. Die Probleme hierbei sind, dass Weitwinkelobjektive das Bild stark verzerren und außerdem die Ausschnittswahl eine Verringerung der Auflösung bedeutet. Die realisierte Aufnahmevorrichtung erlaubt daher die Rotation der Kamera um zwei Achsen, was die Verwendung eines Objektivs mit größerer Brennweite erlaubt. Wegen der Rotation müssen die Bilder jedoch auf eine gemeinsame Ebene reprojeziert werden.

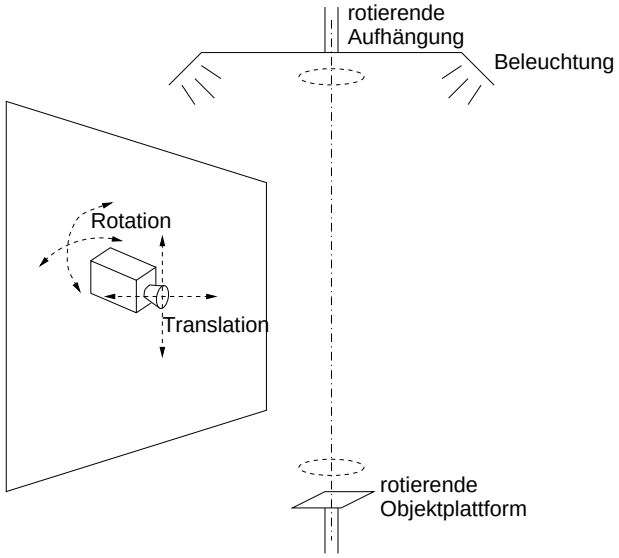


Abbildung 1.8: Schematischer Aufbau der Aufnahmevorrichtung.

Objektabstand: Ein Nachteil der ebenen Kamerabewegung ist der sich abhängig von der Kameraposition ändernde Abstand zum aufzunehmenden Objekt. Dies erschwert es, ein scharfes Bild zu erhalten. Es ist im Prinzip möglich, den Fokus nachzuregeln, aber dabei würde die Verschiebung des Projektionszentrums und Veränderung des Vergrößerungsfaktors zu weiteren Schwierigkeiten führen. Stattdessen wird versucht, die Auswirkungen dieses Fehlers durch gute Beleuchtung und eine kleine Blendeneinstellung, um die Tiefenschärfe zu verbessern, zu verringern.

Bei der Aufnahme eines Light Fields bewegt sich die Kamera für jeden der vier Light Slabs durch eine Anzahl von Gitterpunkten auf einer Ebene und macht dort jeweils eine Aufnahme. Dabei wird sie so gedreht, dass sie immer auf das Objekt im Mittelpunkt zeigt. Da dadurch auch die Bildebene gedreht wird, müssen die aufgenommenen Bilder auf eine gemeinsame Ebene reprojiziert werden, um die erforderlichen gescherten Perspektivprojektionen zu erhalten.

1.3 Kompression

Aufgrund der großen anfallenden Datenmenge bis in den Gigabytebereich ist eine Kompression unumgänglich. Bei der Auswahl eines Kompressionsverfahrens muss auf folgende Eigenschaften von Light Fields eingegangen werden:

Redundanz: Ein gutes Kompressionsverfahren entfernt die Redundanz in einem Signal, ohne dessen Inhalt zu verändern. Light Fields enthalten Redundanz in allen vier Dimensionen. In s und t erwartet man Redundanz wie in jedem anderen Pixelbild, in u und v ist die Redundanz ähnlich der einer Bewegtbildsequenz.

Wahlfreier Zugriff: Die meisten Kompressionsverfahren schränken den wahlfreien Zugriff auf die komprimierten Daten stark ein. Oft müssen ganze Blöcke auf einmal dekodiert werden. Dies ist zum Beispiel bei Vektorquantisierung mit variabler Bitrate, Huffmankodierung oder den arithmetischen Kodierern, wie sie bei JPEG und MPEG eingesetzt werden, der Fall. Wenn Prädiktionsverfahren eingesetzt werden wird es noch komplizierter, weil die Kodierung eines Pixels dann von seinen Vorgängern abhängig ist. Bei Light Fields ist das ein Problem, weil die beim Resampling benötigten Pixelinformationen weit über den Speicher verstreut sind. Wenn sich der Beobachter bewegt, ändern sich die Zugriffsmuster auf komplexe Weise. Deshalb ist ein Kompressionsverfahren erforderlich, dass einen günstigen wahlfreien Zugriff auf die einzelnen Samples zulässt.

Asymmetrie: Kompressionsverfahren können aufgrund ihres Verhältnisses von Kodierungsdauer zu Dekodierungsdauer in symmetrische und asymmetrische Verfahren klassifiziert werden. Da Light Fields vor der Betrachtung aufgenommen und kodiert werden, ist dies eine asymmetrische Anwendung.

Berechnungsaufwand: Es wird ein Kompressionsverfahren benötigt, dass ohne Hardwareunterstützung auskommt und möglichst effizient ist, da außer für die Dekompression auch Rechenzeit für den Anzeigealgorithmus aufgewendet werden muss.

Levoy und Hanrahan wählten eine zweistufige Kompressionspipeline bestehend aus Vektorquantisierung mit fester Bitrate und anschließender Entropiekodierung mit dem Lempel-Ziv-Algorithmus.

1.3.1 Vektorquantisierung

Vektorquantisierung ist ein verlustbehaftetes Kompressionsverfahren, bei dem ein Vektor aus Samples auf den ähnlichsten aus einer Anzahl von Repräsentantenvektoren abgebildet wird. Die Repräsentanten heißen Codewörter und die Menge der Codewörter die zur Verfügung stehen, um eine Quelle zu kodieren, wird Codebuch

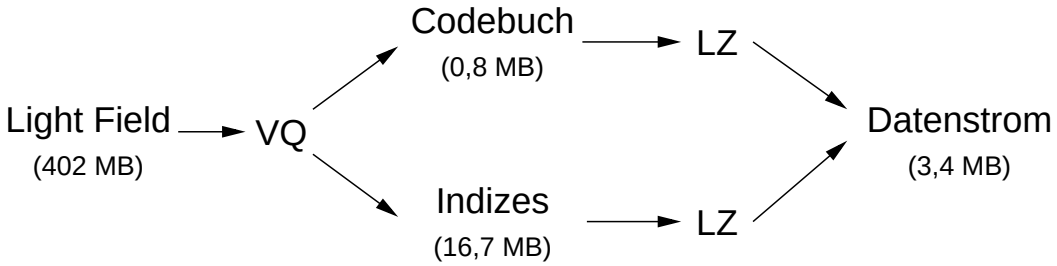


Abbildung 1.9: Kompressionspipeline.

genannt. Das Codebuch wird während einer Trainingsphase konstruiert, bei der ein Quantisierungsalgorithmus die Menge von Codewörtern finden muss, welche eine Menge von Samplevektoren (der *Trainingsmenge*) am besten approximiert. Sobald das Codebuch vorhanden ist, kann die Quelle kodiert werden, indem man sie in Vektoren zerlegt, zu jedem Vektor das ähnlichste Codewort findet und nur noch dessen Index speichert. Die Dekodierung beschränkt sich darauf, zu jedem Index das Codewort nachzuschlagen und auszugeben. Die Dekodierungsgeschwindigkeit ist einer der Hauptvorteile der Vektorquantisierung.

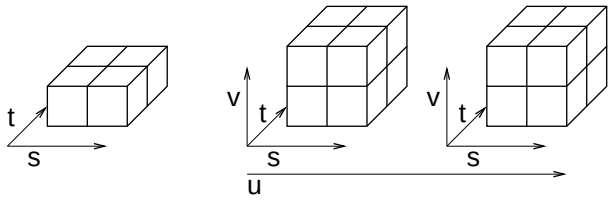


Abbildung 1.10: Zusammenfassung der Samples zu Vektoren. Links: 2D, Rechts: 4D.

In der Praxis wurden sowohl 2D- als auch 4D-Blöcke der Light Fields mit Kantenlänge 2 verwendet. Mit den RGB-Farbkomponenten ergaben sich damit 12- und 48-dimensionale Vektoren. Im ersten Fall wird nur die Redundanz in s und t genutzt.
Um eine möglichst gute Bildqualität zu gewährleisten, wird die Trainingsphase auf einer repräsentativen Untermenge des Light Fields ausgeführt und das Codebuch zusammen mit den Codeindizes gespeichert. Die Beschränkung der Trainingsmenge auf einen Bruchteil der Größe des Light Fields spart viel Rechenzeit. Die Ausgabe der Vektorquantisierung ist eine Sequenz von Codewortindizes fester Länge. Jeder Index besteht aus $\log N$ Bits, wobei N die Größe des Codebuchs ist. Also ergibt sich für die Kompressionsrate R der Wert $R = \frac{kl}{\log N}$. Dabei steht k für die Dimension der Vektoren und l für die Anzahl der Bits pro Element, üblicherweise gilt also $l = 8$. In der Praxis wurden meistens Codebücher mit $2^{14} = 16384$ Einträgen verwendet, was im ersten Abschnitt der Pipeline eine Kompressionsrate von $R = \frac{48 \times 8}{\log 16384} = \frac{384}{14} = 27 : 1$ ergibt. Um die Dekodierung zu vereinfachen, wird jeder Index durch eine ganze Anzahl von Bytes, in diesem

Fall 2, repräsentiert, was die Kompressionsrate leicht auf 24 : 1 verringert.

1.3.2 Entropiekodierung

In der zweiten Stufe der Kompressionspipeline wird `gzip`[1], eine Implementierung der Lempel-Ziv-Kodierung, verwendet, um den Speicherplatzbedarf weiter zu verringern. Da die aufgenommenen Objekte üblicherweise vor einem einfarbigen Hintergrund abgebildet sind, gibt es nach der Vektorquantisierung Indizes, die mit hoher Wahrscheinlichkeit auftreten. Lempel-Ziv-Kodierung verringert die Kosten der Repräsentation dieser Codeindizes. Dadurch ließ sich in den Beispielen aus [Marc Levoy '96] eine Kompressionsrate von etwa 5 : 1 erzielen. Die endgültige Kompressonsrate ist also $24 \times 5 : 1 = 120 : 1$.

1.3.3 Dekompression

Die Dekompression erfolgt in zwei Schritten. Das vektorquantisierte Light Field wird durch `gzip` dekomprimiert und in den Speicher geladen, so dass nun ein wahlfreier Zugriff auf die Samples möglich ist. Bei der Erzeugung einer Ansicht des Light Fields werden Strahlanfragen in Form von Koordinatentupeln gestellt. Für jede Anfrage wird berechnet, in welchem Samplevektor das gesuchte Sample zu finden ist und dessen Index aus dem Feld gelesen. Nun kann der Farbwert des gesuchten Strahls aus dem Codewort mit dem entsprechenden Index ermittelt werden.

1.4 Resampling

Um aus einem Light Field Bilder zu generieren, muss eine zweidimensionale „Scheibe“ aus dem 4D-Array der Lichtstrahlen extrahiert werden. Benötigt werden die Strahlen, welche durch das Projektionszentrum und einen Pixelmittelpunkt auf der Bildebene verlaufen. Diese werden Bildstrahlen genannt. Das Vorgehen teilt sich in zwei Schritte:

1. Berechnung der Strahlparameter (u, v, s, t) für jeden benötigten Bildstrahl
2. Ermittlung des Farbwerts für den Strahl

Die gewählte Light Slab-Repräsentation ermöglicht eine effiziente Berechnung der Strahlparameter, was ja eine der Anforderungen an die Parametrisierung war. Zum einen können die Parameter (u, v) und (s, t) einfach durch Schnitt des Bildstrahls mit den beiden Ebenen des Light Slab ermittelt werden (siehe Abbildung 1.11). Damit könnte jeder Raytracer einfach für die Verwendung von Light Fields

[1] `gzip` ist ein Unix-Tool zur Kompression und Dekompression von Dateien nach dem Lempel-Ziv-Algorithmus, das heute auf praktisch allen Rechnerplattformen zur Verfügung steht..

erweitert werden. Aber auch ein polygonbasiertes Renderingsystem kann zur Anzeige eines Light Slab verwendet werden. Die Transformation von Bildkoordinaten (x, y) auf (u, v)- und (s, t)-Koordinaten kann durch Texturemapping erfolgen. Dazu werden die uv- und st-Vierecke unter der aktuellen Kameratransformation in eine Pixeldarstellung konvertiert. Die dabei berechneten Texturkoordinaten sind gerade die gesuchten Strahlparameter.

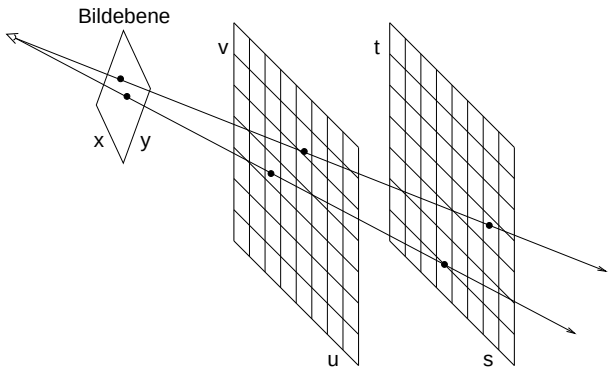


Abbildung 1.11: Erzeugung von neuen Ansichten aus einem Light Field.

Um ein Light Field darzustellen, das aus mehreren Light Slabs besteht, werden diese einfach nacheinander gezeichnet. Light Slabs, bei denen von der aktuellen Kameraposition aus die st-Ebene vor der uv-Ebene liegt, sind von dieser Position nicht sichtbar und können übergangen werden.

Abbildung 1.12: Keine, bilineare und quadralineare Interpolation (von links nach rechts).

Im zweiten Schritt wird der Farbwert für den entsprechenden Bildstrahl ermittelt. Dabei muss ein Resampling vorgenommen werden, da meistens nicht genau der gewünschte Strahl im Light Field vorhanden ist. Zur Vereinfachung wurde in der Implementierung der Resampling-Prozess durch Interpolation der Nachbarsamples approximiert. Dies ist nur dann korrekt, wenn die neue Abtastrate größer als die ursprüngliche ist. Gerade das ist bei der Anzeige von Light Fields aber

normalerweise der Fall, weil sie aufgrund des hohen Speicherbedarfs meistens mit moderater Auflösung aufgenommen werden.
Abbildung 1.12 zeigt den Vergleich zwischen keiner Interpolation (Wahl der Farbe des nächsten Strahls), bilinearer Interpolation in uv und quadralinearer Interpolation. Letztgenannte Variante liefert das beste Ergebnis, benötigt aber auch den größten Rechenaufwand.

1.5 Ergebnisse

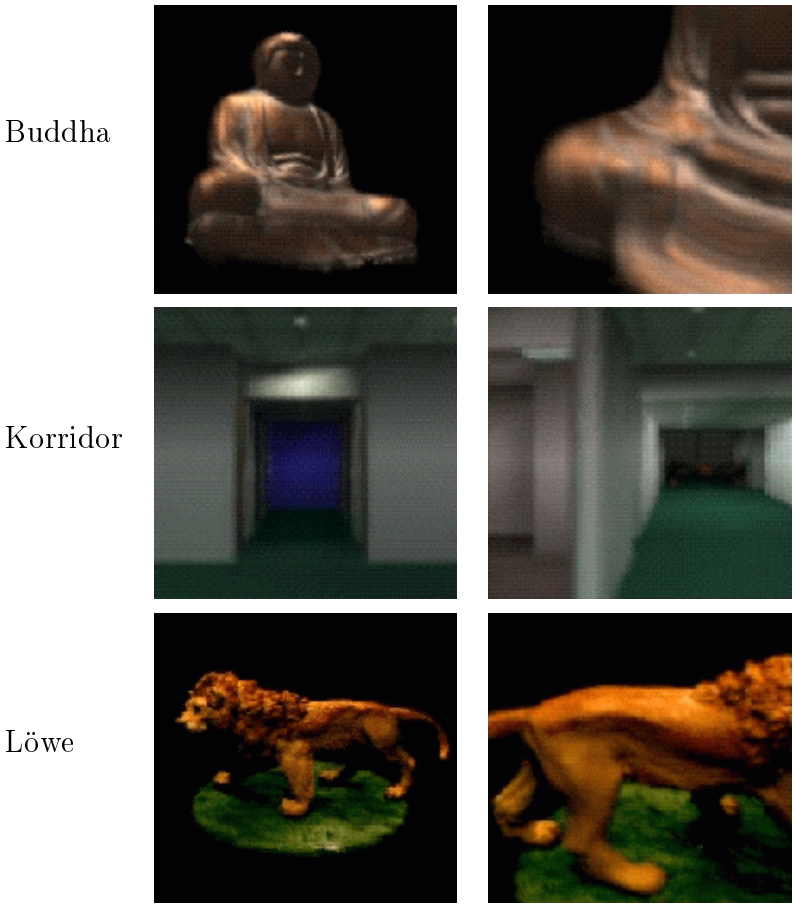


Abbildung 1.13: Beispiele aus Light Fields erzeugter Bilder.

Abbildung 1.13 zeigt Bilder, die aus Light Fields erzeugt wurden. Das erste ist das Light Field einer Buddha-Statue, das aus gerenderten Bildern berechnet wurde. Es wurde ein einzelner Light Slab verwendet.
Das zweite Beispiel ist ein nach außen gerichtetes Light Field, mit vier um 90 Grad gedrehten Light Slabs, ähnlich wie in Abbildung 1.2. Allerdings befinden sich hier die uv-Ebenen im Zentrum und die st-Ebenen im Unendlichen. Damit

ähnelt diese Anordnung einem Quick-Time VR-Panorama, allerdings kann sich der Betrachter hier noch begrenzt bewegen.
Das letzte Beispiel ist das Light Field eines Spielzeuglöwen, das aus digitalisierten Bildern gewonnen wurde. Es wurden vier Light Slabs mit der gleichen Anordnung wie in Abbildung 1.2 verwendet, so dass sich der Löwe komplett um die eigene Achse drehen lässt. Der Aufnahmesensor ermöglichte eine korrekte Pixelfilterung, doch die Blendenöffnung war zu klein für eine vollständige Blendenfilterung. Aus diesem Grund zeigen sich im Light Field einige Aliasingeffekte, die am stärksten nahe des Kopfes und des Schwanzes auftreten, da diese Regionen am weitesten von der Rotationsachse entfernt sind.

Literaturverzeichnis

[Adelson E.H. '91] Bergen J.R. Adelson E.H. The Plenoptic Function and the Elements of Early Vision. In *Computation Models of Visual Processing*. MIT Press, 1991.

[Marc Levoy '96] Pat Hanrahan Marc Levoy. Light Field Rendering. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '96)*, Seiten 31–42, 1996.

2 Lumigraph

Michaela Karcher

2.1 Einleitung

Im folgenden wird der Inhalt des Papers "The Lumigraph" [Steveb J. Gortler '96] von Steven J. Gortler (Microsoft Research), Radek Grzeszczuk (University of Toronto), Richard Szeliski (Microsoft Research) und Michael F. Cohen (Microsoft Research) diskutiert.
Der Lumigraph ist eine Methode neue Sichten aus beliebigen Kamerapositionen ohne Verwendung von Tiefeninformationen oder sonstigen Parametern der Szene zu generieren. Dies geschieht nur durch Wiedergabe und Kombination der vorhandenen Bilder, die sowohl von realer als auch synthetischer Herkunft sein können. Lumigraph führt die Arbeit weiter, die mit Quicktime VR, das es dem Betrachter ermöglicht sich in einer Szene umzuschauen, und der Plenoptischen Modellierung begann. Es entwickelt die Idee weiter, den kompletten Lichtfluss in einer Region der Umgebung einzufangen. Somit ist es auch eine Weiterentwicklung des Light Field. Der Unterschied besteht darin, dass die Kameraposition beliebig wählbar ist. Die plenoptische Funktion stellt den Lichtfluss an jedem Ort (x, y, z) in jede beliebige Richtung (Θ, Φ) dar. Der Lumigraph ist nun die reduzierte 4D Funktion dieser plenoptischen Funktion. Mittels Lumigraph können neue Bilder von Objekten ohne komplexe Geometrie oder komplexe Beleuchtungsmodelle sehr schnell generiert werden.

Von 5D nach 4D
Die Plenoptische Funktion ist also eine Funktion mit 5 Variablen, die den Ort und die Richtung darstellen. Wenn wir von transparenter Luft im freien Raum ausgehen, dann kann die Strahlung als konstant angenommen werden. Wenn wir weiterhin unser Interesse auf das Licht, das die konvexe Hülle eines gebundenen Objektes verlässt, beschränken, dann müssen wir nur den Wert der plenoptischen Funktion entlang einer das Objekt umschliessenden Oberfläche betrachten. Der Einfachheit halber wurde ein Würfel gewählt (siehe Abb. 2.1). Für jeden Strahl durch einen Gitterpunkt des Würfels kann durch Zurückverfolgen bis auf die Oberfläche des Objektes der Wert des Lumigraph bestimmt werden.

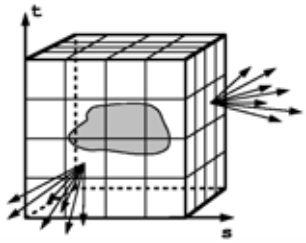


Abbildung 2.1: Umschliessen des Objektes mit einem Würfel

Parametrisierung
Die Parametrisierung basiert auf zwei konzentrischen Würfeln, die die Szene beinhalten. Im folgenden wird die Betrachtung auf ein Würfelseitenpaar beschränkt. Für beide Ebenen werden orthogonale Achsen festgelegt. Die äusseren Achsen werden s und t genannt (siehe Abb. 2.1), die inneren u und v (siehe Abb. 2.2). So kann jeder Strahl mit Hilfe der Druchstosspunkte der 2 Ebenen dargestellt werden. Der Strahl (s, t, u, v) ist also der Strahl, der die st-Ebene in (s, t) und die uv-Ebene in (u, v) durchstösst. Der Ursprung wird in der Mitte der uv-Ebene festgelegt mit einer z-Achse orthogonal dazu. Die st-Ebene soll sich im Abstand $z = 1$ dazu befinden. Der ganze Lumigraph besteht aus sechs solcher Ebenen-Paare mit Normalen entlang der Richtungen $x, -x, y, -y, z$ und $-z$. Es ist oft einfacher den Lumigraph mittels zwei 2D Analogien dazustellen (siehe Abb. 2.2 (b)).

2.2 Sampling

Abb. 2.3 zeigt ein Blockdiagramm des Lumigraph Systems. Der Prozess beginnt mit der Bildaufnahme mit einer Handkamera. Die Kameraposition und -orientierung wird mit Hilfe bekannter Markierungen abgeschätzt. Daraus wird ein einfach approximiertes geometrisches Modell des Objektes erstellt, das zur Tiefenkorrektur der (u, v)-Werte verwendet werden soll. Aus den Pixeln der Bilder werden die Koeffizienten des diskreten Lumigraph abgeschätzt. Alternativ können die Bilder zur Erstellung des Lumigraph auch von synthetischer Herkunft sein.

Bilder aus synthetischen Szenen
Die Erstellung des Lumigraph aus synthetischen Bildern ist recht einfach. Dabei wird von jedem Punkt (i,j) in der st-Ebene mittels einer virtuellen Kamera ein Bild von dem Objekt gemacht. Die uv-Ebene wird als Bildebene genutzt. Der Farbwert des Pixels (p, q) ist dabei der Koeffizient $x_{i,j,p,q}$ des Lumigraph. Die Qualität kann verbessert werden, indem man mehrere Strahlen durch die gejitterte Kamera legt und die Koeffizienten gemittelt werden.

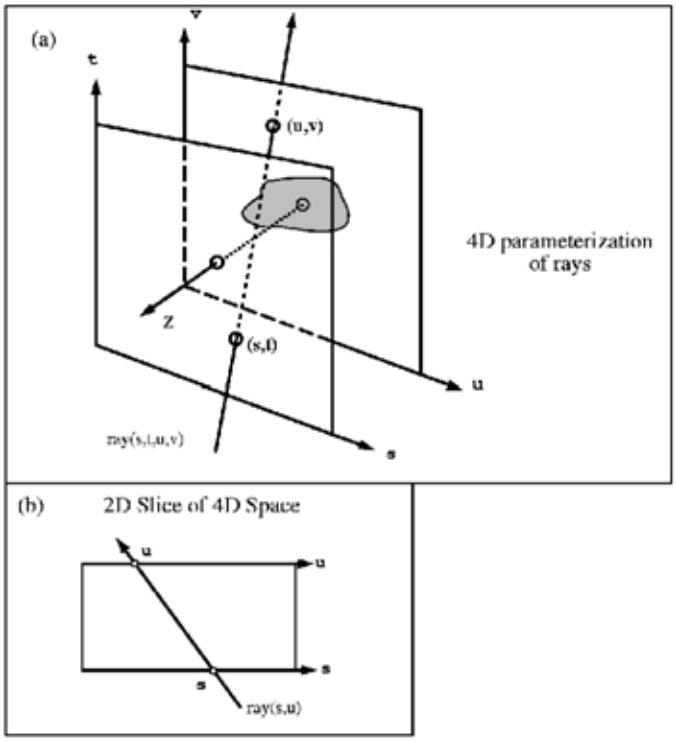


Abbildung 2.2: Parametrisierung des Lumigraph

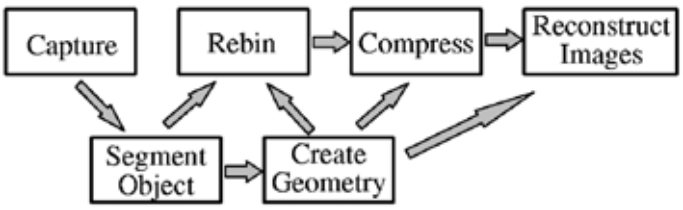


Abbildung 2.3: Das Lumigraph System

Bilder aus realen Szenen
Hierbei müssen viele Bilder des Objektes gemacht werden. Einerseits kann das wie zuvor beschrieben verwirklicht werden, was aber sehr aufwendig und teuer ist. Eine andere Methode ist, die Szene mit einer Handkamera aufzunehmen. Dazu muss man zuerst die Kamera kalibrieren, um die Projektion zwischen den Richtungen und den Bildkoordinaten ermitteln zu können. Als nächstes muss man die Szene mit speziellen Markierungen versehen, damit die Position der Kamera davon abgeleitet werden kann. Um es einfacher zu machen, wäre ein grobes geometrische Modell des Objektes wünschenswert. Das verwirklicht man, indem jedes Bild mit Hilfe der Bluescreen-Technik in einen Umriss umgewandelt wird, um dann daraus ein Volumenmodell zu erstellen.

Kamerakalibrierung und Lageschätzung
Das sind zwei Teile eines Prozesses: Ermittlung einer Projektion zwischen den Bildschirmpixeln und den Strahlen in der Szene. Die Parameter dieses Prozesses werden aufgeteilt in externe und interne Parameter. Die externen Parameter definieren die Kameraposition, die internen definieren die Festlegung der 3D Kamerakoordinaten auf dem Bildschirm. Um die internen Parameter konstant zu halten, damit sie nur zu Beginn geschätzt werden müssen, wird eine Kamera mit fester Linse benutzt. Die externen Parameter ändern sich konstant und müssen für jeden Videoframe neu berechnet werden. Schliesslich kann man dies mit viel weniger Kalibrierungspunkten machen, wenn man die internen Parameter gegeben hat.

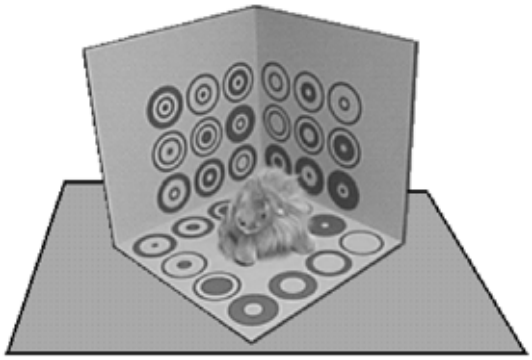


Abbildung 2.4: Erfassungsbühne

Eine speziell designte Bühne hilft bei der Kalibrierung (siehe Abb. 2.4). Die Bühne hat zwei Wände im rechten Winkel zueinander und eine Basis, die von den Wänden losgemacht und in 90 Grad Winkeln gedreht werden kann. Ein Objekt, das auf solch einer beweglichen Basis plaziert wird, kann von allen Richtungen gezeigt werden. Der Bühnenhintergrund ist cyan für spätere Bluescreen-Prozeduren. Dreissig Markierungen, jede aus verschiedenen konzentrischen Ringen in einem dunkleren Ton als cyan, sind über die Bühne und die Basis verteilt. Damit kann man sehr genau die Kamera kalibrieren. Während den externen Kamerakalibrierungen müssen nur 8 oder mehr Markierungen sichtbar sein, um eine Position zuverlässig zu berechnen.

3D Oberflächen Approximation
Eine grobe Abschätzung der Oberfläche des Objektes erhält man mit einem Octree-Konstruktionsalgorithmus. Dazu wird zuerst jedes Bild mit Hilfe der Bluescreen-Technik in ein binäres Objekt/Hintergrund-Bild segmentiert. Dann wird die Octree-Repräsentation eines Würfels, der das ganze Objekt beeinhaltet, initialisiert. Falls jetzt ein Voxel des Octrees auf ausserhalb des Objektumrisses fällt, so wird er entfernt. Wenn er auf den Rand des Umrisses fällt, wird es für eine weitere Unterteilung in acht kleinere Würfel vorgemerkt. Nachdem dies für

einige Bilder gemacht wurde, werden die markierten Voxel weiter unterteilt. Nach zirka 4 Unterteilungsschritten entsteht ein genügend gutes 3D-Modell des Objektes, bestehend aus einer Ansammlung von Voxeln. Die externen Polygone werden zusammengefasst und bilden die Oberfläche, die bei Bedarf noch geglättet werden kann.

2.3 Lumigraph Datenstruktur

Diskretisierung

Um den Lumigraph konkret implementieren zu können, müssen die st- und die uv-Ebene diskretisiert werden. Dazu sucht man eine diskrete Unterteilung jeder der Ebenen. Die st-Ebene soll dabei in $M \times M$ Quadrate unterteilt werden, die uv-Ebene in $N \times N$ (siehe Abb. 2.5). Die Gitterpunkte auf der st-Ebene werden mit (i, j) indiziert, die auf der uv-Ebene mit (p, q). Ein Strahl wird also als 4D Punkt (i, j, p, q) dargestellt. Der Wert des Lumigraph (genauer das RGB-Tripel) an diesem Gitterpunkt wird als $x_{i,j,p,q}$ bezeichnet.

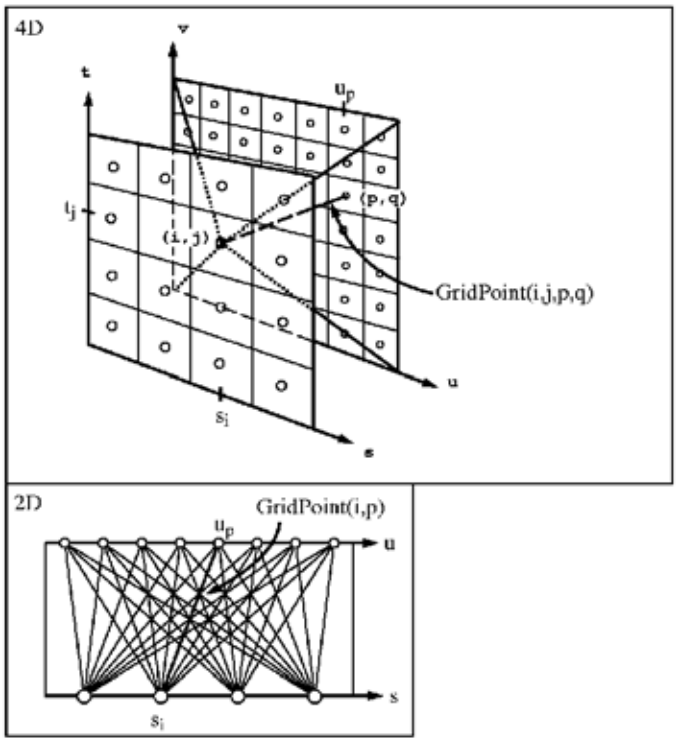


Abbildung 2.5: Diskretisierung des Lumigraph

Wahl der Basis

Es wird zu jedem Gitterpunkt eine Basisfunktion $B_{i,j,p,q}$ assoziiert, so dass der kontinuierliche Lumigraph als folgende lineare Summe rekonstruiert wird $\tilde{L}(s,t,u,v) = \sum_{i=0}^{M} \sum_{j=0}^{M} \sum_{p=0}^{N} \sum_{q=0}^{N} x_{i,j,p,q} B_{i,j,p,q}(s,t,u,v)$.

$\tilde{L}$ ist dabei der endliche Lumigraph, der im Raum durch die Basis B definiert ist. Als Basen kann dabei verschiedenes gewählt werden. Gebräuchlich sind die konstante Basisfunktion mit dem Wert 1 um den Gitterpunkt und 0 sonst, oder die quadralineare Basis mit dem Wert 1 am Gitterpunkt und abfallend gegen 0 zu den Nachbarpunkten.

Projektion in die gewählte Basis
Bei einem gegebenen kontinuierlichen Lumigraph L und der Wahl einer Basis für den endlichen Lumigraph $\tilde{L}$ benötigt man noch eine Projektion von L auf $\tilde{L}$, um die Koeffizienten x zu bestimmen. Die Projektion ist durch das innere Produkt
$x_{i,j,p,q} =< L, \tilde{B}_{i,j,p,q} >$
gegeben.

Auflösung
Eine wichtige Entscheidung betrifft die Auflösung, M und N. Die Wahl von M und N wird von der Vorstellung beeinflusst, dass die uv-Ebene näher am Objekt liegt als die st-Ebene. Deshalb liegt die Auflösung N sehr nahe an der Auflösung des gewünschten Bildes. Entsprechend bekommt die uv-Ebene die Auflösung des Bildes (zirka zwischen 128 und 512). Da eine Änderung des Gitterpunktes in der st-Ebene nur einer kleinen Änderung im Objekt entspricht, kann hier die Auflösung kleiner gewählt werden (zirka zwischen 16 und 64) (siehe Abb. 2.6 (b)).

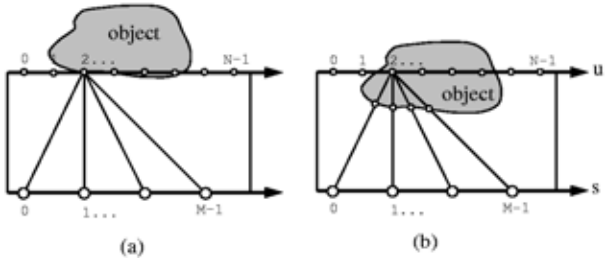


Abbildung 2.6: Wahl der Auflösung

Gebrauch geometrischer Information

Gegeben sei der Strahl (s, u) im 2D Lumigraph (siehe Abb. 2.7). Der am nächstgelegene Gitterpunkt zu diesem Strahl ist (s_{i+1}, u_p). Trotzdem könnten die Gitterpunkte (s_{i+1}, u_{p-1}) und (s_i, u_{p+1}) einen Wert haben, der näher am Wert von (s, u) liegt, weil diese Strahlen das Objekt nahe dem Durchstosspunkt von (s, u) schneiden. Wenn wir nun die Tiefe z wissen, bei der der Strahl (s, u) die Oberfläche des Objektes das erste mal durchstösst, dann kann man für ein gegebenes s_i ein u' finden, so dass der Strahl (s_i, u') die Oberfläche an der selben Stelle durchstösst wie der Originalstrahl. Sei nun $z = 0$ bei uv und $z = 1$ bei st, dann kann u' wie folgt berechnet werden:

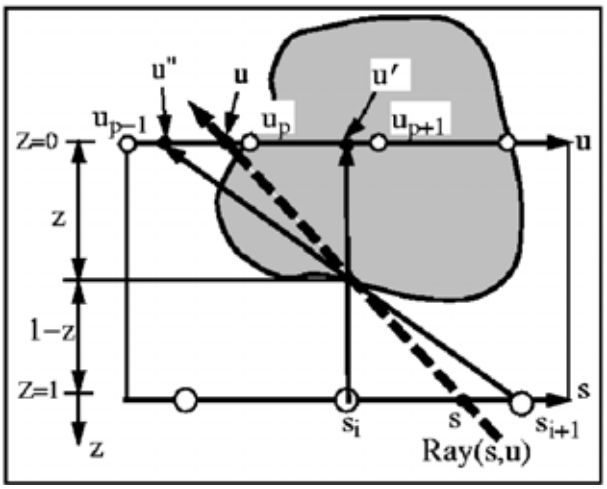


Abbildung 2.7: Tiefenkorrektur der Strahlen

$u' = u + (s - s_i)\frac{z}{1-z}$
Wendet man diese Tiefenkorrektur auf u und v an, so wird der Lumigraph überarbeitet. Die neue Basisfunktion $B'_{i,j,p,q}(s,t,u,v)$ wird nun durch vorheriges Finden von u' und v' und anschliessendem Auswerten von B wie folgt definiert:
$B'_{i,j,p,q}(s,t,u,v) = B_{i,j,p,q}(s,t,u',v')$
Obwohl die tiefenkorrigierte Basis jetzt komplizierter ist, ist $\tilde{L}(s,t,u,v)$ immernoch eine lineare Summe von Koeffizienten, gewichtet mit der entsprechenden Basis.
In der folgenden Berechnung wird der Wert von $\tilde{L}(s,t,u,v)$ durch eine quadralineare Basisfunktion korrigiert:

```
QuadralinearDepthCorrect(s,t,u,v,z)
    Result = 0
    h_st = s_1 − s_0 /* grid spacing */
    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')
            iterpWeight_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
```

2.4 Rekonstruktion/Rebinning

Der Koeffizient, der im Zusammenhang mit der Basis $B_{i,j,p,q}$ steht, wird wie folgt definiert:
$x_{i,j,p,q} = \int L(s,t,u,v)\tilde{B}_{i,j,p,q}(s,t,u,v)dsdtdudv$
Beim Erstellen des Lumigraph mit einer Handkamera kann es zu zwei Problemen kommen: Einerseits kann das Abtastergebnis grosse Löcher enthalten, andererseits ist die Abtastdichte typischerweise nicht gleichverteilt.
Für das erste Problem gibt es einen Algorithmus von [Burt88]. Dort wird eine Bilderpyramide benutzt, um ein hierarchisches Set von Bildern kleinerer Auflösungen zu erzeugen. Jedes dieser Bilder kleinerer Auflösung repräsentiert das verschwommene Abbild des Eingabebildes. In den kleineren Auflösungen werden die Löcher auch kleiner. Die Bilder mit tieferer Auflösung werden dann dazu verwendet die Löcher der Bilder höherer Auflösung zu füllen.
Das zweite Problem löst ein Algorithmus von [Mitchell87]. Dieser Algorithmus vermeidet, dass Bereiche mit höherer Abtastdichte übergewichtet werden. Dazu werden in Bereichen kleinerer Regionen Mittelwerte errechnet. Der endgültige Pixelwert wird dann durch Mittelung der Werte dieser Region berechnet. Der Durchschnitt ist nicht mit der Anzahl der Samples gewichtet, die in diese Region fallen, und beeinflusst deshalb nicht das Endergebnis.
Der Algorithmus zum Erstellen des Lumigraph beeinhaltet diese Ideen. Der Einfachheit halber wird der Algorithmus in 1D beschrieben und benutzt nur den Index i. Die höchste Auflösung wird mit 0 versehen, geringere mit entsprechend höheren Indizes. Zu jedem Koeffizient x_i^r (r ist dabei die Auflösung) gibt es eine Gewichtung w_i^r. Diese Gewichte legen fest, wie die Koeffizienten unterschiedlicher Auflösungen eventuell kombiniert werden.
Der Algorithmus läuft in drei Phasen ab:

- Splatting-Phase: Hier werden die Samples dazu verwendet das Integral zu Berechnung der Koeffizienten x_i^0 und Gewichte w_i^0 zu berechnen. In Regionen, in denen es wenig oder keine Samples gibt, sind die Gewichte klein oder Null. Die Koeffizienten werden durch Monte-Carlo Integration mit folgenden Gewichtungen berechnet:
  $w_i^0 = \sum_k \tilde{B}_i(s_k)$
  $x_i^0 = \frac{1}{w_i^0} \sum_k \tilde{B}_i(s_k)L(s_k)$
  Dieser Schritt läuft linear zur Anzahl der Samples ab.

- Pull-Phase: Hier werden die Bilder niedrigerer Auflösung berechnet, in denen die Löcher kleiner werden (siehe Abb. 2.8)

  Die Koeffizienten kleinerer Auflösungen werden berechnet, indem die Koeffizienten höherer Auflösungen mit Hilfe von $\tilde{h}$, einer diskreten Sequenz, kombiniert werden. Ein Weg das zu tun wäre:
  $w_i^{r+1} = \sum_k \tilde{h}_{k-2i} w_k^r$

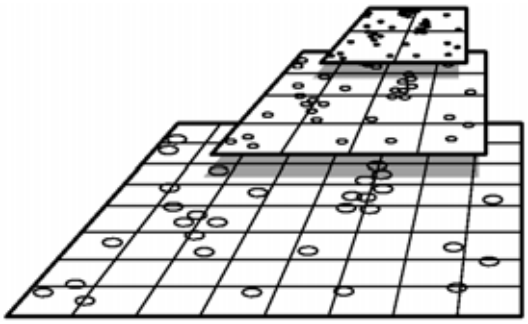


Abbildung 2.8: In kleineren Auflösungen sind die Löcher kleiner

$x_i^{r+1} = \frac{1}{w_i^{r+1}} \sum_k \tilde{h}_{k-2i} w_k^r x_k^r$
Dieser Schritt läuft linear zur Anzahl der Basisfunktionen summiert über alle Auflösungen. Weil jede kleinere Auflösung die halbe Basisfunktionsdichte hat, läuft dieser Schritt linear zur Anzahl der Basisfunktionen der Auflösung 0 ab.

- Push-Phase: Hier werden die Informationen der Bilder niedrigerer Auflösung mit denen der höheren Auflösung kombiniert, indem die Löcher aufgefüllt werden. Wenn ein Koeffizient höherer Auflösung ein grösseres Gewicht als 1 hat, ignorieren wir die Information geringerer Auflösungen an dieser Stelle. Wenn das Gewicht dort aber zu klein ist, so blendet man dort die Information der geringeren Auflösung ein. Um das zu tun, muss die Funktion aus der tieferen Auflösung in die Basis der höheren ausgedrückt werden. Dies geschieht mit den temporären Variablen:
$tw_i^r = \sum_k h_{i-2k} min(w_k^{r+1}, 1)$
$tx_i^r = \frac{1}{tw_i^r} \sum_k h_{i-2k} min(w_k^{r+1}, 1) x_k^{r+1}$
Diese Variablen können jetzt mit den Werten x und w der Auflösung r überblendet werden.
$x_i^r = tx_i^r(1 - w_i^r) + w_i^r x_i^r$
$w_i^r = tw_i^r(1 - w_i^r) + w_i^r$

Diese drei Phasen müssen leicht angepasst werden, wenn man Tiefenkorrektur anwendet. Während der Splatting-Phase werden dann in jedem Strahl $L(s_k, t_k, u_k, v_k)$ die u und v Werte angepasst. Während der Pull- und der Push-Phase benutzt man, anstatt einfach die Koeffizienten mit der Basis und benachbarten Indizes zu kombinieren, die tiefenkorrigierten Indizes.

2.5 Rekonstruktion von Bildern

Gegeben durch eine gewünschte Kamera (Position, Orientierung, Auflösung) färbt die Rekonstruktionsphase jedes Pixel des Outputbildes mit der Farbe, die diese

Kamera erzeugen würde, wenn sie auf das reale Objekt zeigen würde. Die Rekonstruktion der Bilder kann durch Ray Tracing geschehen: Ist ein Lumigraph gegeben kann man ein neues Bild von einer beliebigen Kamera Pixel für Pixel und Strahl für Strahl generieren. Für jeden Strahl werden die entsprechenden Koordinaten (s, t, u, v) berechnet, die nächsten Gitterpunkte ermittelt und ihre Werte eventuell interpoliert. Bei der tiefenkorrigierten Basis werden vor der Interpolation noch die korrigierten Koordinaten (u', v') berechnet. Dies kann mit Hilfe von Harware realisiert werden.
Die Kosten für das Behandeln jedes einzelnen Pixels kann vermieden werden, wenn man Texture Mapping Operationen benutzt.

2.6 Komprimierung/Resultate

Komprimierung
Für ein typisches Beispiel mit einem 32×32 Sampling in der st-Ebene und einem 256×256 Sampling in der uv-Ebene benötigt man beim Speichern aller sechs Seiten des Würfels mit 24-bit pro Pixel $32^2 \cdot 256^2 \cdot 6 \cdot 3 = 1.125$GByte. Auf Grund von sehr viel Redundanz in den Bildern wird eine Kompressionsrate von bis zu 200:1 erwartet ohne grossen Qualitätsverlust. Dies würde die Grösse dieses Lumigraphs auf unter 6MByte reduzieren.

Ergebnisse
Für synthetische Objekte kann der Lumigraph also von polygon-gerenderten Bildern oder solchen aus Ray Tracing erzeugt werden. Alle wichtigen Bilder zu berechnen kann Wochen dauern. Für reale Objekte kann man eine einzelne billige analoge Videokamera verwenden. Dann dauert die Capture-Phase nur eine Stunde oder weniger. Die Verarbeitung auf einer SGI Indy Workstation dauert weniger als einen Tag. Ist ein Lumigraph einmal vorhanden, können neue Bilder mit einer Auflösung von 450×450 mittels Ray Tracing in wenigen Sekunden generiert werden. Mittels Texture Mapping konnten sogar mehrere Bilder in wenigen Sekunden generiert werden.

Literaturverzeichnis

[Steveb J. Gortler '96] Michael F. Cohen Steveb J. Gortler, Radek Grzesczuk; Richard Szeliski. The Lumigraph. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '96)*, Seiten 43–54, 1996.

3 Hardware Light Field Rendering

Richard Hoffmann

Die Verfahren Light Field Rendering und Lumigraph beanspruchen bei direkter Umsetzung hohe Ressourcen. In diesem Kapitel sollen verschiedene Optimierungsmöglichkeiten untersucht werden, die unter anderem direkten Gebrauch von moderner 3D-Hardware einschließt.

3.1 Einführung

Der Lumigraph basiert wie alle Image Based Rendering-Verfahren auf der Plenoptischen Funktion. Die Plenoptische Funktion beschreibt die Position und Richtung aller Lichtstrahlen in einer dreidimensionalen Szene und ist damit eine fünfdimensionale Funktion. Ein Rendering-Verfahren, das direkt auf dieser Funktion aufsetzt, würde allerdings die zur Verfügung stehenden Speicherressourcen aktueller Rechner überfordern. Daher gilt es zunächst diese Funktion einzuschränken. Dies wird beim Lumigraph dadurch erreicht, das nur endliche große konvexe Körper betrachtet werden. Diese Körper können dann von einer Oberfläche eingeschlossen werden. Jeder Strahl in der eingeschlossenen Szene kann jetzt durch 2 Schnittpunkte auf der Oberfläche beschrieben werden. Da jeder Punkt auf einer Oberfläche durch seine (x,y)-Koordinaten bestimmt ist, reduziert sich die plenoptische Funktion auf 4 Parameter. Als begrenzende Oberfläche eignet sich eine Menge von parallelen Ebenen wie z.B. ein Würfel besonders gut, da eine diskretisierte Ebene mit äquidistanten Punkten direkt auf eine Speicherseite im Rechner abgebildet werden kann. Zur weiteren Vereinfachung können nur 2 parallele Ebenen ohne Beschränkung der Allgemeinheit betrachtet werden (vgl. Abb. 3.1). Analog zu [M.F.Cohen '96] wird die erste Ebene durch die Koordinaten s und t, die zweite durch die Koordinaten u und v bestimmt. Jedem Punkt im eingeschlossenen Raum ist damit durch die Funktion L(s,t,u,v) ein Farbwert zugeordnet, der den Lichtstrahl durch diesen Punkt widerspiegelt.

3.2 Diskretisierung

Zur Realisierung im Rechner muß die Oberfläche des Lumigraph diskretisiert werden. Im Falle der Ebenendarstellung ist es naheliegend die hintere uv-Ebene

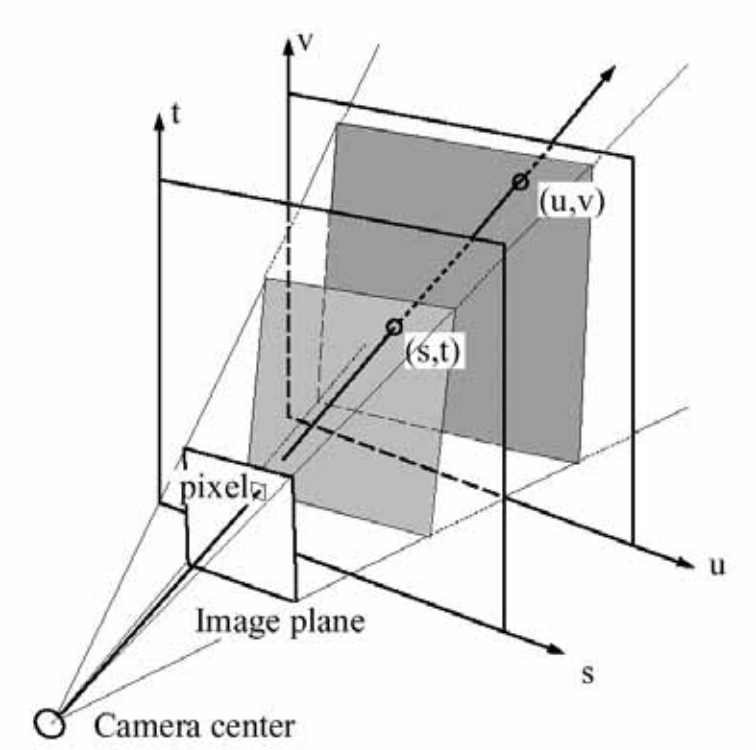


Abbildung 3.1: Parametrisierung des Lumigraph. Entnommen aus [M.F.Cohen '97]

feiner aufzulösen als die vordere st-Ebene. Als praktische Werte haben sich dabei 32x32 Knoten auf der st-Ebene und 256x256 Knoten auf der uv-Ebene herauskristallisiert. Sowohl für das Erstellen eines Lumigraph als auch die Rekonstruktion eines Bildes ist bedingt durch diese recht grobe Rasterung ein Interpolationsverfahren erforderlich. Dazu wird hier die quadralineare Basis wie in Gortler [M.F.Cohen '96] verwendet.

3.3 Rekonstruktion mit Texture Mapping

Die Rekonstruktion eines Bildes aus dem Lumigraph in Echtzeit ist aufgrund der notwendigen Interpolation pro Pixel rechenintensiv. Ein schneller Bildaufbau ist allerdings unter Verwendung der heute üblichen 3D-Beschleuniger möglich. Dabei wird die uv-Ebene als Textur aufgefasst und statt einzelner Pixel werden ganze Dreiecke gezeichnet (vgl. Abb. 3.2). Die Interpolation wird jetzt dadurch realisiert, dass die 3 Eckpunkte eines Dreiecks auf der st-Ebene, denen jeweils eine uv-Textur zugeordnet ist, dreimal mittels Alpha-Blending übereinander gezeichnet werden. Verwendet man für jedes dieser Dreiecke die Alpha-Werte 1.0 am Punkt der zugeordneten Textur und 0.0 an den beiden anderen Punkten, ergeben die drei übereinandergezeichneten Dreiecke wieder das Ursprungsbild. Wie

in Abb. 3.2 zu sehen ist ergeben sich die Texturkoordinaten aus dem Schnitt der Sichtgeraden mit der uv-Ebene. Die Texturkoordinaten sind für die drei Dreiecke identisch. Um ein komplettes Bild zu rekonstruieren müssen also lediglich die Dreiecke auf der st-Ebene je dreimal gezeichnet werden. Diese Aufgabe wird von aktuellen 3D-Beschleunigern leicht bewältigt. Ein Problem stellen allerdings die Speicheranforderungen dar: bei einem Würfel mit 6 Ebenen und RGB-Farbwerten liegt der Speicherbedarf dann bei 256x256x32x32x6x3 = 1.125 GB. Gängige 3D-Beschleuniger stellen aber nur 4-32MB Texturspeicher zur Verfügung. Die Datenmenge läßt sich allerdings durch verschiedene Kompressionsverfahren reduzieren, die im Verlauf dieser Arbeit besprochen werden. Zunächst werden die Möglichkeiten, die st-Ebene zu komprimieren, untersucht.

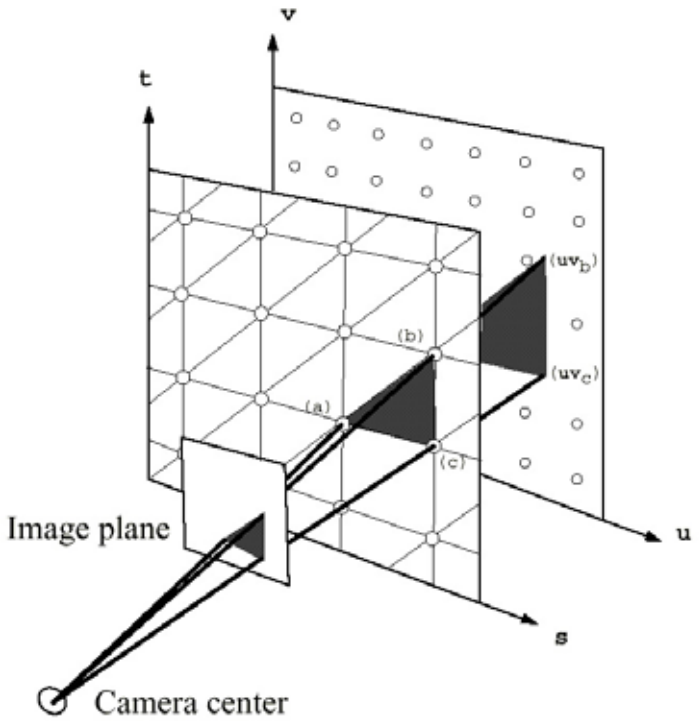


Abbildung 3.2: Rekonstruktion des Lumigraph mit Texture Mapping. Entnommen aus [M.F.Cohen '97]

3.4 Kompression der st-Ebene

3.4.1 Sub-sampling

Die einfachste Möglichkeit der Datenreduktion besteht in der Halbierung der Auflösung der st-Ebene, wodurch der Gesamtspeicherbedarf auf ein Viertel gesenkt wird. Als Folge ergibt sich eine größere Unschärfe, bedingt durch das Alpha-

Blending. Um den Verlust an Bildqualität zu kompensieren können Mipmaps verwendet werden [L.Williams '83].

3.4.2 Zentrumsverteilung

Bei den meisten Bildern ist das Zentrum des Bildes von besonderer Bedeutung, da der Betrachter diesem seine größte Aufmerksamkeit schenkt. Dieser Eigenschaft kann man dadurch Rechenschaft tragen, dass man nur die Knoten auf der st-Ebene im Zentrum verwendet und die Randknoten wegläßt. Exemplarisch werden im folgenden 9 Knoten im Zentrum untersucht (vgl. Abb. 3.3), wobei zunächst von einem unbewegtem Betrachter ausgegangen wird. Die Dreiecke innerhalb der Zentrumsknoten werden wie vorher verarbeitet, d.h dreifach mit den Alpha-Werten 1,0,0 übergeblendet. Um das Bild zum Rand hin zu vervollständigen, wählt man sich fiktive Knoten am Rand der st-Ebene und zeichnet alle übrigen Dreiecke, allerdings nur zweifach, da den fiktiven Randknoten gar keine Textur zugeordnet ist. Diese Vorgehensweise soll an folgendem Beispiel erläutert werden: In Abb. 3.3 wird zunächst Dreieck I mit der Textur, die Knoten (a) zugeordnet ist, gezeichnet. Die Alpha-Werte sind dabei 1 für Knoten (a), 0 für Knoten (b) und 0.5 für den fiktiven Randknoten (a,b). Dreieck I wird nun nochmals mit der Textur von Knoten (b) gezeichnet, und zwar analog mit den Alpha-Werten 1 für Knoten (b), 0 für Knoten (a) und wieder 0.5 für Knoten (a,b), wodurch das Überblenden der beiden Dreiecke wieder zum Originalbild führt, da die Summe der Alpha-Werte an allen Eckpunkten 1 ergibt. Etwas komplizierter wird es bei Dreieck II. Die Textur von Knoten (a) wird hier mit den Alpha-Werten 1 an Knoten (a), 0.5 an Knoten (a,b) und 1 am Eckknoten der st-Ebene gezeichnet. Da der Knoten (a,b) je zur Hälfte aus den Texturen von (a) und (b) besteht, wird das zweite Dreieck mit der Textur von Knoten (b) und dem Alpha-Wert 0.5 an Knoten (a,b) und 0 an den übrigen Knoten gezeichnet. Die Summe der Alpha-Werte ist dann auch hier wieder 1.

3.4.3 Flexible Verteilung

Die Zentrumsverteilung kann dahingehend verallgemeinert werden, dass die st-Ebene aus beliebigen fiktiven Knoten und Knoten mit Texturdaten, im folgenden reale Knoten genannt, besteht. Einzige Einschränkung ist hier, dass jeder fiktive Knoten mindestens einen Nachbarknoten mit Texturdaten haben muss, da sonst an diesem Knoten kein Dreieck gezeichnet werden kann und damit ein Loch im Bild enstehen würde.
Der Algorithmus läßt sich folgendermaßen formulieren:

```
Für alle realen Knoten a
zeichne_realen_knoten(a)
```

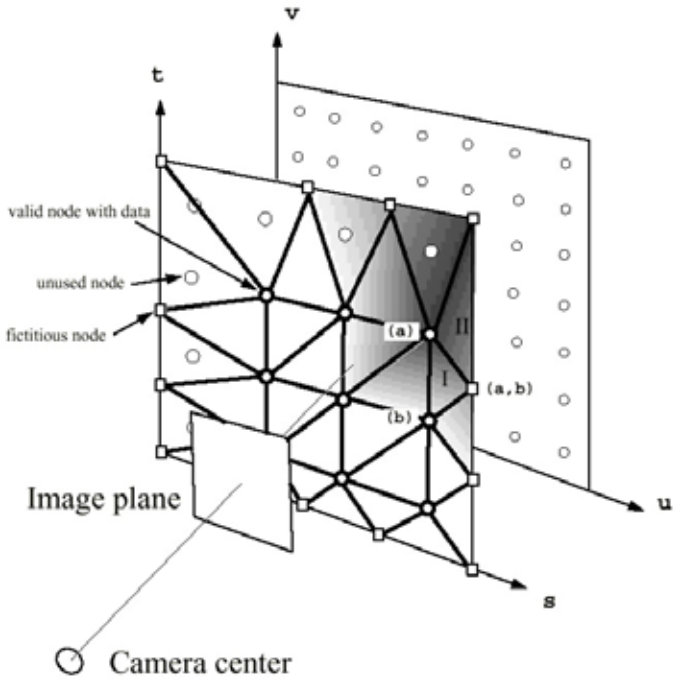


Abbildung 3.3: Zentrumsverteilung der st-Knoten. Entnommen aus [M.F.Cohen '97]

```
zeichne_realen_knoten(Knoten a)
{
setze alpha_a = 1.0
für alle angrenzenden knoten b
setze alpha_b = 0.0

zeichne_alle_an_a_angrenzenden_Dreiecke()

für alle angrenzenden fiktiven Knoten b
alpha = 1.0 / anzahl_an_b_angrenzender_realer_knoten
zeichne_fiktiven_knoten(b, alpha)
}

zeichne_fiktiven_knoten(Knoten b, double alpha)
{
setze alpha_b = alpha;

für alle angrenzenden knoten c
setze alpha_c = 0.0
```

```
zeichne_alle_an_b_angrenzenden_Dreiecke()
}
```

3.4.4 Flexible Verteilung bei bewegtem Betrachter

Bisher wurde der Betrachter (bzw. die Kamera) als statisch angenommen. Bewegt sich aber die Kamera in Echtzeit, müssen Knoten auf der st-Ebene entfernt und hinzugefügt werden. Um Artefarkte beim Entfernen oder Hinzufügen der Knoten zu vermeiden, muss obiges Verfahren erweitert werden. Anstatt die Knoten streng in fiktive und reale Knoten zu unterteilen, wird jedem Knoten ein Attribut Gültigkeit innerhalb [0,1] zugewiesen, mit dessen Hilfe Knoten weich ein- und ausgeblendet werden können. Weiterhin kann mit diesem Verfahren das Bild progressiv verfeinert werden, indem je nach verfügbarer Rechen- und Speicherkapazität mehr Knoten eingeblendet werden können. Das modifizierte Verfahren ist im folgenden skizziert, wobei die Funktion `zeichne_fiktiven_knoten()` unverändert bleibt.

```
zeichne_realen_knoten(Knoten a)
{
alpha_a = Gültigkeit von Knoten a

Für alle angrenzenden Knoten b
setze alpha_b = 0.0

zeichne_alle_an_a_angrenzenden_Dreiecke()

Für alle angrenzenden Knoten b
falls gültigkeit_b < 1.0
alpha = (1.0 - gültigkeit_b) /
anzahl_an_b_angrenzender_realer_knoten
zeichne_fiktiven_knoten(b, alpha)
}
```

3.4.5 Knotenverteilung entlang einer Geraden

Eine weitere Möglichkeit zu einer günstigeren Umverteilung der st-Knoten besteht darin, die Knoten entlang der Bewegungsgeraden des Betrachters anzuordnen (vgl. Abb. 3.4). Dabei verwendet man jetzt Trapeze, deren linke und rechte Seite rechtwinklig zur Bewegungsgeraden verlaufen. Die Trapeze sind weiterhin so gewählt dass jede Seite genau einen realen Knoten enthält; damit wird jedes Trapez zweifach gezeichnet, analog zu den Dreiecksverfahren mit den Alpha-Werten

1 für die zwei jeweils realen Punkte und 0 für die beiden gegenüberliegenden Punkte.

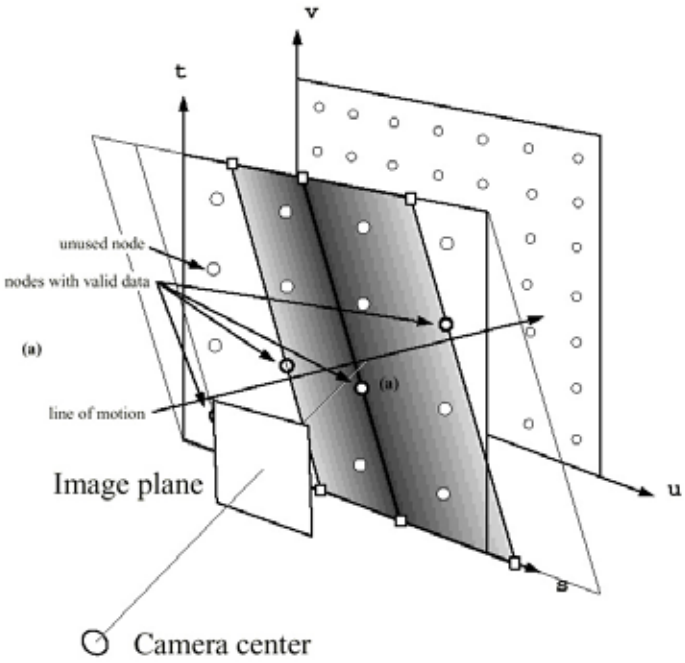


Abbildung 3.4: Knotenverteilung entlang einer Geraden. Entnommen aus [M.F.Cohen '97]

3.5 Texturprojektion

Die bisher beschriebenen Verfahren zur Bildrekonstruktion aus dem Lumigraph knnen bei schnellen Kamerabewegungen viel Rechenzeit beanspruchen, da Knoten ausgetauscht und neue Texturen nachgeladen werden müssen. Eine einfache Möglichkeit zur Reduktion des Rechenaufwands besteht darin, innerhalb einer Kamerabewegung einige Bilder nur anzunähern, etwa jedes zweite Frame. Dies kann durch Texturprojektion erreicht werden. Das zuletzt errechnete Bild wird dazu als Textur verwendet und mit Hilfe einer Matrix entlang der Bewegungsrichtung gedreht. Ist die Geometrie des betrachteten Objekts annähernd bekannt, kann diese Vorgehensweise verbessert werden. Anstatt nur eine Textur zu projizieren, wird das Ausgangsbild auf die angenäherte Geometrie rückprojiziert, entsprechend der Bewegungsrichtung verschoben und neu zusammengesetzt. Durch nderung der Sichtverhältnisse bei Kamerabewegungen erlaubt allerdings auch eine recht genaue Geometrie keine Interpolation über weite Strecken. Bei einer Rotation werden beispielsweise i.d.R. Artefarkte sichtbar, wenn die Geometrie um mehr als 10 gedreht werden muss.

3.6 Ergebnisse

Im folgenden sind die Ergebnisse der besprochenen Verfahren zusammengestellt (vgl. Tab.1). Die Füllrate gibt hierbei über die durchschnittliche Anzahl der zu zeichnenden texturierten Pixel, sog. Texel, Auskunft. Bei Gleichverteilung der st-Knoten werden z.B. immer 3 Dreiecke übereinandergezeichnet, wodurch 3 Texel pro Bildpixel gezeichnet werden müssen. Die Auslastungsschwankung bezieht sich auf den Texturspeicher. Bei Gleichverteilung der st- Knoten z.B. hängt die Auslastung stark von der Kamerabewegung ab, während die flexible Verteilung Belastungen mit Hilfe der teilgültigen Knoten ausgleichen kann. Mit Genauigkeit ist schließlich die Detailgenauigkeit innerhalb einer kleinen Umgebung angegeben. Bei der Zentrumsverteilung wird von 9 Knoten im Zentrum ausgegangen, bei der flexiblen Verteilung mit teilgültigen Knoten bezeichnet x den Prozentsatz an gültigen Knoten, der mindestens 25% betragen muss. Bei der Texturprojektion wird von der Annahme ausgegangen, dass etwa die Hälfte des zu rekonstruierenden Bildes grob durch Geometrie angenähert wurde.

Methode	Füllrate	Auslastungsschwankung	Genauigkeit
Gleichvert.	3	Hoch	Niedrig
Zentrum	2.25	Niedrig	Hoch
Flexibel (I)	3	Sehr niedrig	Mittel
Flexibel (II)	6x-3x+3	Sehr niedrig	Hoch
Gerade	2	Niedrig	Hoch
Projektion	1.5	-	Hoch

Tab.1 Ergebnisse

Literaturverzeichnis

[L.Williams '83] L.Williams. Pyramidal Parametrics. In *Computer Graphics Proceedings, vol.17*, Seiten 1–11, 1983.

[M.F.Cohen '96] S.J. Gortler; R.Grzeszczuk; R.Szeliski; M.F.Cohen. The Lumigraph. In *Computer Graphics Proceedings, Annual Conference Series*, Seiten 43–54, 1996.

[M.F.Cohen '97] S.J.Gortler; P.Sloan; M.F.Cohen. Time-Critical Lumigraph Rendering. In *Computer Graphics Proceedings, Annual Conference Series*, Seiten 2–3, 1997.

4 Quicktime VR

Erik Sohns

Dieses Kapitel widmet sich einem Software–Produkt der Firma Apple Inc. mit dem Namen QuickTime VR (QTVR). Es nimmt im Zusammenhang der in dieser Ausarbeitung besprochenen IBR–Ansätze insofern eine Sonderrolle ein, daß es sich hier eigentlich um ein kommerzielles Produkt, nicht um ein spezielles Verfahren handelt. Dieser Aspekt wurde berücksichtigt und spiegelt sich in der Gliederung dieses Kapitels wieder.
Die Software wurde unter Anwendung von IBR–Verfahren geschrieben und implementiert einige zur Thematik relevante Ansätze, auf die im Laufe des Kapitels eingegangen wird.
Informationen zu dieser Ausarbeitung wurden verschiedenen Quellen entnommen, besonders hilfreich waren dabei [Chen '95] und [Kitchens '98].

Der erste Abschnitt gibt eine Einführung zu Hintergrund und den grundlegenden Ideen. Im weiteren Verlauf werden die Ansätze beschrieben, die diese Ideen implementieren. Der letzte Abschnitt behandelt die Umsetzung dieser Ansätze und gibt einen Überblick zu den einzelnen Komponenten, aus denen QTVR besteht.

4.1 Allgemeines und Hintergrund

QTVR ist im wesentlichen eine Erweiterung der bestehenden QuickTime–Technologie (QT), die ebenfalls von Apple Inc. entwickelt wird. Sie wurde in die Version 3.0 integriert und ist seitdem fester Bestandteil der Software. Die aktuelle QTVR Version ist 2.1 und auf diese bezieht sich auch diese Ausarbeitung.
QTVR wurde erstmals auf einer Fachmesse in Cannes 1994 vorgestellt. Dem folgte dann im Juni 1995 die Erscheinung der Version 1.0. Ab QT 3.0 verfügt die Software über eine plattformübergreifende API (Windows/Macintosh), welches das Arbeiten mit und die Integration von QTVR–Inhalten in Applikationen vereinfachen soll.
Aufgrund der angesprochenen Tatsache, daß es sich hier sozusagen um ein eingebettetes Modul handelt, macht es Sinn, zunächst mit einigen einführenden Worten die überliegende Technologie zu erläutern.

4.1.1 QuickTime

QuickTime ist eine Multimedia–Systemsoftware und bietet ein Rahmenwerk für multimediale Inhalte verschiedenster Art. Dazu gehören digitales Video, Animationen, Sprites, digitales Audio, MIDI, Text und Einzelbilder. QT definiert eine einheitliche Schnittstelle um diese Elemente einerseits abspielen-, aber auch exportieren (z.B. in andere Applikationen einbinden) zu können.
Ein wichtiges Merkmal von QT–Inhalten ist deren einheitliche Organisation in lineare Spuren (sog. *tracks*). Der Zugriff erfolgt über einen gemeinsamen Zeitindex, sodaß sich auch mehrere Spuren synchron abspielen lassen (z.B. Audio und Video, siehe Abb. 4.1).
QuickTime ist erweiterungsfähig und relativ flexibel, so bietet es z.B. die Möglichkeit, neue Spurtypen zu definieren, alternative Abspielsoftware zu verwenden und verschiedene CODECs einzusetzen (Ein CODEC ist eine in Software geschriebene Kodier–/Dekodiereinheit für Multimedia–Elemente, siehe in diesem Zusammenhang 4.3.1).

Abbildung 4.1: QuickTime–Spuren

4.1.2 Grundidee von QTVR

Die Idee hinter QTVR ist im Namen versteckt — der Versuch, *virtual reality* (VR) in QuickTime zu integrieren. Dabei will man dem Benutzer die Möglichkeit geben, sich in einer virtuellen Umgebung umsehen– und bewegen zu können. Anspruch ist es, diese Erfahrung möglichst realistisch zu gestalten. Die wesentlichen Designaspekte sind also:

- Effizienz von Interaktion und Darstellung. Das System soll so schnell arbeiten, daß keine nennenswerten Wartezeiten entstehen, welche das interaktive Erlebnis einer virtuellen Umgebung beeinträchtigen.

- Hardwareunabhängigkeit der Software. Geringe Systemanforderungen sollen VR–Technologie "nach Hause" bringen. Spezielle Hardware–Beschleunigung werden weder gefordert noch grundsätzlich unterstützt. Das gilt auch für Ein– und Ausgabegeräte wie z.B. Datenhandschuhe und Head–Mounted Displays.

- Realismus der Szenarien. Unterstützt werden sowohl reale als auch synthetisch generierte Umgebungen. Reale Szenen besitzen normalerweise eine

große Komplexität aufgrund der hohen Detailvielfalt. Da sie nur schwer modellierbar sind, ist ihre Handhabung und Erstellung im Computer schwierig. Diesen Aufwand will man möglichst vermeiden, ohne auf Realismus zu verzichten.

- Hochauflösende Darstellung am Bildschirm. Viele VR–Systeme gehen bei der Darstellung einen Kompromiß zwischen Auflösung und Szenenkomplexität ein, um den Echtzeitanforderung gerecht zu werden. Hier soll aber die Darstellung unabhängig von der Bildqualität und Komplexität der Umgebung sein.

Aufgrund der Anforderungen ist in diesem Zusammenhang der Einsatz von IBR–Verfahren besonders interessant. Insgesamt will man virtuelle Umgebungen erzeugen, die so real sind, daß der Benutzer darin "eintaucht". Eine solche Technik bezeichnet man als *immersiv*, in diesem Spezialfall spricht man deshalb auch von *immersive Imaging*.

4.1.3 Ansatz: Virtuelle Kamera

Grundsätzlich wird also eine virtuelle Kamera in einer realen oder synthetischen Umgebung (man spricht von einer *Szene*) simuliert. Die Kamera repräsentiert das menschliche Auge, da das, was auf dem Bildschirm zu sehen ist, auch so vom Benutzer wahrgenommen wird. Um ein immersives Umfeld zu erzeugen muss der

Abbildung 4.2: Virtuelle Kamera

Benutzer in der Lage sein, mit seiner Umgebung zu interagieren. So berücksichtigt QTVR z.B. die freie Wahl der Blickrichtung, die Änderung des Standorts sowie Interaktion mit Objekten in der Szene. Da die dargestellte Perspektive nur von der Kameraposition und -ausrichtung abhängt, läßt sich Bewegung folgendermaßen klassifizieren:

- Rotation einer statischen Kamera (*camera rotation*)

- Rotation einer Kamera um ein Objekt (*object rotation*)
- Freie Kamerabewegung (*camera movement*)

Zusätzlich hat der Benutzer die Möglichkeit, einen Szenenausschnitt heranzuholen oder auszublenden, um entweder mehr Details oder einen grösseren Ausschnitt zu sehen. Dieses allgemein als *zooming* bekannte Verfahren fällt in eine spezielle Klasse *camera zooming*.

4.2 Freiheitsgrade in einer virtuellen Umgebung

Im folgenden wird nun auf die genannten Bewegungs–Klassen im einzelnen eingegangen. Insbesondere sollen hier die Umsetzungsproblematik und Lösungsansätze aufgezeigt werden, die das Softwaredesign reflektieren.

4.2.1 Camera Rotation

Geht man von einer festen Kamerastandpunkt aus, lassen sich die möglichen Kamerabewegungen mit drei Parametern beschreiben, es gibt drei Freiheitsgrade. Die Drehung der Kamera um eine virtuelle vertikale Achse durch das optische Zentrum (Brennpunkt der Kamerlinse) nennt man *yaw* oder *pan*, die Drehung um eine horizontale Achse *pitch* oder *tilt*. Zusätzlich läßt sich die Kamera um eine Achse in Blickrichtung drehen (*roll*).

Umsetzung

Bei der Umsetzung dieser Art von Kamerabewegungen in Software hilft die Erkenntnis, daß sich alle so möglichen Ansichten aus einer kanonischen Projektion errechnen lassen. Eine solche Projektion nennt man ein *environment map*, man kann sie sich als orientierungsunabhängige Ansicht einer Szene vorstellen.

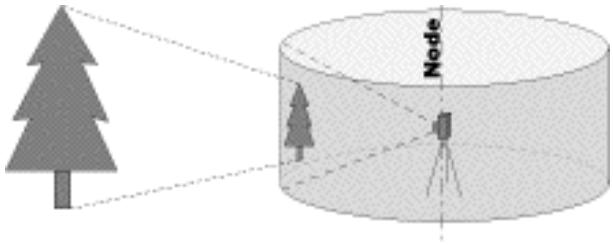


Abbildung 4.3: Zylindrische Evironment–Map

Der verfolgte Ansatz ist hier, in Abhängigkeit der spezifischen Parameter (diese entsprechen der momentanen Kameraorientierung), aus einer solchen Projektion die gewünschte Ansicht zu extrahieren. Dies geschieht durch eine Projektion der

kanonischen Projektion auf eine Bildfläche, dieses Verfahren nennt man *Reprojektion.*

Zylindrische environment maps

Die Reprojektion zur Erzeugung einer spezifischen Ansicht hängt von der Art des environment maps (z.B. kubisch, sphärisch) ab. Verzichtet man auf eine komplette 360 Rundumsicht, kommen zusätzlich noch andere Varianten in Frage. Eine zylindrische Form erlaubt z.B. eine vollständige horizontale Drehung, die vertikale Drehung ist dabei aber auf < 180 beschränkt.
Die mathematische Berechnung und die Aquisition einer zylindrischen Projektion ist gegenüber einer sphärischen wesentlich einfacher, weshalb man sich hier für eine solche entschieden hat.

Projektion auf einen Zylinder

Bei einer solchen Projektion ist für jeden Punkt p' auf der Projektionsfläche Π, der Zylinder, ein Halbstrahl g vom Projektionszentrum C durch p' definiert. Dieser Projektionsstrahl entspricht einem Lichtstrahl in der Szene, und es gilt allgemein, dessen Farbe zu berechnen. So gesehen ist die Projektion also eine Auswertung der *plenoptischen Funktion* auf dem Zylinder.
Dieser, hier senkrecht auf der xz–Ebene, wird durch zwei Parameter definiert, $u \in [0, 2\pi)$ und v (entspricht der Höhe des Zylinders) und einer Konstante δ, der minimalen Entfernung vom Projektionszentrum:

$$Z(u, v) = \begin{pmatrix} \delta \sin(u) \\ v \\ \delta \cos(u) \end{pmatrix} + C$$

Bei gegebenem Szenepunkt P ist der Halbstrahl g gegeben durch:

$$g : X = C + \mu(P - C), \mu \geq 0$$

Berechnet man den Schnitt des Zylinders mit den gegebenen Halbstrahlen für alle relevanten P, erhält man die gesuchten Punkte p'.

Reprojektion auf die Bildebene

Die Reprojektion zur Darstellung funktioniert dann ähnlich; für alle relevanten Punkte P auf dem Zylinder kann man Projektionsstrahlen g durch C definieren, die man nun mit der Bildebene schneidet (man kann sie sich z.B. als tangential an den Zylinder angeschmiegte Ebene vorstellen).

Ein Projektionsstrahl g vom Projektionszentrum durch einen Punkt $p' = (u, v, 1)^T$ auf dem Zylinder ist gegeben durch:

$$g : X = C + \lambda \begin{pmatrix} \delta \sin(u) \\ v \\ \delta \cos(u) \end{pmatrix}, \lambda \geq 0$$

4.2.2 Object Rotation

Die Drehung der Kamera um ein Objekt läßt sich mit dem eben beschriebenen Prinzip nicht umsetzen. Der hier verfolgte Ansatz ist der, daß man sich zunächst eine virtuelle Hüllkugel um das Objekt vorstellt. Die Kamera bewegt sich auf der Kugel, wobei das zentrale Objekt stets anvisiert bleibt. Der Bewegungsspielraum

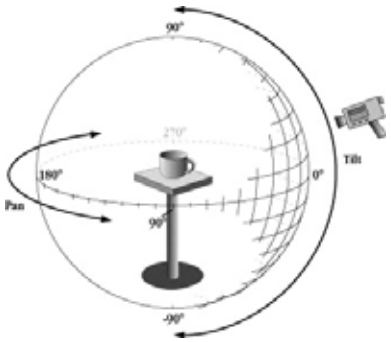


Abbildung 4.4: Virtuelle Hüllkugel

der Kamera wird dabei derart gerastert, daß die möglichen Kamerapositionen den Schnittpunkten von virtuellen Breiten– und Längengeraden entsprechen. Es werden nun von diesen ausgezeichneten Punkten Einzelaufnahmen des Objektes gemacht. Bei der Darstellung wird die Äquivalenz von Objektdrehung und Kameradrehung ausgenutzt, d.h. es ist für den Betrachter unerheblich, wie die Aufnahmen erstellt wurden, da derselbe Effekt erzielt werden kann.

4.2.3 Camera Movement

Das Hauptproblem bei der Umsetzung von Kamerabewegungen ist, daß sich hier sowohl die Kameraausrichtung als auch die Kameraposition dynamisch verändern können. Um diesem komplexen Problem beizukommen, reduziert man die Problemstellung und nimmt eine vereinfachende Einschränkung vor: man rastert die Szene insofern, daß man die möglichen Kamerapositionen auf die diskreten Orte beschränkt, an denen environment maps auch tatsächlich vorliegen.
Diese Beschränkung hat natürlich große Auswirkungen auf die Navigationsmöglichkeiten innerhalb der Szene, löst jedoch elegant das vorliegende Problem. Die Navigation zwischen verschiedenen Kamera–, und damit auch Betrachterpositionen

funktioniert über sogenannte *hot spots*, welche später (siehe 4.3.3) eingehender beschrieben werden.

4.2.4 Camera Zooming

Die Funktionalität, in einer virtuellen Umgebung ein Objekt oder einen Szenenausschnitt genauer betrachten zu können, gehört zu den Designvorgaben und wird im Fall von spezifischen Objekten in der Szene über die *Object Movies* realisiert.

Die Vergrößerung bzw. Verkleinerung eines Auschnittes der Szenerie kann grundsätzlich verschieden implementiert werden. Der einfachste Ansatz hierfür ist sicherlich einen Ausschnitt des Bildes zu skalieren, aber diese recht simple Idee birgt in sich ein Problem. Da es sich hier stets um gerasterte Flächen handelt, werden z.B. bei einer Vergrößerung mit diesem Algorithmus lediglich den entsprechenden Bildpunkten eine größere Fläche zugewiesen. Also wird man so keine feineren Details zu sehen bekommen, sondern lediglich eine gröbere Teilansicht mit effektiv niedrigerer Auflösung.
Ein anderer Lösungsansatz wäre z.B. die Verwendung mehrerer environment maps unterschiedlicher Auflösung. Diese könnten in eine Baumstruktur (z.B. Quad Trees) integriert werden, bei der Wahl einer Ansicht könnte man dann ähnlich dem Verfahren mit *mipmaps* beim texture mapping vorgehen und zwischen den zur Verfügung stehenden Ansichten interpolieren.

4.3 QTVR Inhalte

In diesem Abschnitt werden nun die einzelnen Komponenten bzw. Formate beschrieben, die QTVR ausmachen. Dabei wird besonders auf die Umsetzungen der in Abschnitt 4.2 besprochenen Ansätze eingegangen, auffällige Spitzfindigkeiten werden erläutert.

4.3.1 Panorama Movies

Panorama Movies sind letztlich Panoramabilder, in denen der Benutzer dynamisch seine Blickrichtung ändern, also "sich umsehen" kann. Dabei kommt die in 4.2.1 bereits angedeutete Idee zum tragen. Der Einsatz zylindrischer environment maps hat zur Folge, daß der Benutzer sich zwar horizontal um volle 360 Grad drehen kann, er jedoch vertikal nur einen Bildausschnitt von weniger als 180 Grad sehen kann. Das bedeutet faktisch, er kann nicht senkrecht nach oben oder unten sehen, oder etwa den Kopf in den Nacken legen und nach hinten sehen. Diese Einschränkung nimmt man hier in Kauf, da sie die Berechnung, Handhabung und Speicherung der Bilddaten erheblich vereinfacht.

Abbildung 4.5: Panoramabild

Erstellung von Panorama Movies

Die Aquisition zylindrischer Panoramabilder, welche man für entsprechende environment maps braucht, ist nicht ganz einfach. Es unterscheiden sich prinzipiell drei verschiedene Möglichkeiten, solche Bilder zu erstellen. Zunächst muß man dabei erst einmal synthetische und reale Szenen unterscheiden. Ist die zu betrachtende Szene synthetischer Natur, also eine generierte Landschaft aus einem Rechner, dann ist es lediglich eine Frage der Software, ob sie solche Ansichten ausgeben kann.
Will man andererseits eine reale Szene fotografieren, muß auf eine herkömmliche Kamera zurückgegriffen werden. Der Markt für spezielle Panoramakameras ist nicht sehr groß, es gibt jedoch Geräte, die solche Aufnahmen erlauben, ihr Einsatz und die Anschaffung ist i.A. sehr kostspielig.
Ein anderer Ansatz, der auch von den Machern von QTVR nahegelegt wird ist der, eine Serie normaler Fotos zu verwenden, die im Rahmen einer Bildbearbeitung (sog. *stitching*–Verfahren) zu einem Panorama zusammengesetzt werden. Dieser Vorgang wird nun beschrieben und ist natürlich auch mit gerenderten Einzelbildern möglich.

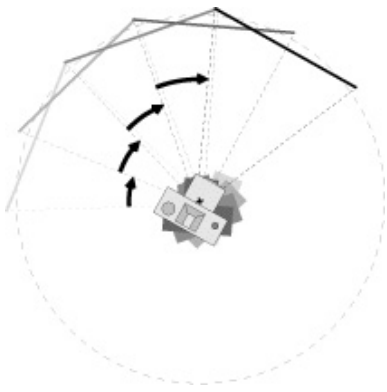


Abbildung 4.6: Panoramen aus Einzelbildern

Panoramen aus Einzelbildern

Die Grundidee ist also die, eine Reihe von Einzelaufnahmen einer Szene zu machen, jeweils aus einer etwas versetzten Blickrichtung. Abgesehen von Überlegungen zur Anzahl der Aufnahmen (also dem Grad der Rasterung) sind bei dieser Methode einige wichtige Dinge zu beachten:

- Die Aufnahmen müßen um einen festen *Rotationspunkt* gemacht werden, da sich sonst keine einheitliche Perspektive und keine klaren Überlappungsmerkmale zwischen den Bildern ergeben. Hierzu wird die Verwendung eines Stativs empfohlen.
- Es ist erforderlich, daß sich die Bilder überlappen, damit die benötigte Redundanz für den Stitch-Algorithmus gewährleistet ist.
- Der Rotationspunkt sollte auf den Brennpunkt der Kameralinse ausgerichtet sein, da sich sonst zwischen den Bildern einer Serie der sogenannte Parallax-Effekt einstellt. Darunter versteht man die unerwünschte Eigenschaft, daß sich nahe Objekte gegenüber weiter entfernten relativ zueinander verschieben.
- Eine einheitliche Beleuchtung ist, ebenso wie der Einsatz eines Stativs, entscheidend für die Qualität des Endergebnisses.
- Die benutzte Kameralinse sollte möglichst *rectilinear* sein. Das bedeutet, daß die Linse gerade Linien auch so auf dem Foto abbildet. Dies hängt wesentlich von der Qualität der Linse ab, kann aber notfalls auch nachträglich in einem Nachbearbeitungsschritt korrigiert werden.

Sind erstmal diese Einzelbilder erstellt und liegen diese digitalisiert vor, kann der Stitch-Algorithmus angewendet werden. Dabei werden anschaulich die Bilder übereinander gelegt und zu einem Zylinder gekrümmt. Mathematisch wird

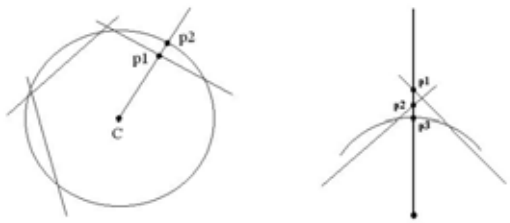


Abbildung 4.7: Reprojektion erfordert evtl. Interpolation

dieser Vorgang durch eine Projektion der Bildflächen auf einen Zylinder und anschliessende Pixelzuweisung realisiert. Dabei entsteht durch die Rasterung normalerweise ein Qualitätsverlust, da eventuell mehrere Bildpunkte auf den Fotos auf denselben Punkt auf der Zylinderfläche abgebildet werden.

Speicherung und Kompression

Die so entstandenen Panoramen werden zur Speicherung in kleinere Teilbilder aufgeteilt (sog. *tiles*). 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 zusätzlich mindestens noch eine Spur (meistens sogar zwei), die Informationen zu den einzelnen Panoramen und deren Verknüpfungen untereinander enthält, siehe dazu 4.3.3.
Bei der Kompression werden vorzugsweise asymmetrische Kompressionsverfahren eingesetzt. Da die Dekompression zeitkritisch ist sollte sie schneller erfolgen als der Komprimiervorgang, welcher durchaus länger dauern kann. Einschlägige Software-CODECs, die solches leisten, sind z.B. *Cinepak* und *Sorenson Video*. Aufgrund der erheblich besseren Kompressionsrate hat sich aber auch der *JPEG–Standard* für Panoramen etabliert. Hierbei handelt es sich jedoch um einen symmetrischen Kompressionsalgorithmus, der Kompressionsschritt dauert ebenso lange wie die Dekompression, was bei der Darstellung den Prozessor also mehr belastet.
Eine typische Kompressionsrate beim JPEG–Verfahren ist circa 10:1, sie läßt sich jedoch einstellen, sodaß an dieser Stelle eine Abwägung (*tradeoff*) zwischen Bildqualität (verlustbehaftetes Verfahren) und Darstellungsgeschwindigkeit geschieht.

Wiedergabe

Der Betrachter eines Panorama Movies sieht einen Auschnitt des Gesamtpanoramas auf seinem Bildschirm. Er kann nun innerhalb dieser Szene *navigieren*, indem er sich z.B. um sich selbst dreht (360 Grad) und dabei nach oben oder unten sieht. Wie bereits erwähnt kann der Benutzer einen Bildausschnitt im Detail betrachten, also heranzoomen, oder aber auch wieder herauszoomen.
Dabei geschieht folgendes:

1. Die an dem momentanen Bildausschnitt beteiligten tiles werden in einen schnellen Zwischenspeicher (*cache*) kopiert,

2. und in einem speziellen Puffer (*offscreen buffer* dekomprimiert. Dieser Puffer ist normalerweise kleiner als das volle Panorama, da jeweils nur ein Bruchteil des Panoramas sichtbar ist. Solange der momentane Bildausschnitt in dieser Region bleibt, ist also auch keine weitere Dekompression nötig.

3. Der sichtbare Bildausschnitt wird aus diesem Bereich extrahiert,

4. und es wird eine Perspektivenkorrektur (sog. *image warp*, siehe 4.2.1) durchgeführt, bevor das Bild auf dem Monitor zu sehen ist.

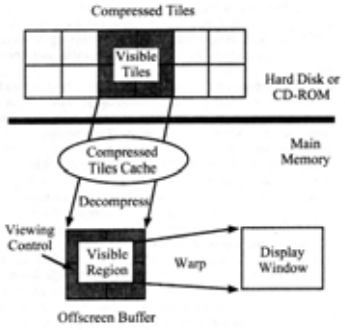


Abbildung 4.8: Wiedergabe

Der image warp, also die Reprojektion des zylindrischen Bildes auf eine planare Ansicht geschieht in Echtzeit, mit Hilfe eines zweistufigen Algorithmus'. In einem ersten Schritt findet die eigentliche Reprojektion statt, dies geschieht während sich der Benutzer noch in der Szene bewegt. Bleibt er stehen, wird zusätzlich eine Bildglättung durchgeführt, die *Aliasing*–Effekte der Reprojektion ausgleicht.

4.3.2 Object Movies

QTVR Object Movies bieten horizontale und vertikale Bewegungsfreiheit zur Manipulation eines Objektes. Grundsätzlich sind zwei verschiedene Typen von Object Movies zu unterscheiden, *Rotating Movies* und *Absolute Movies*. Im folgenden werden im wesentlichen Rotating Movies beschrieben, da Absolute Movies eher eine Spielart darstellt und aus der anderen Technik hervorgeht.

Rotating Movies

Bei Rotating Movies geht man von der Vorstellung aus, der Benutzer könne ein Objekt in die Hand nehmen und nach Belieben drehen und wenden. Dazu werden von dem Objekt eine Reihe von Aufnahmen aus unterschiedlichen Perspektiven gemacht, die in einem 2–dimensionalen Feld (*array*) indiziert werden (siehe 4.2.2). Die Fotos sind in dieser Struktur so angeordnet, daß sich benachbarte Bilder

Abbildung 4.9: Aufnahmen in einem Bild–Array

ähnlich sind. Dazu ordnet man alle Aufnahmen von einem virtuellen Breitengrad aus horizontal in dem Feld an, die von Längengraden vertikal.

Aufnahme der Einzelbilder

Um das erforderliche Bildmaterial für ein Object Movie zusammenzustellen, bedarf es einer gewissen Apparatur. Das Objekt befindet sich dabei normalerweise auf einer Art Plattform. Nun gibt es viele denkbare Ansätze, eine virtuelle Hüllkugel zu erzeugen. Eine Möglichkeit ist sicherlich der Einsatz eines Roboterarms, eine andere, einfachere Lösung wäre z.B., eine bewegliche Plattform zu verwenden und zusätzlich einen Meridian nachzubauen, an dem die Kamera befestigt ist. Die nun mögliche kombinierte Bewegung der Kamera auf dem Längengrad und dem Objekt auf der Plattform erlaubt es, die nötigen Ansichten zu erzeugen.

Typischerweise verwendet man eine Art Studio mit neutraler Hintergrundfarbe, um anschliessend die *Bluescreen*–Technik anwenden zu können. Auf diese Weise kann man in einem Nachbearbeitungsschritt irrelevante Informationen aus den einzelnen Bildern herausnehmen, bzw. –retouchieren.
Wichtig, auch in diesem Zusammenhang, ist die Anzahl der Aufnahmen, also die Rasterung der Hüllkugel. Eine von Apple empfohlene Lösung sieht eine Einteilung in 10 Grad–Abschnitte vor. Bei einer nötigen vollen horizontalen Umdrehung von 360 Grad und einer vertikalen Bewegung um 180 Grad (einschließlich der "Pole"), entspräche dies also

$$\frac{360 * 19}{10} = 684$$

erforderlichen Aufnahmen.

Beleuchtung

Bei dem geschilderten Vorgang spielt die ambiente Beleuchtung sowie die Ausleuchtung des Objekts eine wichtige Rolle. Diese Überlegungen hängen wiederum von der Beschaffenheit des Objekts und dem erwünschten Effekt beim Betrachter des Object Movies ab.
Handelt es sich beispielsweise um ein Objekt mit einer reflektierenden Oberfläche, so sollte man berücksichtigen, was in der Reflektion zu sehen ist (i.A. möchte man z.B. die Reflektion der aufnehmenden Kamera auf der Objektoberfläche vermeiden).
Grundsätzlich hat man bei der Beleuchtung zwei Optionen, zum einen eine stationäre Lichtquelle, zum anderen eine, die an der Kamera befestigt ist und sich somit mit um das Objekt bewegt. Wichtig ist hier die Einsicht, wie sich diese Beleuchtung auf die spätere Betrachtungsweise auswirkt. Bei einer nichtstationären Lichtquelle wird das Objekt auf allen Fotos von derselben Entfernung (also mit derselben Intensität) und stets frontal beleuchtet sein. Der Beobachter wird also das Gefühl haben, er wäre stationär und könne das Objekt manipulieren.
Ist hingegen die Beleuchtung durch eine stationäre Lichtquelle realisiert, werden, abhängig von der Perspektive, zusätzlich Schatten auf den Bildern zu sehen sein,

der Betrachter hat den Eindruck, er bewege sich um das Objekt herum, während er es betrachtet.
Eine hybride Beleuchtungsform entsteht, wenn man eine stationäre Lichtquelle verwendet und sich das Objekt auf einer bewegenden Plattform dreht (z.B. die zweite Apparatur oben). Dem Betrachter fällt nun auf, daß sich bei der Drehung des Objekts seine Position relativ zur Lichtquelle nicht verändert, obwohl er sich doch vermeintlich um das Objekt herum bewegt.

Absolute Movies

Diese Art von Object Movies nutzt die Datenstruktur des Feldes anders aus, der Zugriff auf die Feldelemente erfolgt über absolute Koordinaten, was also wahlfreien Zugriff auf die Elemente erlaubt. Dabei geht die Illusion von einer *Persistenz* der Ansicht, wie sie bei Rotating Movies gegeben ist (siehe unten), verloren. Andererseits lassen sich andere Effekte erzielen, sie gehören jedoch nicht mehr in den Bereich des IBR.

Speicherung und Kompression

Object Movies werden ebenfalls in einer normalen QT–Videospur gespeichert. Dabei wird allerdings über einen speziellen *Object Movie Handler* zusätzlich die Feldindizierung erhalten. Also werden die Daten in einer Folge gespeichert aber der Zugriff darauf erfolgt nichtlinear. Obwohl die Einzelbilder linear ge-

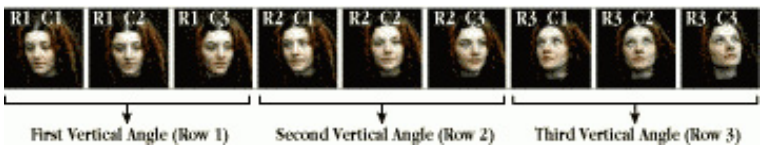


Abbildung 4.10: Lineare Speicherung eines Object Movies

speichert werden, kann natürlich aufgrund der Zugriffsstrategie keine *keyframe*–Kompression wie z.B. MPEG angwandt werden. Vielmehr bietet sich eine Kompression der Einzelbilder nach bewährtem Standards, wie z.B. JPEG, an.

Wiedergabe

Die Wiedergabe erfolgt über einen Zugriff auf das logische Feld der Einzelaufnahmen. Dreht der Betrachter das Objekt horizontal, dann wird auf die nebeneinander angeordneten Bilder zugegriffen, die vertikale Bewegung verläuft analog.
Absolute Movies unterscheiden sich in diesem Punkt, da der Zugriff auf das Feld absolut geschieht, es wird keine Relation zu dem zuletzt dargestellten Bild hergestellt.

Animationen

Animationen lassen sich erzeugen, indem man für eine Perspektive mehrere Einzelbilder aufnimmt. Diese werden dann bei der Wiedergabe hintereinander abspielt. Dadurch erhöht sich die Anzahl der Bilder natürlich drastisch (der Fakor entspricht der Anzahl der zusätzlichen Bilder pro Einstellung).

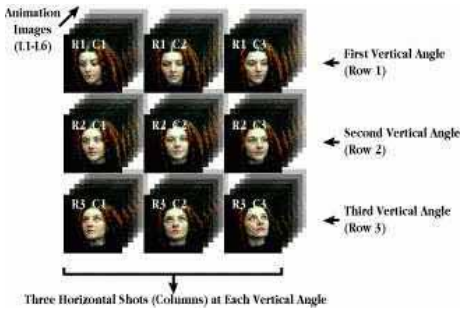


Abbildung 4.11: Animationen von Objekten

4.3.3 Multinode Panoramas

Diese Art von Panoramen integrieren den im Abschnitt 4.2.3 beschriebenen Ansatz: die Veränderung der Kameraposition in einer Szene. Der Name impliziert bereits die inhärente Struktur, einzelne Panoramen (oder *nodes*) werden zu einem Komplex verknüpft und geben so die Szene wieder. Die Navigation zwischen Pan-

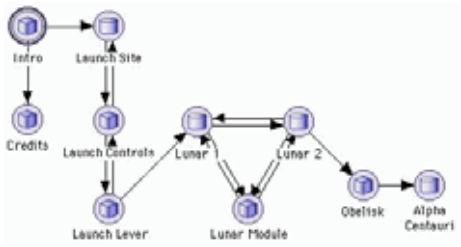


Abbildung 4.12: Verknüpfung von Panoramen

oramen ist möglich und funktioniert mittels ausgewiesener, interaktiver Stellen. Ein Multinode Panorama besteht aus drei QT–Spuren: einer Spur, die die logischen Verknüpfungen zwischen den einzelnen Panoramen enthält, und zwei Videospuren, die jeweils die Panoramen und ihre zugeordneten hot spot–Bilder beinhalten.

hot spots

Hot spots werden über eine eigene QT–Videospur gespeichert. Normalerweise wird zu jedem Panorama aus dem Panorama-Track ein gleichgrosses hot spot–Bild gespeichert. In diesem Bild sind diejenigen Bildelemente (z.B. eine Tür)

farbig markiert, die interaktiv reagieren können (z.B. auf einen Mausklick). Bleiben wir bei unserem Beispiel, so könnte bei einem Mausklick auf die Tür ein Sprung zu einem Panorama von der Eingangshalle des nun betretenen Hauses geschehen. Hot spots können jedoch nicht nur Panoramen, sondern auch Pan-

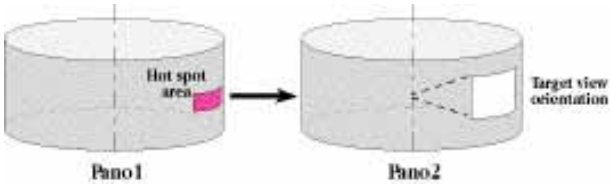


Abbildung 4.13: Hotspots zur Navigation

oramen mit Object Movies oder anderen multimedialen Inhalten (*URL*s, Bilder, Text, Audio, etc.) verbinden.

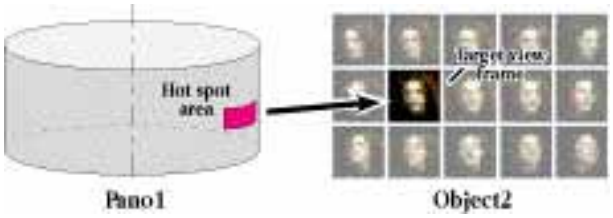


Abbildung 4.14: Navigation zwischen verschiedenen Elementen

4.3.4 Zoomen in QTVR

Das zoomen innerhalb eines Panoramas oder Object Movies wird von QTVR unterstützt. Der implementierte Algorithmus verwendet das Prinzip der Bildskalierung, was eine sehr effiziente Berechnung erlaubt. Die in 4.2.4 angesprochenen Probleme werden eingeschränkt, indem man einen Parameter (sog. *zoom factor* einführt, der das Heranholen eines Bildausschnitts nur innerhalb fester Schranken erlaubt.
Die derzeitige Implementierung unterstützt kein Mehrbild–basiertes zoomen, diese Option wird aber in [Chen '95] diskutiert und für zukünftige Versionen offen gehalten.

Literaturverzeichnis

[Chen '95] Shenchang Eric Chen. QuickTime VR - An Image-Based Approach to Virtual Environment Navigation. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '95)*, Seiten 29–38, 1995.

[Kitchens '98] Susan A. Kitchens. *The QuickTime VR Book*. 1998.

5 View Morphing

Fred Schättgen

Ziel des hier vorgestellten Verfahrens ist es, anhand von mehreren Ansichten einer Szene möglichst originalgetreu die Bildsequenz zu berechnen, die entstehen würde, wenn man sich mit einer Kamera auf einem Pfad zwischen den Kamerapositionen der Einzelansichten bewegen würde. Das Verfahren verwendet gängige Morphingtechniken als elementaren Bestandteil. Da Morphing alleine dem angepeilten Einsatzbereich - nämlich Interpolation zwischen Bildern verschiedener Ansichten derselben Szene - nicht gerecht werden würde, werden die Bilder zusätzlich einem Vor- und Nachverarbeitungsschritt unterworfen. Man versucht dabei, auf die explizite Verwendung von 3D-Informationen zu verzichten. Dieser Artikel basiert im wesentlichen auf den Artikeln [Seitz & Dyer '96a] und [Seitz & Dyer '96b] von Steven Seitz und Charles Dyer von der University of Wisconsin-Madison.

5.1 Ansatz

Der erste Gedanke, der sich aufdrängt, wenn man mit einem Computer eine Ansicht einer Szene generieren soll, ist der, dass man erst einmal eine geeignete Repräsentation der Szene - z.B. auf Dreiecks- oder Splinebasis - im Rechner braucht. Damit ist es dann ein Leichtes, beliebige Ansichten zu synthetisieren. Die Rekonstruktion eines 3D-Modells aus Bildern ist jedoch kein einfaches Unterfangen, da das Problem i.d.R. hoffnungslos unterbestimmt ist. Außerdem müssen die Daten hier von einer Repräsentation in eine völlig andere und wieder zurück transformiert werden, so dass im Vergleich mit dem Ausgangsmaterial mit einem erheblichen Qualitätsverlust zu rechnen ist. Ohne Tricks wie z.B. Texture Mapping bedarf es außerdem einer enormen Menge von Dreiecken, um aus dem Modell wieder ein passables Bild generieren zu können. Die Verwendung von Texturen dagegen ist für ein Verfahren, das idealerweise automatisch ablaufen sollte zu aufwendig. Nun darf man nicht vergessen, dass ein 3D-Modell der Szene zwar hilfreich wäre, aber dennoch nicht das eigentlich Ziel ist. Ein vollständiges 3D-Modell ist möglicherweise mehr, als man eigentlich braucht. Gesucht ist lediglich ein Verfahren, das zwei Bilder kontinuierlich ineinander überführt. Das legt den Einsatz von Morphingmethoden nahe, die vor einigen Jahren bei Filmen und in der Werbung eine große Popularität erreicht haben. Hierbei wird normalerweise zwischen zwei völlig verschiedenen Bilder derart interpoliert, dass beim

Betrachter der Eindruck entsteht, dass sich das Objekt im ersten Bild langsam in das im zweiten Bild verwandelt. Diese Verfahren haben in der Regel mit 3D-Rekonstruktion nichts zu tun, und darauf baut View Morphing auf. Damit behält man beim View Morphing während der gesamten Verarbeitung die ursprüngliche Repräsentation der Bilder - also Rasterdaten - bei, so dass man eine bessere Bildqualität als bei anderen Ansätzen erwarten kann.

5.2 Morphing

Beim Morphing versucht man zwischen den Bildern von zwei Objekten so zu interpolieren, dass es den Anschein hat, als würde sich Objekt A langsam in Objekt B verwandeln. Was das genau heißt bleibt erst einmal offen, wichtig ist lediglich, dass es "echt" aussieht. Bei einer derart schwammigen Aufgabenstellung ergeben sich eine Unmenge von Freiheitsgraden, so dass man nicht erwarten kann, dass es dafür *die* Lösung gibt. So ist z.B. offen, wie verschiedene, zusammengehörige Merkmale in den Bildern gefunden werden und wie dabei möglicherweise auftretende Mehrdeutigkeiten aufgelöst werden, so dass sich in der Bildsequenz auch tatsächlich eine Ecke von ihrer Ausgangsposition zur Zielposition im zweiten Bild bewegt. Und auch der Pfad, auf dem sie das tun, ist damit noch nicht festgelegt. Daher verlangen Morphingprogramme an dieser Stelle gewöhnlich Handarbeit. Das ist für ihr ursprüngliches Einsatzgebiet bis zu einem gewissen Grad auch erwünscht, um die erzeugte Sequenz den eigenen Vorstellungen anpassen zu können. Die beeindruckendsten Morphingeffekte lassen sich also logischerweise mit manuellen bzw. halbautomatischen Methoden erzielen.
Bei gängigen Morphingprogrammen gibt man für einen kleinen Teil des Bildpaares die Korrespondenzen von Hand an - so auch in Abb. 5.1. Im einfachsten Fall setzt man einzelne Markierungspunkte auf sichtbare Merkmale im einen Bild und verschiebt dann im anderen Bild die Markierung auf die Position, an der sich das entprechende Merkmal in diesem Bild befindet. An diesen Punkten kennt das Programm somit genau den Partnerpunkt im anderen Bild. Bei Bildpunkten, die nicht mit einem Markierungspunkt versehen wurden muss dagegen der korrespondiernde Punkt irgendwie interpoliert werden. Dabei werden die Markierungspunktpaare als Verschiebungsvektoren aufgefasst und umso höher gewichtet, je näher sie dem Punkt liegen, dessen Partnerpunkt ermittelt werden soll. Die meisten Programme arbeiten allerdings nicht mit einzelnen Punkten, sondern mit Linien. Dabei sind bilden manchmal nur die Endpunkte eine eindeutige Zuordnung zweier Punkte zueinander, während für einen Punkt in der Mitte der Linie nur sichergestellt ist, dass er auch im anderen Bild mit einem Punkt auf der entsprechenden Linie verknüpft wird; manchmal besteht auch eine eins-zu-eins-Beziehung zwischen den Punkten auf der Linie. Entsprechend benutzen einige Programme auch Flächen oder Splines. Für das View Morphing wären aber vollautomatische Methoden wünschenswert, da hier beim Ergebnis sowieso

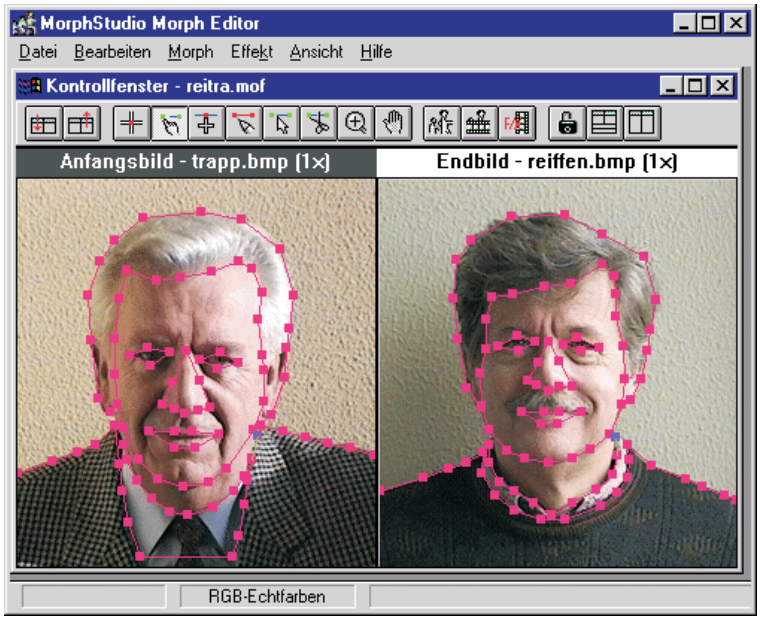


Abbildung 5.1: Die Arbeitsumgebung eines typischen Morphingprogramms

kein kreativer Spielraum erwünscht ist.

Ein grosser Vorteil des View Morphings ist es gerade, dass sich dafür vergleichsweise einfach automatisch die Korrespondenzen bestimmen lassen, da dafür gesorgt wird, dass man lediglich eine Bildzeile durchsuchen muss, um ein Merkmal aus einem Bild im anderen Bild wiederzufinden.

Halbwegs einfach ist das Auffinden von zueinander passenden Punkten damit aber auch nur an Kanten im Bild. Über das Innere von homogen eingefärbten Flächen lässt sich ohne Hintergrundwissen nichts weiter sagen. Aber wie bereits eingangs erwähnt interessiert uns nicht eine vollständige Rekonstruktion des Objekts; wir wollen lediglich wissen, wie die Szene von irgendwo zwischen den beiden Kameras aussehen würde. Dazu reicht es aber aus, dass wir von allen sichtbaren Kanten die Korrespondenzen kennen (Abb. 5.2). Der Bereich dazwischen - nach Voraussetzung homogen - muss nur zwischen den richtigen Kanten liegen, auf die genauen Pixelkorrespondenzen kommt es hier gar nicht an. Allerdings müssen die beiden Ansichten dazu eine bestimmte Bedingung (Monotonie) erfüllen, die im nächsten Abschnitt behandelt wird.

Damit View Morphing korrekte Ergebnisse liefert, stellt es an die verwendete Morphingmethode einige Bedingungen, die aber für Morphingprogramme gewissermaßen bereits den kleinsten gemeinsamen Nenner darstellen: Zwischen den korrespondierenden Bildmerkmalen sollte linear interpoliert werden; ein Merkmal bewegt sich also gleichförmig auf einer Linie vom Ausgangs- zum Zielpunkt - es geht hier also nur um die Bewegungsgleichung eines Punktes während des

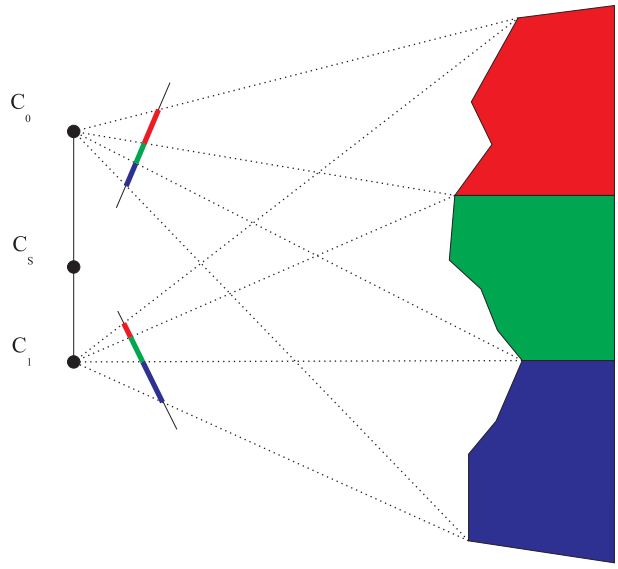


Abbildung 5.2: Um das Erscheinungsbild der Szene von C_s aus vorherzusagen, müssen nur Korrespondenzinformationen für die Kanten im Bild vorliegen.

Überblendens, nicht um die Interpolationsmethode, mit der die Korrespondenzen für Bildbereiche, in denen keine Korrespondenzen explizit angegeben wurden bestimmt werden. Ansonsten hat man bei der verwendeten Morphingmethode freie Wahl.

5.3 Monotonie

Offenbar sind nicht in allen Paaren von Ansichten einer Szene genügend Informationen vorhanden, um zwischen ihnen interpolieren zu können. So führen beispielsweise Verdeckungen dazu, dass man keine Aussage über die Korrespondenz von einzelnen Bildteilen machen kann. Die Eigenschaften, die ein Bildpaar aufweisen muss, um prinzipiell die Interpolation von Zwischenansichten zu ermöglichen werden formal durch das Monotoniekriterium festgelegt:
Gegeben seien zwei Ansichten V_1 und V_2 mit den zugehörigen Kamerapositionen C_1, C_2 und den Bildebenen $\mathcal{I}_1$, $\mathcal{I}_2$. Durch C_1, C_2 und einen beliebigen, sichtbaren Punkt P wird eine Epipolarebene aufgespannt, deren Schnitte mit $\mathcal{I}_1$ und $\mathcal{I}_2$ die zu P gehörenden Epipolarlinien sind - die Linienpaare also für die gilt: wenn ein Szenenpunkt bei einer Projektionsebene auf eine bestimmte Epipolarlinie abgebildet wird, dann kommt das Bild des Punktes auf der anderen Projektionsebene auf der korrespondierenden Epipolarlinie zu liegen. Die Monotoniebedingung verlangt nun, dass für jede Menge von sichtbaren Punkten in der Szene, die dieselbe Epipolarebene aufspannen, die Punkte so auf die jeweilige Bildebene abgebildet werden, dass die Bilder jedes Punktes auf beiden Epipolarlinien jeweils die gleiche

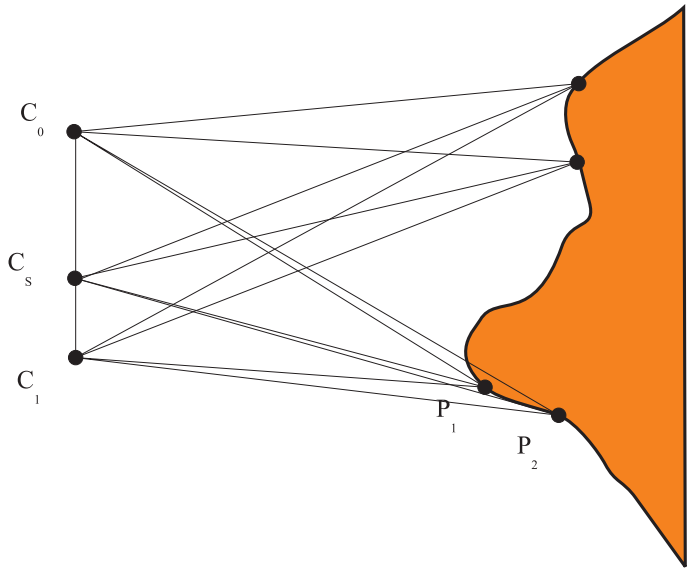


Abbildung 5.3: Szene mit verletzter Monotonie: P_2 ist von C_0 aus nicht sichtbar. Und selbst wenn er von beiden Kameras aus sichtbar wäre, würde P_2 von C_0 aus links von P_1 liegen, von C_1 aus aber zur Rechten von P_1, womit die Monotonie verletzt wäre.

Reihenfolge beibehalten (s. Abb. 5.3).
Dabei fällt auf, dass das Monotoniekriterium nur die Kameraposition einschränkt, nicht jedoch die Lage der Projektionsebenen, also im wesentlichen die Blickrichtung und die Brennweite. Das bedeutet, dass die Monotonie bei einer Reprojektion der Bilder erhalten bleibt. Aus der Monotonie zweier Ansichten V_1, V_2 folgt weiterhin die Monotonie der Ansichtspaare V_1V_s und V_sV_2, wobei V_s eine Ansicht sei, deren optisches Zentrum C_s auf der Verbindungslinie der optischen Zentren C_1 und C_2 liegt: Die Monotoniebedingung wird nur dann verletzt, wenn es zwei sichtbare Punkte P_1, P_2 in der Szene gibt, so dass die Gerade $\overline{P_1P_2}$ die Strecke zwischen C_1 und C_2 schneidet. Ist also (o.b.d.A.) V_1V_s nicht monoton, so kann auch V_1V_2 nicht monoton sein. Also folgt aus der Monotonie für ein Paar von Ansichten auch die Monotonie auf allen Teilstücken der Strecke zwischen den optischen Zentren der Ansichten.

5.4 View Morphing

In diesem Abschnitt wird gezeigt, wie aus zwei Ansichten eine neue, dazwischen liegende Ansicht erzeugt werden kann. Gegeben seien zwei Ansichten mit den Bilder $\mathcal{I}_0$ und $\mathcal{I}_1$ und den Projektionsmatrizen Π_0 und Π_1. Außerdem sei die Projektionsmatrix Π_s für die gesuchte Ansicht bekannt. Gesucht ist nun das Bild $\mathcal{I}_s$, das bei einer Aufnahme der Szene mit einer Kamera mit der Konfiguration Π_s entstehen würde.

5.4.1 Parallele Ansichten

Betrachten wir zuerst einen Sonderfall der oben geschilderten Aufgabe: Wir wollen uns erst einmal auf Ansichten mit parallelen Bildebenen beschränken. Wir nennen solche Ansichten dementsprechend parallele Ansichten. In diesem Fall haben die Projektionsmatrizen Π_0 und Π_1 folgende Form:

$$\Pi_0 = \begin{bmatrix} f_0 & 0 & 0 & 0 \\ 0 & f_0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

$$\Pi_1 = \begin{bmatrix} f_1 & 0 & 0 & -f_1 C_X \\ 0 & f_1 & 0 & -f_1 C_Y \\ 0 & 0 & 1 & 0 \end{bmatrix}$$

Dabei haben wir uns das Weltkoordinatensystem so zurechtgelegt, dass C_0 gerade auf dem Koordinatenursprung zu liegen kommt und die z-Koordinate von C_1 gerade 0 ist. Die einzigen Variablen sind dann die Position der zweiten Kamera und die Brennweite der beiden Kameras. Nun versuchen wir, eine neue Ansicht zwischen C_0 (hier $[0\,0\,0]$) und C_1 dadurch zu erzeugen, dass wir zwischen zwei korrespondierenden Bildpunkten $p_0 \in \mathcal{I}_0$ und $p_1 \in \mathcal{I}_1$ linear interpolieren. Für die Position des neuen Bildpunktes in $\mathcal{I}_s$ ergibt sich damit

$$(1-s)p_0 + sp_1 = (1-s)\frac{1}{Z}\Pi_o P + s\frac{1}{Z}\Pi_1 P = \frac{1}{Z}\Pi_s P$$

wobei gilt:

$$\Pi_s = (1-s)\Pi_0 + s\Pi_1$$

Das bedeutet, dass das Bild, das durch lineare Interpolation der Bilder p_0 und p_1 eines Szenenpunktes P entstanden ist, auch durch eine perspektivische Projektion der Szene mit Hilfe der Projektionsmatrix Π_s hätte erzeugt werden können. In diesem Fall liefert die lineare Interpolation also ein "sinnvolles" Ergebnis, nämlich eine neue Ansicht der Szene, wie sie eine reale - wenn auch idealisierte - Kamera gemacht haben könnte.
Wenn man den Parameter s von 0 bis 1 laufen lässt und jeweils die entsprechende neue Ansicht berechnet, scheint sich die Kamera linear von C_0 nach C_1 zu bewegen und dabei die Brennweite von f_0 nach f_1 zu ändern.

5.4.2 Nicht-parallele Ansichten

Wenn wir nun den allgemeinen Fall mit beliebig orientierten Projektionsebenen betrachten, müssen wir erst einmal untersuchen, ob man dieses Problem nicht genauso wie den Sonderfall der parallelen Ansichten angehen kann.

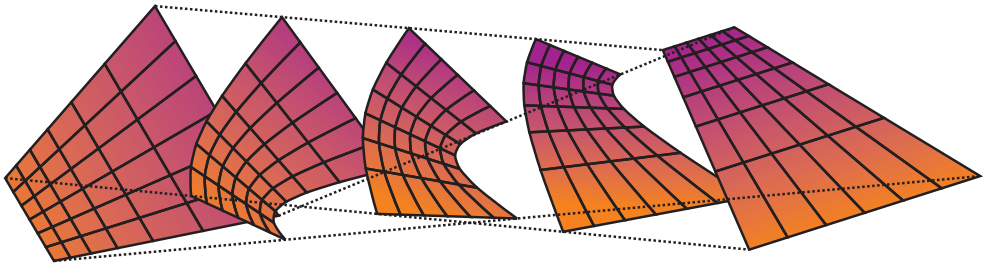


Abbildung 5.4: Lineare Interpolation zwischen zwei Ansichten kann zu solchen Verzerrungen führen.

In Abb. 5.4 wurde zwischen den Bildern eines rechteckigen Gitters linear interpoliert. Der Winkel zwischen den beiden Aufnahmen ist extrem gross gewählt, aber der Effekt ist auch bei kleinen Winkeln zu sehen, wenn er bei realen Szenen auch erst bei einer Animation deutlich wird. Wie kann es nun sein, dass eine lineare Interpolation die Ränder eines Rechtecks zu Kurven verbiegt?
Betrachten wir dazu einmal einen einzelnen Bildpunkt. Da die beiden Bilder $\mathcal{I}_0$ und $\mathcal{I}_1$ perspektivische Projektionen einer Szene sind, gibt es eine Matrix

$$\dot{H} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix}$$

so dass für die zwei Bildpunkte p_0 und p_1 eines Szenenpunktes P gilt: $p_0 H = p_1$, wobei p_0 und p_1 in homogenen Koordinaten gegeben seien. Rechnet man mit euklidischen Koordinaten, so muss man die homogene Komponente normieren und erhält damit für zwei Bilder eines Punktes folgende Beziehung:

$$\dot{H}(x,y) = (\frac{ax+by+c}{gx+hy+i}, \frac{dx+ey+f}{gx+hy+i})$$

Wenn man nun beispielsweise eine Ansicht genau in der Mitte berechnen wollte, so erhielte man für die Koordinaten eines Punktes

$$p_{0,5} = \frac{1}{2}(p + H(p)) = \frac{1}{2}(\frac{ax+by+c}{gx+hy+i} + x, \frac{dx+ey+f}{gx+hy+i} + y)$$

$$= \frac{1}{2}(\frac{ax+by+c+gx^2+\ldots}{gx+hy+i}, \ldots)$$

Offensichtlich lässt sich diese Abbildung nicht mehr als Multiplikation mit einer 3×3-Matrix interpretieren. Damit ist auch das Ergebnis bei linearer Interpolation im Allgemeinen keine perspektivische Abbildung mehr, so dass der Ansatz, der bei parallelen Bildebenen funktioniert hat hier so nicht mehr in Frage kommt.
Der allgemeine Fall der Problemstellung wird nun dadurch gelöst, dass man ihn auf den parallelen Fall reduziert. Das ganze Verfahren stellt damit einen dreistufigen Prozess dar:

1. Parallelisieren der Ansichten. Dieser Schritt wird auch *Prewarp* genannt.

2. Lineare Interpolation der beiden parallelen Ansichten (*Morphing*).

3. Reprojektion des interpolierten Bildes auf die gewünschte Bildebene (*Postwarp*)

Schreibt man die Projektionsmatrizen Π_0 und Π_1 in der Form $[H_0|-H_0C_0]$ bzw. $[H_1|-H_1C_1]$, so sieht man sofort, mit welcher Tranformation man die Bildebenen parallel ausrichten kann: Auf das erste Bild wendet man ${H_0}^{-1}$ an, auf das zweite Bild ${H_1}^{-1}$. Allerdings ist diese Vorgehensweise problematisch, da durch die Reprojektion extreme Verzerrungen auftreten können. Man sollte also die Achsen seines Koordinatensystems so auswählen, dass die Verzerrungen durch die Reprojektion so gering wie möglich ausfallen. Als praxistauglicher Ansatz schlagen die Autoren vor, C_1 auf die x-Achse zu legen und als y-Achse das Kreuzprodukt aus den beiden Bildnormalen zu nehmen. Das funktioniert dann besonders gut, wenn die beiden Kameras auf denselben Szenenpunkt ausgerichtet sind, was sicher eine vertretbare Annahme darstellt. Bei entsprechenden Kamerakonfigurationen kann diese Daumenregel allerdings auch beliebig schlecht abschneiden. Die ganze Prozedur sieht nun kurz so aus (vgl. Abb. 5.5) :

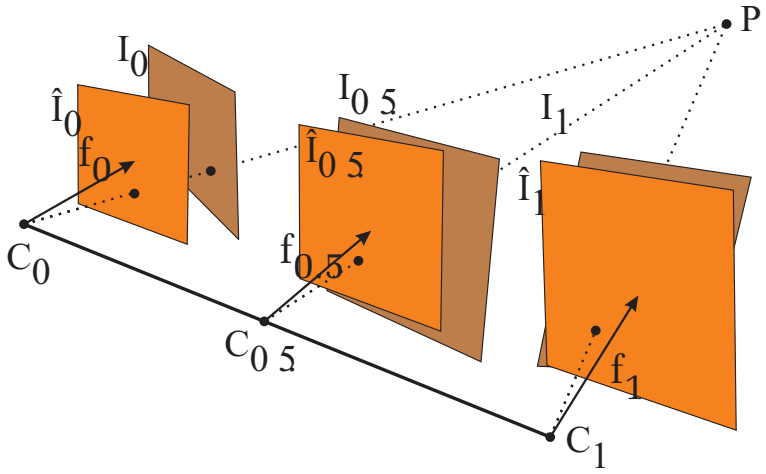


Abbildung 5.5: Der Prewarpschritt parallelisiert I_0 und I_1; dann wird zwischen den gefundenen Korrespondenzen interpoliert und das Ergebnis $\hat{I}_{0,5}$ mit dem Postwarp zu $I_{0,5}$ zurücktransformiert.

1. Prewarp: H_0^{-1} auf $\mathcal{I}_0$, H_1^{-1} auf $\mathcal{I}_1$ anwenden.

2. Lineare Interpolation der beiden parallelen Ansichten ergibt $\hat{\mathcal{I}}_s$.

3. Postwarp: Zielansicht $\mathcal{I}_s$ erzeugen, indem man H_s auf $\hat{\mathcal{I}}_s$ anwendet.

Entscheidend beim Prewarping-Schritt ist, dass durch die Reprojektion nur die Bildebene verändert wird, nicht aber das Projektionszentrum. Die Monotonie bleibt daher auch in den reprojezierten Bildern erhalten. Weil wir die Bildebenen mit den zugehörigen Projektionsmatrizen $[H_n| - H_nC_n]$ mit Hilfe von ${H_n}^{-1}$ parallelisiert haben, haben die Abbildungsmatrizen der Bilder nach dem Prewarpschritt die Form $[I| - C_n]$. Damit die Bildebenen parallel sind wäre es nicht zwingend notwendig, dass der linke Teil der Projektionsmatrix die Identität ist. Allerdings kommt einem diese Tatsache bei der Implementierung des View Morphing gerade recht, denn damit liegen die zwei Bilder eines Punktes P in $\mathcal{I}_0$ und $\mathcal{I}_1$ in derselben Zeile. Damit wird es sehr viel einfacher, ein Punktmatchingverfahren zum Auffinden der Korrespondenzen in den beiden Bildern zu implementieren. Um einen korrespondierenden Punkt zu finden, muss man nur noch entlang einer Bildzeile suchen - oder ggf. in einem engen Schlauch, um die unvermeidbaren Ungenauigkeiten bei der Kamerakalibrierung zu berücksichtigen.
Für zwei Ansichten mit den Projektionsmatrizen $\Pi_0 = [I|0]$ und $\Pi_1 = [H_1| - H_1C_1]$, die diese Eigenschaft haben, haben H_1 und C_1 diese Form:

$$H_1 = \begin{bmatrix} a & b & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$C_1 = [C_X\, 0\, 0]^T$$

Ansichten in einer solchen Anordnung befinden sich in einer sogenannten *kanonischen Konfiguration.*

5.5 View Morphing mit unkalibrierten Ansichten

Bei den bisherigen Betrachtungen sind wir davon ausgegangen, dass die genaue Kamerakonfiguration bekannt ist. Bei den intrinsischen Parametern wie z.B. der Brennweite stellt das für gewöhnlich auch kein großes Problem dar. Die Position und Ausrichtung der Kamera zu bestimmen kann jedoch aufwendig werden, insbesondere, wenn man auf bereits vorhandenes Bildmaterial zurückgreifen muss, für das diese Informationen nicht bekannt sind. Glücklicherweise kommt man auch mit weniger Informationen als dem vollständigen Satz an Kameraparametern aus. Vielmehr genügt es, dass die Fundamentalmatrix der beiden Ansichten bekannt ist. Die Fundamentalmatrix ist eine 3×3-Matrix mit Rang 2 und erfüllt die Bedingung

$$p_1^T F p_0 = 0$$

wobei p_0 und p_1 die Bilder eines beliebigen Szenenpunktes P auf $\mathcal{I}_0$ und $\mathcal{I}_1$ sind. F ist bis auf skalare Vielfache eindeutig bestimmt und lässt sich vor allen Dingen aus einer kleinen Anzahl (mind. 8) bekannter Punktkorrespondenzen berechnen.

5.5.1 Berechnen der Fundamentalmatrix

Um die Fundamentalmatrix hinreichend genau zu bestimmen, sollte man aber deutlich mehr als acht Punkte verwenden und für dadurch überbestimmte Gleichungssystem die Lösung mit dem kleinsten quadratischen Fehler suchen.
Es seien m, m' die beiden Bilder eines Szenenpunktes. Für jedes Punktepaar muss dann für die Fundamentalmatrix gelten $(m')^T F m = 0$. Ausgeschrieben ergibt sich

$$x x' f_{11} + x y' f_{12} + x f_{13} + y x' f_{21} + y y' f_{22} + y f_{23} + x' f_{31} + y' f_{32} + f_{33} = 0$$

wobei m und m' in homogenen Koordinaten als $[x\,y\,1]$ bzw. $[x'\,y'\,1]$ dargestellt sein sollen. Für alle Punkte ergibt sich dann ein lineares Gleichungssystem der Form

$$\begin{bmatrix} x'_1 x_1 & x'_1 y_1 & x'_1 & y'_1 x_1 & y'_1 y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x'_n x_n & x'_n y_n & x'_n & y'_n x_n & y'_n y_n & y'_n & x_n & y_n & 1 \end{bmatrix} f = Af = 0$$

Da dieses Gleichungssystem aufgrund zufälliger Fehler i.A. keine Lösung hat sucht man $\min_f ||Af||^2$ mit $||f|| = 1$. Die Lösung dieser Gleichung ist der Eigenvektor zum kleinsten Eigenwert von $A^T A$ (siehe [Zisserman '97]).

5.5.2 Ausrichten der unkalibrierten Bilder

Da wir nun die Projektionsmatrizen für die beiden Kameras nicht mehr explizit gegeben haben, ist es nicht mehr offensichtlich, wie man die beiden Bilder reprojezieren muss, damit die neuen Bildebenen identisch ausgerichtet sind.
Es seien $\mathcal{I}_0$, $\mathcal{I}_1$ zwei Ansichten mit den Projektionsmatrizen $\Pi_0 = [I|0]$ (o.B.d.A.) und $\Pi_1 = [H_1|-H_1 C_1]$. Die Epipole, also die Projektionen der Projektionszentren C_n in die jeweils andere Ansicht (C_1 in $\mathcal{I}_0$ und umgekehrt) sind dann:

$$e_0 = C_1$$

$$e_1 = -H_1 C_1 \,.$$

Für einen Vektor $p = [x\,y\,z]^T$ sei

$$[p]_\times = \begin{bmatrix} 0 & -z & y \\ z & 0 & -x \\ -y & x & 0 \end{bmatrix} .$$

Damit lässt sich die Fundamentalmatrix wie folgt schreiben:

$$F = [e_1]_\times H_1$$

Wenn sich die Ansichten in einer kanonischen Konfiguration befinden, dann folgt daraus, dass $e_1 = [e_x\, 0\, 0]^T$. Damit ist die Fundamentalmatrix

$$\hat{F} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix}.$$

Es sei nun eine Fundamentalmatrix in der obigen Form gegeben. Weiterhin sei $\Pi_0 = [I|0]$ und $\Pi_1 = [H_1|-H_1C_1]$, wobei C_1 und H_1 unbekannt seien. Der Epipol e_0 spannt einen Nullraum von $\hat{F}$ auf, also

$$e_0 = [e_x\, 0\, 0]^T$$

für ein unbekanntes e_x. Damit ist $C_1 = [e_x\, 0\, 0]^T$. Um die beiden Bildebenen auszurichten kann man also H_1^{-1}auf $\mathcal{I}_1$ anwenden. Die neue Fundamentalmatrix wäre damit $H_1^T\hat{F}$. Das Ergebnis muss aber wieder bis auf skalare Vielfache $\hat{F}$ sein. Damit muss gelten $H_1^T\hat{F} = \hat{F}$. H_1 muss folglich von folgender Form sein:

$$H_1 = \begin{bmatrix} a & b & c \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Zusammen mit $C_1 = [e_x\, 0\, 0]$ folgt daraus, dass die beiden Ansichten eine kanonische Konfiguration haben.
Es lässt sich also an der Fundamentalmatrix ablesen, ob zwei Ansichten eine kanonische Konfiguration haben. Um zwei Ansichten, für die lediglich die Fundamentalmatrix gegeben ist, in eine kanonische Konfiguration zu bringen, wählt man also H_0 und H_1 so, dass gilt

$$H_1^T F H_0 = \hat{F}$$

und führt dann den Prewarpingschritt mit $\hat{I}_0 = H_0^{-1}\mathcal{I}_0$ und $\hat{I}_1 = H_1^{-1}\mathcal{I}_1$ aus.

5.5.3 Der Prewarp-Schritt

Nun gibt es eine ganze Reihe von Abbildungen, die die obige Bedingung erfüllen. Schließlich kann die neue Projektionsebene im Wesentlichen frei gewählt werden - man gibt einzelnen Projektionsebenen lediglich deshalb den Vorzug, weil bei ihnen das Bild beim Resampling weniger deformiert wird. In diesem Abschnitt wird eine mögliche Methode vorgestellt, um die Projektionsebenen in eine kanonische Konfiguration zu bringen.
Sei d_0 ein beliebiger Fernpunkt aus $\mathcal{I}_0$ - d_0 hat also die Form $[d_x\, d_y\, 0]^T$. d_0 stellt damit die Projektion eines Punktes P dar, so dass $D = P - C_0$ parallel zur Bildebene $\mathcal{I}_0$ liegt. E sei nun der Vektor $C_1 - C_0$. Im allgemeinen spannen nun

D und E eine Epipolarebene auf. Man dreht nun die Bildebene um D, so dass sie auch parallel zu E liegt. Wir machen das, indem wir $\mathcal{I}_0$ um den Winkel θ_0 um d_0 drehen, so dass der Epipol e_0 zu einem Fernpunkt wird. Gesucht ist also eine Rotationsmatrix $R_{\theta_0}^{d_0}$, so dass $\tilde{e}_0 = R_{\theta_0}^{d_0} e_0$ die Form $\tilde{e}_0 = [\tilde{e}_x\, \tilde{e}_y\, 0]$ hat. Die Rotation um d_0 hat die Form

$$R_{\theta_0}^{d_0} = \begin{bmatrix} d_x^2 + (1 - d_x^2)\cos\theta_0 & d_x d_y(1 - \cos\theta_0) & d_y \sin\theta_0 \\ d_x d_y(1 - \cos\theta_0) & d_y^2 + (1 - d_y^2)\cos\theta_0 & -d_x \sin\theta_0 \\ -d_y \sin\theta_0 & d_x \sin\theta_0 & \cos\theta_0 \end{bmatrix} .$$

Für den Drehwinkel θ_0 ergibt sich damit

$$\theta_0 = -\frac{\pi}{2} - \tan^{-1}\left(\frac{d_y e_x - d_x e_y}{e_z}\right) .$$

Damit sind die Epipolarlinien im Bild $\tilde{I}_0 = R_{\theta_0}^{d_0} \mathcal{I}_0$ parallel, da die Richtung der Epipolarlinien nun nur noch von d_0 abhängt. Nun wird das Bild noch um ϕ_0 um die $[0\,0\,1]^T$-Achse rotiert, so dass die Epipolarlinien horizontal liegen:

$$\phi_0 = -\tan^{-1}\frac{\tilde{e}_y}{\tilde{e}_x}$$

$$R_{\phi_0} = \begin{bmatrix} \cos\phi_0 & -\sin\phi_0 & 0 \\ \sin\phi_0 & \cos\phi_0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Dasselbe wird auch auf $\mathcal{I}_1$ angewandt. Die beiden Winkel θ_1 und ϕ_1 werden wie oben berechnet. Wir bezeichnen mit d_1 den Schnittpunkt der durch D und E aufgespannten Epipolarebene mit $\mathcal{I}_1$. Dabei berechnet sich d_1 folgendermassen: Wenn $[x\, y\, z]^T = F d_0$ dann ist $d_1 = [-y\, x\, 0]^T$. Der Kürze halber setzen wir nun $\tilde{H}_0 = (R_{\theta_0}^{d_0} R_{\phi_0})^{-1}$ und $\tilde{H}_1 = (R_{\theta_1}^{d_1} R_{\phi_1})^{-1}$. Die neue Fundamentalmatrix für die gedrehten Bilder hat damit dann die Form

$$\tilde{F} = \tilde{H}_1^T F \tilde{H}_0 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & a \\ 0 & b & c \end{bmatrix} .$$

Zu guter Letzt wird wird das zweite Bild nun noch horzizontal und vertikal mit folgender Matrix skaliert:

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -a & -c \\ 0 & 0 & b \end{bmatrix}$$

Damit hat die Fundamentalmatrix nun die Form

$$\tilde{F} = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix},$$

und die beiden Ansichten sind somit in einer kanonischen Konfiguration mit leicht handhabbaren Scanlines. Insgesamt werden auf die Bilder also folgende Transformationen angewandt:

$$\hat{I}_0 = R_{\phi_0} R_{\theta_0}^{d_0} \mathcal{I}_0 = H_0^{-1} \mathcal{I}_0$$

$$\hat{I}_1 = T_1 R\phi 1 R_{\theta_1}^{d_1} \mathcal{I}_1 = H_1^{-1} \mathcal{I}_1 .$$

Um beide Bildebenen parallel auszurichten würde eine Drehung (nämlich $R_{\theta_i}^{d_i}$) genügen; mit zwei Drehungen kann man aber wie im Fall der kalibrierten Ansichten übereinstimmende Scanlines in den beiden Bildern erhalten.

5.5.4 Interpolation und der Postwarp-Schritt

Wie bei kalibrierten Ansichten kann nun eine Ansicht durch lineare Interpolation zwischen korrespondierenden Punkten gewonnen werden. Obwohl die tatsächlichen Kamerapositionen nicht bekannt waren, können wir sicher sein, dass es sich bei dem erzeugten Bild um eine Ansicht von einem Punkt auf der Verbindungsgeraden zwischen den beiden Projektionszentren C_0 und C_1 handelt. Bei abschließenden Postwarping hat man aber das Problem, dass gar keine Ziel-Kamerakonfiguration bekannt ist. Damit ist auch erst einmal unklar, mit welcher Transformation das interpolierte Bild in die erwünschte Blickrichtung gedreht werden soll.
Die naheliegendste Lösung ist es, direkt zwischen den Matrizen H_0^{-1} und H_1^{-1} elementweise zu interpolieren. Damit handelt man sich aber das Problem ein, dass die interpolierten Bilder z.B. geschert werden und damit keine akzeptable "Kameraaufnahme" mehr darstellen. Stattdessen sollte man die Transformation für den Postwarp so wählen, dass die Ecken des fertigen Bildes die Ecken des ursprünglichen Bildes linear interpolieren.

5.6 Erweiterung auf mehrere Ansichten

Bis hierher haben wir uns auf genau zwei Ansichten beschränkt. Wir sind damit in der Lage, eine beliebige Ansicht mit einem Projektionszentrum auf der Verbindungsstrecke zwischen den beiden Kamerapositionen C_0 und C_1 zu berechnen. Da einem diese Bewegungsfreiheit auf Dauer sicher nicht ausreicht, stellt sich die Frage, was man mit mehr als zwei Ansichten erreichen könnte. Auf alle Fälle

wird man so eine Ansicht zwischen jedem Paar von Ursprungsansichten berechnen können. Man sieht aber leicht ein, dass man mit drei Ansichten bereits das ganze von ihnen umschlossene Dreieck abdecken kann:

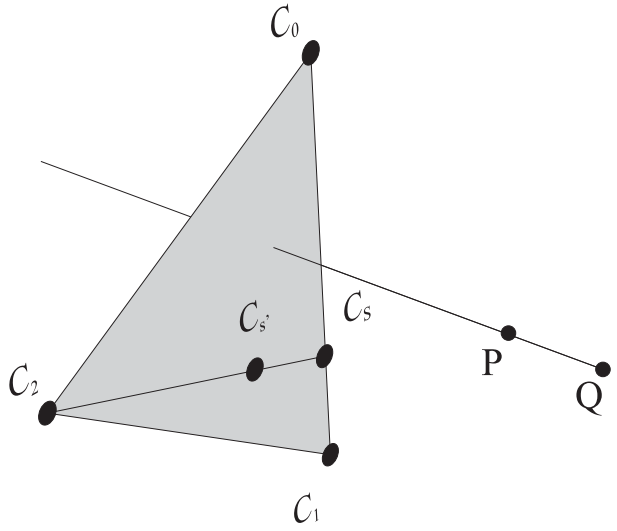


Abbildung 5.6: Bei drei verfügbaren Ansichten interpoliert man zuerst zwischen zweien (C_0 und C_1) und dann zwischen dem Ergebnis daraus und der verbleibenden Ansicht (C_s und C_2). Die erweiterte Monotoniebedingung fordert, dass es *keine* P, Q gibt, so dass die Gerade durch P und Q das Dreieck schneidet.

Wir gehen davon aus, dass die drei Ansichten jeweils paarweise die Monotoniebedingung erfüllen. Seien nun C_0, C_1 und C_2 die optischen Zentren der drei gegebenen Ansichten (s. Abb. 5.6). Dann kann man sich jede beliebige Ansicht C_s auf der Strecke zwischen C_0 und C_1 berechnen. Allerdings kann man nicht davon ausgehen, dass nun auch C_s und C_2 die Monotoniebedingung erfüllen. Man muss an dieser Stelle eine etwas stärkere Bedingung als nur die paarweise Monotonie stellen: Man fordert nun, dass für zwei beliebige, sichtbare Szenenpunkte P und Q die Gerade PQ nicht das Dreieck $\triangle C_0C_1C_2$ schneidet (bei paarweiser Monotonie fordert man nur, dass PQ keine der Strecken $\overline{C_0C_1}$, $\overline{C_1C_2}$ und $\overline{C_2C_0}$ schneidet). Mit dieser stärkeren Voraussetzung kann man nun auch zwischen C_s und C_2 interpolieren und erhält damit eine beliebige Ansicht von irgendeinem Punkt von $\triangle C_0C_1C_2$ aus.

Für den Sprung von der zweiten in die dritte Dimension kommen wir ohne weitere Forderungen aus. Wenn für eine Menge von Ansichten mit den optischen Zentren $V_1 \ldots V_n$ alle Tripel $V_a\, V_b\, V_c$ für alle a, b, c monoton sind, dann kann man für jeden Punkt in der konvexen Hülle $\mathcal{H}$ über die V_n eine Ansicht berechnen: Angenommen, für einen Punkt P im Inneren von $\mathcal{H}$ wäre ein Tripel P, V_a, V_b nicht monoton. Dann gäbe es zwei Punkte S_1, S_2 in der Szene, so dass die Gerade $\overline{S_1S_2}$ das Dreieck $\triangle V_aV_bP$ scheiden würde. Damit würde aber $\overline{S_1S_2}$ aber auf jeden Fall auch

ein Dreieck $\triangle V_c V_d V_e$ auf der Oberfläche von $\mathcal{H}$ schneiden, was nach Voraussetzung nicht sein darf.
Man kann auf diese Weise zwar einen nahezu beliebig großen Teil des Raumes abdecken, aber durch die Forderung, dass jedes Tripel monoton sein muss kann man niemals mehr Punkte der Szene abdecken, als von einem Standpunkt aus beobachtet werden können. Wenn für eine Menge von Ansichten mit den Kamerapositionen W_n nicht alle Tripel monoton sind, dann muss man aus der Produktmenge über den W_n die Elemente heraussuchen, für die wieder alle Tripel der darin enthaltenen Ansichten monoton sind. Für jede derart gültige Teilmenge $W_{n_1} \dots W_{n_m}$ kann dann wieder jeder Ansicht in der konvexen Hülle berechnet werden.

5.7 Ergebnisse

Die folgenden Bilder zeigen jeweils die rektifizierten Ursprungsbilder (links und rechts) und ein interpoliertes Bild jeweils in der Mitte:

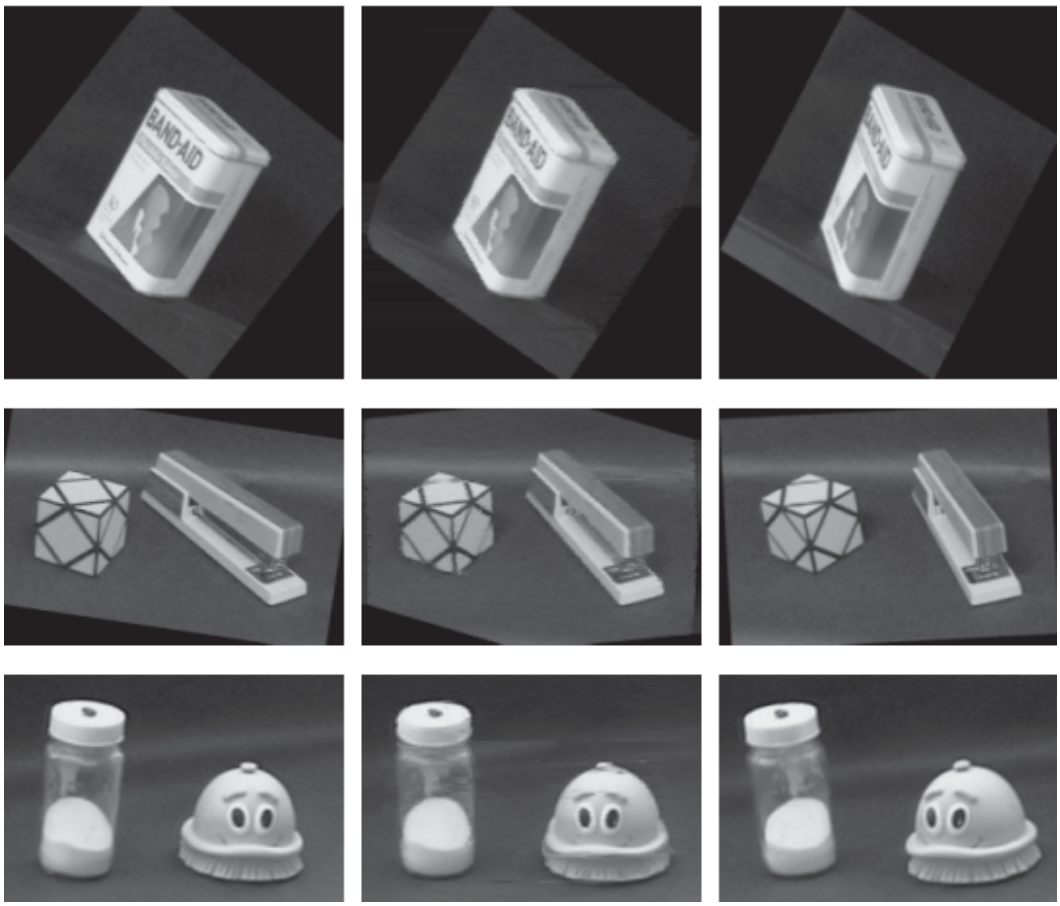


Abbildung 5.7: Beispiele

An einigen Stellen kann man bei den mittleren Bildern Fehlzuordnungen erkennen. So wurde zum Beispiel der Schatten des kleinen Zipfel am Kopf des Männchens links unten im rechtem Bild der Kante zum Hintergrund leicht rechts davon zugeordnet. Daran kann man gut die scanlinebasierte Arbeitsweise des hier verwendeten Matchingverfahrens erkennen. Es wird auch deutlich, dass man mit automatischen Verfahren nur recht kleine Winkel zwischen den Kameraaufnahmen abdecken kann. Damit funktioniert das View Morphing selbst mit einfachen Matchingverfahren bereits recht überzeugend.

5.8 Zusammenfassung

Mit View Morphing kann man aus zwei gegebenen Ansichten V_1 und V_2, die die Monotoniebedingung erfüllen sämtliche Ansichten auf der Strecke zwischen V_1 und V_2 berechnen. Dazu ist müssen entweder die Projektionsmatrizen für beide Ansichten oder die Fundamentalmatrix der Ansichten bekannt sein. Die Berechnung einer Zwischenansicht erfolgt in drei Schritten:

1. Prewarp: Transformation der Ansichten in eine kanonische Konfiguration.
2. Lineare Interpolation aller Bildpunktpaare.
3. Postwarp: Rücktransformation des interpolierten Bildes in seine endgültige Lage.

Für die Interpolation müssen die korrespondierenden Bildpunktpaare bestimmt werden. Dafür kommen manuelle oder automatische Verfahren in Frage. Das automatische Suchen von Korrespondenzen wird durch die Scanlineeigenschaft der Bilder nach dem Prewarpschritt erheblich vereinfacht. Wenn man drei oder mehr Ansichten zur Verfügung hat, kann man auch Ansichten innerhalb eines ganzen Dreiecks oder eines nahezu beliebigen Volumens berechnen.

Literaturverzeichnis

[Seitz & Dyer '96a] Steven M. Seitz und Charles R. Dyer. Toward image-based scene representation using View Morphing. In *Technical Report 1298*. University of Wisconsin-Madison, 1996.

[Seitz & Dyer '96b] Steven M. Seitz und Charles R. Dyer. View Morphing. In *SIGGRAPH'96 Conference Proceedings*, Seiten 21–30, 1996.

[Zisserman '97] Andrew Zisserman. Geometric framework for vision. http://www.dai.ed.ac.uk/CVonline/LOCAL_COPIES/EPSRC_SSAZ/node24.html, 1997.

6 Plenoptic Modeling

Markus L. Noga

Dieses Kapitel behandelt die plenoptische Funktion als Grundlage des Image-based Rendering und stellt das Verfahren des Plenoptic Modeling vor. Dabei baut es im wesentlichen auf dem 1995 in den "Proceedings of SIGGRAPH" erschienenen Artikel "Plenoptic Modeling: An Image-Based Rendering System" von Leonard McMillan und Gary Bishop auf. Insbesondere wurden die Abbildungen 6.1 und 6.8 sowie die in Abbildung 6.2 montierten Aufnahmen dem Artikel [McMillan & Bishop '95] entnommen.

6.1 Grundlagen des Image-Based Rendering

Image-Based Rendering befasst sich mit der Erzeugung neuer Bilder aus gegegebenen Aufnahmen. Diese allgemeine Aussage kann präzisiert werden, indem sie auf eine mathematische Grundlage gestellt wird. Das dazu nötige Fundament ist die plenoptische Funktion.

6.1.1 Die plenoptische Funktion

Die plenoptische Funktion 6.1 beschreibt die Gesamtheit aller Lichtstrahlen in einer Szene. Unter Szene sind hierbei sowohl virtuelle, d.h. allein als Beschreibung im Rechner vorliegende, als auch reale Szenen zu verstehen.

$$p = P(\theta, \phi, \lambda, V_x, V_y, V_z, t) \tag{6.1}$$

Genauer gesagt, gibt die plenoptische Funktion für jeden Punkt (V_x, V_y, V_z) im dreidimensionalen Raum und jeden Zeitpunkt t die Intensität des aus Richtung (θ, ϕ) einfallenden Lichtstrahls der Wellenlänge λ an (siehe Abbildung 6.1). Damit sind die optischen Eigenschaften einer Szene komplett beschrieben - im Gegensatz zur Radiosity-Gleichung oder zur allgemeinen optischen Flussgleichung völlig ohne explizites Wissen über die vorliegenden Geometrien.
Ist die plenoptische Funktion einer Szene bekannt, so kann man beliebige neue Ansichten leicht berechnen - es genügt, die Kameradaten, den gewünschten Zeitpunkt und die nötigen Wellenlängen in die Formel einzusetzen.

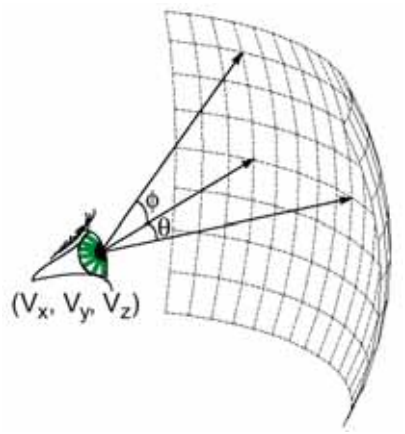


Abbildung 6.1: Anschauliche Deutung der plenoptischen Funktion

6.1.2 Definition des Image-based Rendering

Image-based Rendering befasst sich mit der Rekonstruktion der plenoptischen Funktion aus diskreten Stichproben und der Berechnung neuer Stichproben mit Hilfe der so rekonstruierten Funktion.
In diesem Rahmen lässt sich die Vielzahl unterschiedlicher Verfahren nach folgenden Aspekten einordnen:

- Welche Vereinfachungen der plenoptischen Funktion werden vorgenommen?
  - Gängig ist z.B. die Annahme zeitlicher Invarianz der Szene.
  - Häufig wird die Wellenlänge λ weggelassen (Grauwertbild) oder auf drei diskrete Werte für rot (560 nm), grün (530 nm) und blau (430 nm) beschränkt.
- Welche interne Darstellung der Funktion wird verwendet? Mögliche Darstellungen für die einfallenden Strahlen in einem Punkt sind z.B. sphärische, kubische oder zylindrische oder Umgebungen.
  - Die sphärische Umgebung ist schlecht parametrisierbar. Entweder muss großer Rechenaufwand in Kauf genommen werden (z.B. Merkatorprojektion aus der Kartographie), oder stark uneinheitliche räumliche Auflösung (z.B. Darstellung in Polarkoordinaten, dabei werden die Pole viel genauer aufgelöst als der Äquator).
  - Die kubische Umgebung ist einfach parametrisierbar. Insbesondere im Bereich der Kanten ist die räumliche Auflösung ebenfalls uneinheitlich.
  - Die zylindrische Umgebung ist einfach parametrisierbar und einheitlich, aber unvollständig. Im Gegensatz zu Kugel und Würfel deckt der Zylindermantel nicht alle möglichen Richtungen ab.

- Nach welchem Schema, wenn überhaupt, werden Stichproben entnommen bzw. Aufnahmen gemacht?
- Wie werden die Stichproben in die interne Darstellung überführt?
- Wie werden neue Ansichten aus der internen Darstellung erzeugt?

6.2 Plenoptic Modeling

Plenoptic Modeling ist ein von Leonard McMillan und Gary Bishop entwickeltes Image-Based Rendering-Verfahren. Im Folgenden werde ich Plenoptic Modeling unter den oben erarbeiteten Gesichtspunkten diskutieren.

6.2.1 Vereinfachungen

Plenoptic Modeling geht davon aus, dass die Szene gegenüber der Zeit invariant ist bzw. nur für einen einzigen Zeitpunkt zu modellieren ist. Daher wird der Parameter t der plenoptischen Funktion ignoriert.
Weiterhin beschränkt sich Plenoptic Modeling in der vorgelegten Form auf drei beobachtete Wellenlängen, die dem Sehapparat des menschlichen Auges entsprechen.

6.2.2 Interne Darstellung

Beim Plenoptic Modeling werden die beobachteten Intensitäten des einfallenden Lichts für jeden Augpunkt der Stichprobe getrennt gespeichert. Um eine einheitliche und effiziente Behandlung zu ermöglichen, wird hierzu eine zylindrische Umgebung des Augpunkts verwendet. Diese Darstellung lässt sich in der Ebene abrollen - sie entspricht somit den altbekannten rechteckigen Texturen. Die Einschränkung des Blickfelds auf den Zylindermantel wird dabei in Kauf genommen. Zusätzlich wird zu je zwei Zylindern ein Disparitätsbild gespeichert, das Tiefeninformation enthält.

6.2.3 Stichproben

Die Stichproben werden mittels einer gewöhnlichen Videokamera genommen, die auf einem Stativ befestigt ist. Für jeden Augpunkt wird die Kamera dabei einmal komplett um die vertikale Achse gedreht. Diese Form der Datenerfassung kommt der später verwendeten Zylinderumgebung schon ziemlich nahe.
Die Drehgeschwindigkeit sollte halbwegs gleichmäßig und gegenüber der Bildrate der Kamera klein sein, ist ansonsten aber nicht von Belang. Um die Veränderung von Kameraparametern während der Aufnahme zu vermeiden, ist allerdings der

Autofokus auszuschalten. Aus Rekonstruktionsgründen ist es außerdem vorteilhaft, die Drehachse des Stativs genau auf den Brennpunkt der Kamera auszurichten. Kleine Abweichungen sind hierbei aber durchaus tolerabel.

6.2.4 Überführung in die interne Darstellung

Zwei planare Projektionen ein- und derselben Szene mit gemeinsamem Augpunkt sind durch Homotopie 6.2 verbunden:

$$\begin{bmatrix} u \\ v \\ w \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{6.2}$$

Die Bedingungen, unter denen die Bilder eines einzelnen Schwenks aufgenommen wurden, sind in vielerlei Hinsicht gleich. Insbesondere sind die internen Kameraparameter (z.B. die Brennweite), die Position des Brennpunktes sowie die rotatorische Ausrichtung zweier Achsen identisch. Zwei beliebige Bilder unterscheiden sich also nur durch eine Drehung um die Vertikale.
Daher kann man die Gleichung 6.2 in einen paarunabhängigen Bestandteil S und einen paarabhängigen Bestandteil R_i spalten:

$$\bar{u} = H_i \bar{x} = S^{-1} R_i S \bar{x} \tag{6.3}$$

Der paarabhängige Bestandteil

Der paarabhängige Bestandteil R_i ist bei geeigneter Wahl des Bezugssystems eine Rotationsmatrix:

$$R_i = \begin{bmatrix} \cos\theta_i & 0 & \sin\theta_i \\ 0 & 1 & 0 \\ -\sin\theta_i & 0 & \cos\theta_i \end{bmatrix} \tag{6.4}$$

Gegeben zwei aufeinanderfolgende Bilder, wie kann der Drehwinkel θ_i zwischen den Aufnahmen bestimmt werden? Dabei hilft folgende Überlegung. Die projektiven Verzerrungen der Kamera beschränken sich in starkem Maße auf die Randbereiche der Bilder. Wie in Abbildung 6.2 zu sehen, wirkt sich eine kleine Rotation um die Vertikale im mittleren Drittel des Bildes nahezu ausschließlich als horizontale Translation aus. Diese Verschiebung t_i von einem Bild zum nächsten kann effizient durch eindimensionale Optimierung des Korrelationskoeffizienten der Bildmitten berechnet werden.
Die Verschiebung t_i und der Drehwinkel θ_i stehen über die Brennweite f der Kamera in folgendem Verhältnis:

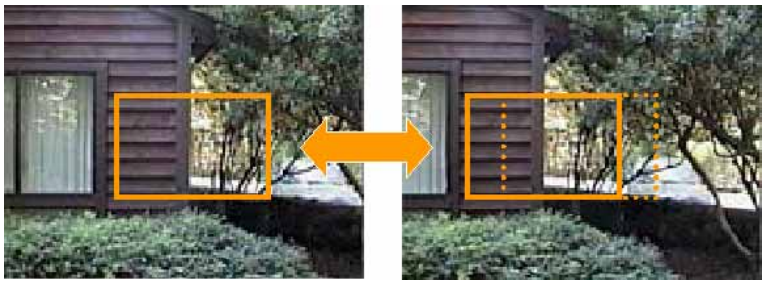


Abbildung 6.2: Rotation wirkt wie Translation

$$\theta_i = \arctan\left(\frac{t_i}{f}\right) \tag{6.5}$$

Wie, wenn nicht durch Messung, erhält man f? Hier hilft die Tatsache, dass die Kamera einmal komplett im Kreis geschwenkt wurde - die Summe aller Drehwinkel muss also 2π ergeben. Also kann f durch numerische Lösung dieser Gleichung mit dem Newton-Verfahren gefunden werden:

$$\sum_{i=1}^{N} \arctan\left(\frac{t_i}{f}\right) = 2\pi \tag{6.6}$$

Damit sind auch alle θ_i bestimmt, und somit die paarabhängigen Bestandteile R_i.

Der paarunabhängige Bestandteil

Die verbleibende paarunabhängige Matrix S kann weiter in zwei Rotationsmatrizen und eine Kameramatrix P zerteilt werden:

$$S = \Omega_x \Omega_Z P \tag{6.7}$$

Dabei hat die Kameramatrix P folgende Gestalt:

$$P = \begin{bmatrix} 1 & \sigma & -C_x \\ 0 & \rho & -C_y \\ 0 & 0 & f \end{bmatrix} \tag{6.8}$$

Die freien Parameter dieser Matrizen werden über ein Mehrvariablen-Optimierungsverfahren bestimmt, das die Korrelation der mittleren Bildbereiche aller aufeinanderfolgenden Bildpaare optimiert. Dazu können z.B. Gradientenanstieg, Sintflutverfahren, Simplex Range Searching oder Simulated Annealing eingesetzt werden. McMillan und Bishop verwenden ein Verfahren nach Powell aus [Press et al. '88].

Projektion auf die Zylinder

Mit den Matrizen S und R_i 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 eine Zylinderhülle oder beliebige andere Geometrien ist dann trivial möglich.

6.2.5 Berechnung der Disparitätsbilder

Die mit einem einzelnen Zylinder erzielbaren Resultate entsprechen etwa denen von Quicktime-VR. Bewegung kann nicht mit korrekter Verdeckung dargestellt werden, einzig Zoomen ist möglich.
Verwendet man hingegen zwei Zylinder, so wird die Darstellung von Kamerabewegungen mit echter Bewegungsparallaxe und echtem räumlichen Tiefeneindruck möglich. Dazu wird in einem Vorverarbeitungsschritt aus jeweils zwei Zylindern Tiefeninformation in Form sogenannter Disparitätsbilder extrahiert.

Relative Position zweier Zylinder

Zunächst ist die relative Position der beiden Zylinder in der Szene festzustellen. Dafür werden manuell korrespondierende Punktepaare auf den Zylindern markiert. Damit ergeben sich nach Abbildung 6.3 Paare von Sehstrahlen im Koordinatensystem der jeweiligen Zylinder.

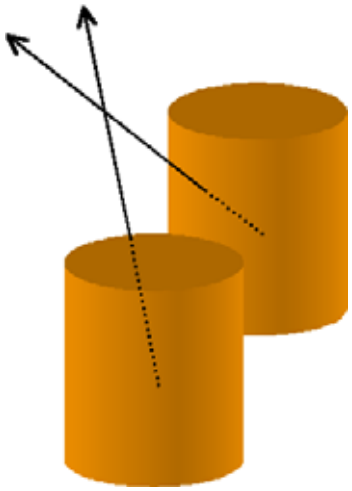


Abbildung 6.3: Positionsbestimmung durch Sehstrahlen

Die unbekannte Transformation des ersten Systems in das zweite kann - bei identischer verwendeter Kamera - auf eine Kombination von Rotation und Translation eingeschränkt werden. Die Parameter der Transformationsmatrix werden

wie oben durch Mehrvariablen-Optimierung bestimmt. Zu optimierender Wert ist dabei die Summe der Abstände der Sehstrahlpaare.
Damit sind die homogenen Projektionsmatrizen bis auf eine freie Variable bestimmt. Diese, der Abstand zwischen den Augpunkten, muss physikalisch gemessen werden - nur von den Bildern her wäre unentscheidbar, ob es sich um Aufnahmen eines Gegenstands oder seines Modells im Maßstab 1:10 handelt.
In der Praxis haben McMillan und Bishop festgestellt, dass 12 oder mehr Punktepaare durchweg gute Ergebnisse liefern. Diese können unter vertretbarem Aufwand von Hand markiert werden. Offen ist noch, wie für die verbleibende Mehrzahl der Punkte ein automatisches Matching geschehen kann. Wie wir gleich sehen werden, unterliegen die möglichen Partner starken geometrischen Einschränkungen.

Epipolargeometrie

Die klassische Epipolargeometrie aus Abbildung 6.4 ist ein alter Bekannter aus dem Maschinensehen. Sie setzt zwei ebene perspektivische Projektionen einer Szene in Bezug.

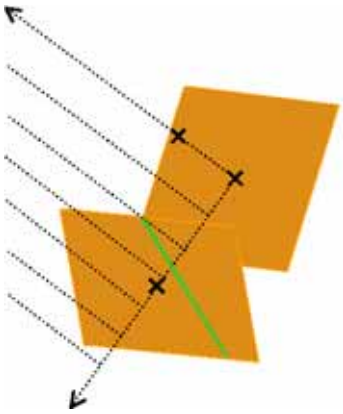


Abbildung 6.4: Klassische ebene Epipolargeometrie

Seien zwei Bildebenen und die dazugehörigen Projektionszentren gegeben. Mit einem Punkt auf einer Bildebene und den beiden Projektionszentren ist eine Ebene eindeutig bestimmt. Ihr Schnitt mit der anderen Bildebene ergibt eine Grade - sie enthält den korrespondierende Punkt auf dieser Ebene (Unter speziellen Umständen wird sogar eine Grade auf eine Grade abgebildet). Der Parameter λ des Punktes auf der Grade wird Disparität genannt und ist ein Maß für die Tiefe des zugehörigen Punkts der Szene.
Für die beim Plenoptic Modeling vorliegende Situation zweier zylindrischer perspektivischer Projektionen (siehe Abbildung 6.5) lässt sich eine ähnliche Einschränkung formulieren.

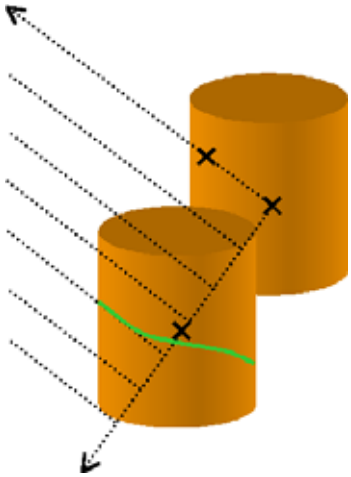


Abbildung 6.5: Zylindrische Epipolargeometrie

Seien zwei Bildzylinder und die dazugehörigen Projektionszentren gegeben. Mit einem Punkt auf einem Zylinder und den beiden Projektionszentren ist eine Ebene eindeutig bestimmt. Ihr Schnitt mit dem anderen Bildzylinder ergibt eine Sinusoide - sie enthält den korrespondierende Punkt auf diesem Zylinder (Unter speziellen Umständen wird sogar eine Grade auf eine Sinusoide abgebildet). Der Parameter λ des Punktes auf der Sinusoide wird Disparität genannt und ist ein Maß für die Tiefe des zugehörigen Punkts der Szene.
Damit beschränkt sich die Suche nach den korrespondierenden Punkten einer Grade auf einem Zylinder auf die zugehörige Sinusoide des anderen Zylinders - eine eindimensionale Aufgabe, die bereits ausgiebig erforscht wurde. Im Folgenden werden einige der Verfahren vorgestellt, die dafür in Frage kommen. Weiterführende Information entnehme man [Faugeras '93].

Matching durch Korrelationsfenster

Dieses Verfahren wurde bereits im vorigen Kapitel über View Morphing behandelt.

Matching durch Relaxation

Die der Relaxation nach Marr-Poggio zugrunde liegende Idee ist einfach: Beginne zunächst mit vielen möglichen Kandidaten für die Partner jedes Punktes. Prüfe diese in immer größer werdenden Umgebungen auf ihre Verträglichkeit mit den Nachbarn, bis nur noch ein Kandidat verbleibt.
Mathematisch gesehen ist dies ein iteriertes Mengensystem. Seien m_i die Punkte einer Graden, n_i die der anderen. Sei $c^k(m_i, n_j)$ die charakteristische Funktion der k-ten Iteration des Mengensystems: sie gibt an, ob Punkt n_j noch als möglicher Match für m_i angesehen wird.

Mit der Grauwert-Intensität $I(x)$ eines Punktes wird das System folgendermaßen initialisiert:

$$c^0(m_i, n_j) \quad = \begin{cases} 1 & \text{falls} \mid I(m_i) - I(n_j) \mid < \delta_0 \\ 0 & \text{sonst} \end{cases} \tag{6.9}$$

Die Iteration erfolgt nach folgendem Schema:

$$c^{k+1}(m_i, n_j) \quad = \begin{cases} 1 & \text{falls} \sum_{i' \in U_m(i)} \sum_{j' \in U_n(j)} c^k(m_{i'}, n_{j'}) \; > \delta_{k+1} \\ 0 & \text{sonst} \end{cases} \tag{6.10}$$

Dabei sind die δ_{k+1} eine streng monoton wachsende Folge von Schwellwerten und $U_m(i)$, $U_n(j)$ Umgebungen eines Punktes auf der jeweiligen Graden.

Matching durch dynamisches Programmieren

Diese Methode betrachtet das Matchingproblem als zweidimensionale Wegsuche (siehe Abbildung 6.6). Gegeben ein Quadrat, auf dessen oberer Kante der erste und auf dessen rechter Kante der zweite Intensitätsverlauf angetragen ist, finde einen Weg von links oben nach rechts unten, der die Intensitätsdifferenz über seinen Verlauf minimiert.

Abbildung 6.6: Grauwertmatching mit dynamischer Programmierung

Hierbei macht man sich zunutze, daß aus optischen Beschränkungen alle gültigen Wege monoton in X- wie in Y-Richtung sind. Nun lässt sich eine optimale Lösung wie folgt aus optimalen Teillösungen zusammensetzen:
Sei (m_0, m) der minimale Pfad von der linken oberen Ecke zu einem gegebenen Punkt. So ist offensichtlich

$$c(m_0, m) = \min_{n \in N(m)} c(m_0, n) + c(n, m) \tag{6.11}$$

wobei $N(m)$ die Menge der unmittelbar links oder über m liegenden Nachbarpunkte bezeichnet. Die Kosten $c(n, m)$ des Pfades zwischen benachbarten Punkten sind dabei beispielsweise als unmittelbare Intensitätsdifferenz festgelegt.
Eine derartige Lösung ist durch dynamische Programmierung effizient berechenbar. Die resultierende Kurve liefert die gesuchten Punktkorrespondenzen.

Speicherformat

Die Entfernung zweier korrespondierender Punkte entlang ihrer eindimensionalen Umgebung (Grade bzw. Sinusoide) wird als Grauwertbild abgespeichert.

6.2.6 Berechnung neuer Ansichten

Seien nun zwei Zylinderumgebungen A, B, ihre Positionen und ihre Disparität gegeben. Gesucht ist die Projektion auf einen neuen Zylinder V mit bekannter Position. Abbildung 6.7 illustriert das geometrische Problem.

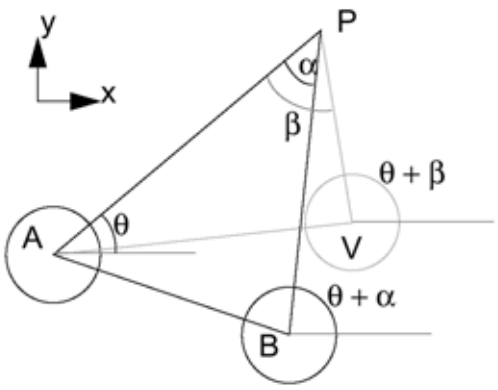


Abbildung 6.7: Geometrie der Berechnung neuer Ansichten

Der Punkt P der Szene erscheint in A unter dem Winkel θ. Aufgrund der bekannten Disparität $\text{disp}(A, B, p) = \alpha$ zwischen A und B erscheint P in B unter dem Winkel $\theta + \alpha$.
Aus den Winkeln und den Positionen von A und B kann die Position des unbekannten P und somit der Winkel $\theta + \beta$, unter dem P in V erscheint, berechnet werden. Dabei ist $\beta = \text{disp}(A, V, p)$ die auf V übertragene Disparität. Praktisch geschieht diese Übertragung der Disparität natürlich, ohne dass der Punkt P der Szene explizit berechnet wird.
Durch spaltenweise Vorberechnung von Teiltermen der Übertragungsfunktion und Vorverarbeitung der Disparitäten kann diese Transformation mit interaktiven Geschwindigkeiten erfolgen (Details entnehme man [McMillan & Bishop '95]). Dabei ist es zusätzlich möglich, die anschließende Projektion auf eine Ebene im selben Schritt auszuführen.

Verdeckung

Es ist möglich, dass viele der Pixel der Ursprungszylinder auf denselben Pixel der neuen Ansicht projiziert werden. Die dabei entstehenden Aliasing-Effekte können prinzipiell nur durch aufwendigere Rekonstruktionsfilter beseitigt werden. Der zusätzlich auftretenden Verdeckungsproblematik kann man allerdings durch einfache Verfahren Herr werden.
Modell ist hier der Painter's Algorithm der Bilderzeugung ([Berg et al. '97]). Durch Zeichnen der Szene von hinten nach vorne wird sichergestellt, dass kein vorderes Objekt durch ein hinteres verdeckt wird. Dieses Verfahren lässt sich auch ohne Kenntnis der Objektgeometrie anwenden.

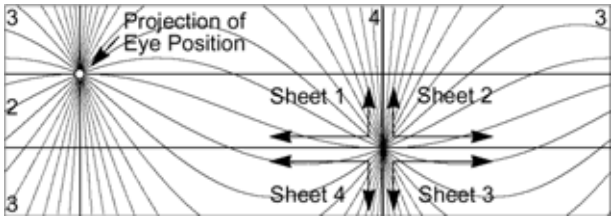


Abbildung 6.8: Zerteilung des Zylindermantels

Dazu wird die Grade vom Augpunkt zum Projektionszentrum des Zylinders mit dem Zylindermantel geschnitten. Die beiden Schnittpunkte teilen den Zylinder in toroidale Streifen (siehe Abbildung 6.8. Um eine Projektion mit korrekter Verdeckung zu erhalten, genügt es bereits, die Streifen vom entfernteren (vom Auge aus gesehen) zum näheren Schnittpunkt hin abzuarbeiten.

6.3 Ergebnisse

Beim Plenoptic Modeling ist es möglich, komplexe natürliche Szenen mit gängiger Hardware aufzunehmen. Aufbauend auf einem Vorverarbeitungsschritt mit geringem menschlichem Zutun können neue Ansichten mit interaktiver Geschwindigkeit berechnet werden. Der Blickwinkel ist dabei nur durch die fehlenden Zylinderkappen eingeschränkt. Der Augpunkt ist dabei - in sinnvollem Abstand zu den usprünglichen Aufnahmepunkten - frei wählbar. Die erzeugten Bilder geben Tiefenverhältnisse korrekt wieder und weisen korrekte Verdeckung und Bewegungsparallaxe auf.
Die Repräsentation der plenoptischen Funktion als Zylindertextur mit Disparität, also als rechteckige Farb- bzw. Grauwert-Textur, ermöglicht die effiziente Speicherung und Übertragung. Mit dem etablierten JPEG-Verfahren sollten sinnvolle Kompressionsraten bei 1:10. Sein wavelet-basierter Nachfolger JPEG 2000 verspricht sogar Raten bis zu 1:100. Ein standardisiertes Format wird von den Autoren aber nicht vorgeschlagen.

Seine Probleme teilt Plenoptic Modeling mit vielen anderen Verfahren. Eine bessere Filterung der Neuprojektion erzielt deutlich bessere Ergebnisse, ist aber zeitlich kostspielig. Schwerwiegender ist, dass Gebiete, über die aufgrund von Verdeckung in den Originalbildern keine Information vorliegt, in neu berechneten Ansichten als schwarze Flächen vorliegen. Hier könnte eventuell Abhilfe geschaffen werden, indem weiterer Zylinder einbezogen werden.

Literaturverzeichnis

[Berg et al. '97] Mark de Berg, Marc van Kreveld, Mark Overmars und Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications.* Springer-Verlag, Heidelberg - Berlin, 1997.

[Faugeras '93] Olivier Faugeras. *Three-Dimensional Computer Vision - A Geometric Viewpoint.* The MIT Press, Cambridge, Massachusetts, 1993.

[McMillan & Bishop '95] Leonard McMillan und Gary Bishop. Plenoptic Modeling. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '95)*, Seiten ??–??, 1995.

[Press et al. '88] W. H. Press, B. P. Flannery, S. A. Teukolsky und W. T. Vetterling. *Numerical Recipes in C.* Cambridge University Press, Cambridge, Massachusetts, 1988.

7 Ein hybrides Verfahren

Stefan Preuß

Abbildung 7.1: Der Campanile-Turm der Universität in Berkeley.

Hier wird ein Verfahren vorgestellt, in dem verschieden Techniken zur Bilderzeugung verschmolzen werden. Der auf der Siggraph 1996 vorgestellte Rundflug um den Campanileturm (vgl. Abbildung 7.1), in dem übergangslos computergenerierte und reale Filmsequenzen zu sehen sind, ist ein Beispiel für die Verknüpfung von bild- und geometriebasierten Verfahren zur Bilderzeugung (zu sehen in [Debevec '97]). Er enstand aus Arbeiten von Paul Debevec (beschrieben in [Debevec et al. '96] und [Debevec et al. '98]), die sich damit beschäftigen, neue Ansichten architektonischer Gebilde auf der Grundlage mehr oder weniger geeigneten Bildmaterials zu generieren.
Bei den üblichen image-based-rendering Verfahren müssen speziell zu diesem Zweck optimierte Bilder vorliegen. Dazu müssen die Aufnahmen i.d.R. von fest definierten Kamerastandpunkten fotografiert sein. Ferner muss meistens eine Vielzahl dieser Aufnahmen für jede mögliche Ansicht vorhanden sein, was zu erheblichen Datenaufkommen führt. Aufgrund der Ausdehnung von Bauwerken ist aber

eine exakte Kameraführung zu deren vollständigen Erfassung kaum zu realisieren. Ein weiteres Problem besteht, wenn neue Ansichten auf der Grundlage von Archivaufnahmen nicht mehr vorhandener Gebäude generieren werden sollen. Dann hat man überhaupt keinen Einfluss auf die Erzeugung der Ausgangsbilder; folglich müssen die zur Bilderzeugung verwendeten Verfahren beliebige Fotografien verarbeiten können.
Das von [Debevec et al. '96] vorgestellte Verfahren bewältigt diese Probleme indem der meist einfache, streng geometrische und symmetrische Aufbau von Bauwerken ausgenutzt wird. Dabei werden auf der Grundlage eines einfachen geometrischen Modells des Gebäudes, das mit Informationen aus diversen Bildern verknüpft wird, überraschend realistisch wirkende neue Ansichten generiert.

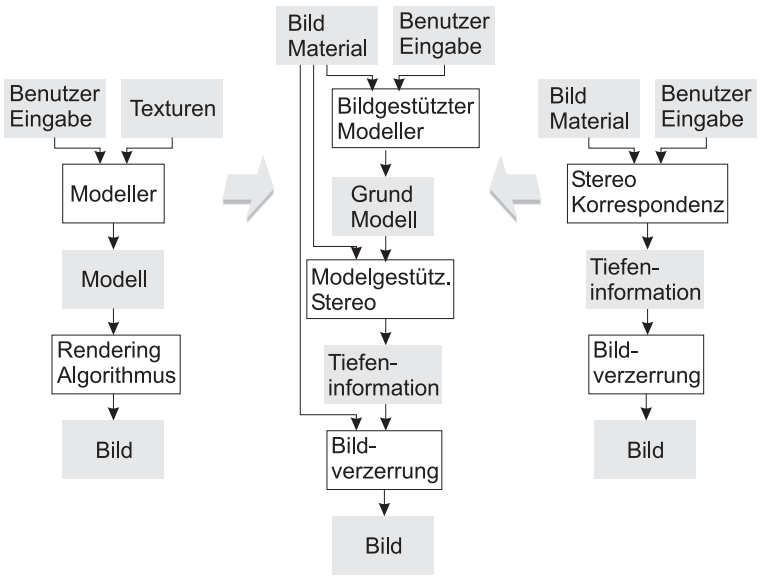


Abbildung 7.2: In dem hier vorgestellten Ansatz werden Techniken der geometriebasierten Bilderzeugung (links) mit denen der bildbasierten Bilderzeugung (rechts) zu einen hybriden Image-based-rendering Verfahren (mitte) verschmolzen.

7.1 Erzeugung der Geometrieinformationen

Grundlage des Verfahrens bildet eine vereinfachte geometrische Darstellung des Gebäudes, welche aus dem Bildmaterial erzeugt wird.

7.1.1 Eingabe der Geometrie

Die Geometrie wird von einem menschlichen Benutzer interaktiv unter Verwendung des Bildmaterials eingegeben. Der Grund ist, dass ein Mensch in der Lage ist, anhand eines Bildes sofort ein grobes geometrisches Modell zu abstrahieren.

Die Erstellung einer relativ simplen Geometrie sowie deren logischer Aufbau verlangt ein großes Maß an vielfältigem Hintergrundwissen, zu dessen Verknüpfung heutige Rechner nicht (oder nur mit immensen Aufwand) in der Lage sind. Bei der genauen Positionierung, Berechnung und Datenhaltung kann der Rechner hingegen gut unterstützen. So werden die Primitive, die später ein Objekt darstellen sollen, von Hand in die aus unterschiedlichen Ansichten gewonnenen Bilder gezeichnet und zugeordnet. Der Rechner unterstützt hierbei den Benutzer bei der genauen Positionierung der eingezeichneten Kanten. Mit Hilfe von Gradientenverfahren kann der Rechner die eingegebenen Kanten mit Genauigkeiten von weniger als einem Pixel an die vom Benutzer erkannten Linien im Bild anpassen, d.h. er errechnet ähnlich einem umgekehrten Anti-Alias-Verfahren den genauen Verlauf der Kante.

Abbildung 7.3: Das Glockengehäuse des Campanile.

Zur Verdeutlichung des Zusammenspiels zwischen den kognitiven Fähigkeiten des Benutzers und den Möglichkeiten des Rechners ein Beispiel: Das Bild 7.3 zeigt das Glockengehäuse des Turmes. Aus dem Bild kann man sofort folgende Grundstruktur ersehen: Auf den Grundmauern - ein Quader, der auf dem Boden steht - schließt sich ein etwas breiterer Sims an: Ein weiterer Quader geringer Höhe, der mittig auf dem ersten liegt. Hierauf folgt (ebenfalls mittig) das Glockengehäuse, welches ungefähr die Form eines Würfels besitzt. Darauf ein zentrierter hohler Quader für das Geländer, in dessen Ecken vier gleiche Spitzen stehen; innerhalb das Turmzimmer mit hohem, spitzem Dach. Dies ergibt elf Grundkörper, deren Parameter zum Großteil, wie in Abbildung 7.4 zu sehen, miteinander verknüpft

sind. Diese werden mit ihren Abhängigkeiten vom Benutzer grob in den verschiedenen Ansichten eingegeben. Vom Rechner werden Sie auf das zugrundeliegende Bild genau eingepasst und zu *einem* dreidimensionalen Modell verknüpft.

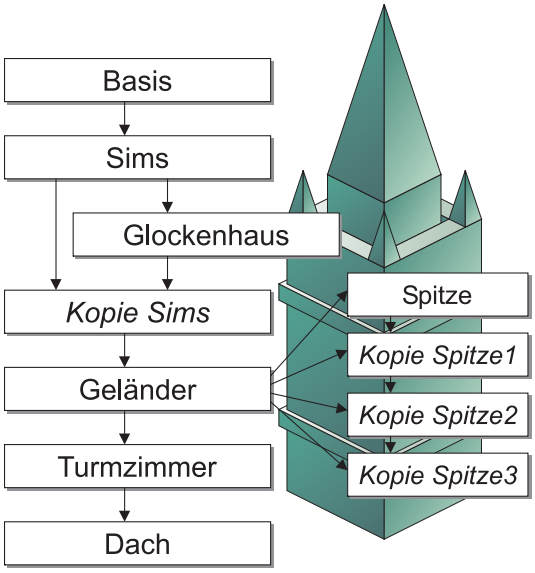


Abbildung 7.4: Logischer Aufbau der Geometriedaten aus dem Bild.

Die Vorteile dieses Verfahrens sind offensichtlich: Da da wir es mit Gebäuden zu tun haben, können unsere Objekte leicht mit geometrischen Grundkörpern einfacher (meist rechtwinkligen) Beziehungen angenähert werden. Das menschliche Einschätzungsvermögen garantiert zudem, dass die Primitiven, trotz eventueller Verzerrungen im Bild, die richtigen Proportionen zueinander haben. Die Modellierung mit geometrischen Primitiven sorgt außerdem für ein einigermaßen minimiertes Datenaufkommen; ein Rechner hätte das Objekt mit Tausenden von Polygonen aproximiert und sicher mit einigen störenden Ungenauigkeiten und Artefakten versehen. In unsrem Beispiel vom Glockengehäuse haben wir nur ungefähr 60 relativ genau definierte Flächen. Den einfachen Aufbau der gesammten Geometrie, die für den Campanile-Film verwendetet wurde, sieht man in Bild 7.5.

7.1.2 Bestimmung der Kameraposition

Das Folgende stellt nun eine Aufgabe dar, bei der die Leistungen eines Rechners wieder gebraucht werden. Es wurde nun ein ungefähres Modell der Gebäude erzeugt. Um aber die Modellwelt sinnvoll mit den Aufnahmen verknüpfen zu können, müssen die Kameraparameter bestimmt werden, unter denen die Aufnahmen erstellt wurden. Das heißt, die Modellwelt muss mit den Aufnahmen zur Deckung gebracht werden.

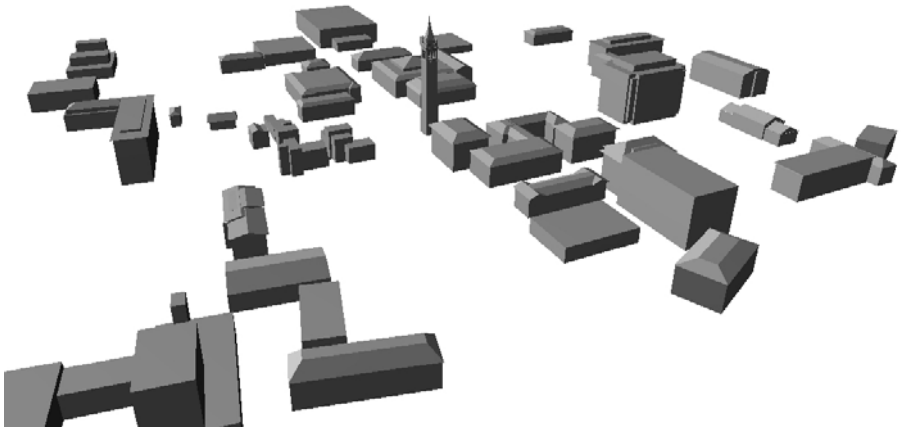


Abbildung 7.5: Die komplette Geometrie, die im Campanile-Film verwendet wurde.

Kameraparameter

Eine Kamera definiert sich über innere und äußere Parameter. Die inneren Parameter beschreiben, wie der Name schon sagt, den inneren Aufbau unserer Modellkamera. Hier sind die Position des Brennpunktes, die Brennweite und die Auflösung des Bildes oder des Aufnahmefeldes (bei CCD-Kameras) zu nennen, aber auch die meist störenden Effekte wie Linsenverzerrungen. Diese innerrn Kameraparameter ermittelt man über genau definierte Referenzbilder, vertraut auf Herstellerangaben oder schätzt sie ungefähr ab. Letzteres ist das einzige, was bei Bildmaterial, dessen Verfahren zur Aufnahme nicht mehr zu rekonstruieren ist (wie etwa alte Archivaufnahmen) zu tun bleibt und erstaunlicherweise befriedigende Ergebnisse liefert, da die genaue Rekonstruktion der auesseren Parameter solche Fehler ausgleicht. Die äußeren Kameraparameter, Position und Drehung der Kamera, lassen sich nun rechnerisch ermitteln. [Debevec et al. '96] minimiert hierbei eine Fehlerfunktion, die die Abweichung zwischen eingegebener Kante und Modellberechnung aus einer initialen Kameraposition beschreibt.

Initialparameter bestimmen

Um erste ungefähre Werte der äußeren Kameraparameter zu bekommen, definiert man sich eine Projektionsebene, die durch den Brennpunkt der Kamera und einer vom Benutzer eingegebenen Kante definiert wird. Zur Erinnerung: die Brennweite wurde vorher schon (mehr oder weniger genau) als innere Parameter ermittelt. Den Normalenvektor $\mathbf{m}$, der so eine Ebene definiert, errechnet sich bezüglich des Kamerakoordinatensystems folgendermaßen: Das Kamerakoordinatensystem besitz seinen Ursprung im Brennpunkt; die Z-Achse definiert die Blickrichtung, die Y-Achse weist zum oberen Teil der Kamera.

$$\mathbf{m} = \begin{pmatrix} PunktA_x \\ PunktA_y \\ Brennweite \end{pmatrix} \times \begin{pmatrix} PunktB_x \\ PunktB_y \\ Brennweite \end{pmatrix}$$

Die Punkte **A** und **B** sind hierbei die Eckpunkte einer eingegebenen Kante auf dem Bildmaterial, welches in diesen Berechnungen als Projektionsebene der Kamera interpretiert wird. Dieser Vektor besitzt folgende Eigenschaften: Da unser Modell die Objekte auf den Bildern wiederspiegeln soll, muß die entsprechende Modellkante in der Projektionsebene liegen. Der in das Kamerakoordinatensystem rotierte Kantenvektor **v** muss senkrecht auf dem Vektor **m** stehen (siehe Abbildung 7.6).

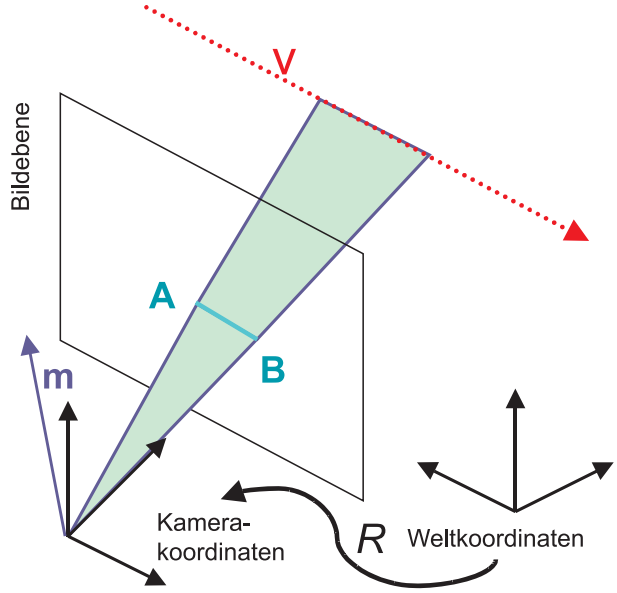


Abbildung 7.6: Bestimmung der Kamerarotation.

$$\mathbf{m}^T R\,\mathbf{v} = 0$$

Aus dieser Bedingung können wir die Rotation ins Kamerasystem, d.h. die Drehung der Kamera berechnen. Da wir mehrere Objektkanten gezeichnet haben können wir aus folgendem überbestimmten Gleichungssystem einen guten Anfangswert ermitteln, indem wir O_1 möglichst klein halten ($O_1 = 0$ bekäme man bei perfekter Orthogonalität).

$$O_1 \quad = \quad \sum_{Kanten\, i} \left(\mathbf{m}^T\, R\, \mathbf{v}_i\right)^2$$

Um die Position der Kamera zu ermitteln, betrachten wir nun die Ortsvektoren zu den Endpunkten einer Objektkante. Auch sie müssen, wenn sie ins Koordinatensystem der Kamera rotiert und auf den Brennpunkt (Ursprung) translatiert werden, natürlich in unserer Projektionsebene der Kante liegen (siehe Abbildung 7.7). Das bedeutet, dass auch diese Vektoren senkrecht auf dem Normalenvektor **m** stehen müssen.

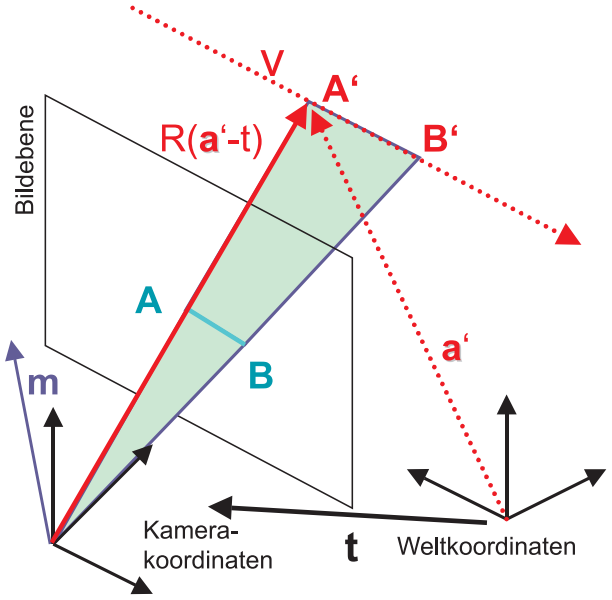


Abbildung 7.7: Bestimmung der Kameraposition.

$$\begin{aligned} \mathbf{m}^T R\,(\mathbf{a}' - \mathbf{t}) &= 0 \\ \mathbf{m}^T R\,(\mathbf{b}' - \mathbf{t}) &= 0 \end{aligned}$$

Da wir auch hierfür wieder mehrere Kanten verwenden können (die natürlich auch nicht perfekt senkrecht auf t stehen), erhalten wir bei möglichst kleinen O_2 einen guten Anfangswert für unsere Kameratranslation t.

$$O_2 \quad = \quad \sum_{Kanten\, i} (\mathbf{a}' - \mathbf{t})^2 + (\mathbf{b}' - \mathbf{t})^2$$

Verfeinerung

Zur weiteren Verfeinerung der äußereren Kameraparameter definiert man jetzt eine Fehlerfunktion, welche die Abweichung von einem Modellbild aus der bisherigen Kameraposition und den vom Benutzer eingezeichneten Kanten definiert und mittels einem mehrdimensionalen Newtonverfahren minimiert wird, wie es zum Beispiel in [Press et al. '92] vorgestellt wird. Hierbei wird der Gradient der Funktion berechnet und iterativ in der Funktion verwendet. Wie in Abbildung 7.8 zu sehen, wird als messbarer Fehler hier die Fläche zwischen der eingezeichneten und der projizierten Modellkante genommen. D.h. es wird über den quadrierten Abstand zwischen den Kanten integriert.

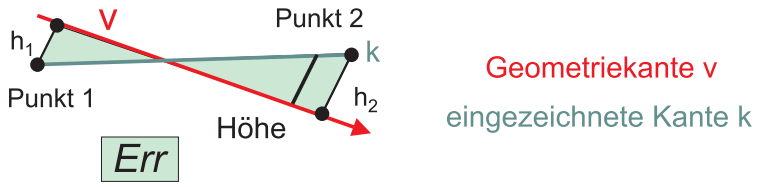


Abbildung 7.8: Bestimmung der Abweichung der berechneten äußeren Initialparameter anhand des Unterschiedes zwischen errechneter Geometriekante und vom Benutzer bestimmter Kante in der Aufnahme.

$$
\begin{aligned}
Err_{Kante\,i} &= \int_0^{Länge\,\mathbf{v}_i} Höhe^2(s)\mathrm{d}s \\
&= \frac{Länge\,\mathbf{v}_i}{3}\left(h_1^2 + h_1 h_2 + h_2^2\right) \\
&= \mathbf{m}^T(A^T B A)\mathbf{m}
\end{aligned}
$$

mit

$$
\begin{aligned}
A &= \begin{pmatrix} PunktA_x & PunktA_y & 1 \\ PunktB_x & PunktB_y & 1 \end{pmatrix} \\
B &= \frac{Länge\,\mathbf{v}_i}{3\left(m_x^2 + m_y^2\right)} \begin{pmatrix} 1 & 0,5 \\ 0,5 & 1 \end{pmatrix}
\end{aligned}
$$

Diese Fehlerfunktionen werden nun für alle Kanten aufsummiert. Diese Summe wird dann über das Newtonverfahren minimiert. Für die Berechnung der Initialrotation hat man nun ein Gleichungssystem mit neun Unbekannten oder 3 Winkeln (je nach Ansatz) zu lösen. Hierbei liefert jede Kante des Objektes eine

Gleichung. Im Anschluss müssen die Ergebnisse noch normiert werden. Für eine erste Kameraposition ist wiederum ein überbestimmtes Gleichungssystem mit diesmal nur drei Unbekannten zu lösen. Auch hierfür liefert jede definierte Kante eine Gleichung. Danach folgt eine Ableitung der Fehlerfunktion mit anschließender Newtoniteration für jede Kante. Eine Iterationstiefe von 10 erwies sich bei [Debevec et al. '96] dabei mehr als ausreichend, so daß die Objekte nicht mehr als ein Pixel von den Fotografien abwichen.

7.2 Texturen bestimmen

7.2.1 Bildmaterial auf Geometrie projizieren

Die nun anhand des Bildmaterials erstellte unterstützende Geometrie wird im nächsten Schritt mit Texturen gefüllt. Hierfür und für folgende Verfeinerungen muss jedoch die Geometrie erst aufgeteilt werden (Beispiel Abbildung 7.9). Teile der Flächen sind auf mehreren, einem oder sogar keinen der Bilder zu sehen, so dass die Flächen der Geometrie gemäss ihrer Anteile auf den jeweiligen Bildern aufgesplittet werden müssen.

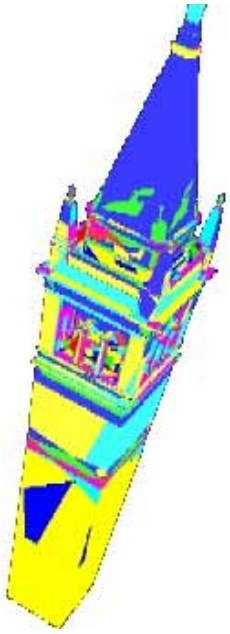


Abbildung 7.9: Aufteilung der Geometrie nach den Texturgrenzen anhand des Bildmaterials.

Diese Teilflächen werden nun mit dem vorhanden Bildmaterial texturiert. Hierbei werden die für jedes Bild errechneten Kamerawerte benutzt, um von eben jenen Positionen das Bildmaterial, ähnlich einem Diaprojektor, auf das Objekt zu projizieren (siehe Abbildung 7.10). Für Flächenstücke, die von mehreren Bildern zu sehen sind, muss eine geeignete Auswahl aus den projizierten Texturen getroffen

werden. (Hierauf wird im folgenden Kapitel näher eingegangen). Für Flächen, die in keinem der Bilder zu sehen sind, wird anhand von Durchschnittsfarben der Texturen ihrer Nachbarflächen über Gouraud-shading eine Verlaufstextur interpoliert. Die Projektion der Bilder läßt sich inzwischen hardwaregestützt berechnen. Hierbei ist zu beachten, daß so eine Hardware Überschattungen nicht beachtet. Es werden alle Flächen im Projektionskegel gefüllt, so dass vorher geklärt werden muss, welche Flächenstücke in dieser Projektion gefüllt werden. Um Speicherplatz (vor allem in der Grafikhardware) zu sparen und die Berechnungen zu beschleunigen, werden schließlich noch die Bilder entlang der Flächen, denen sie als Textur dienen, beschnitten.

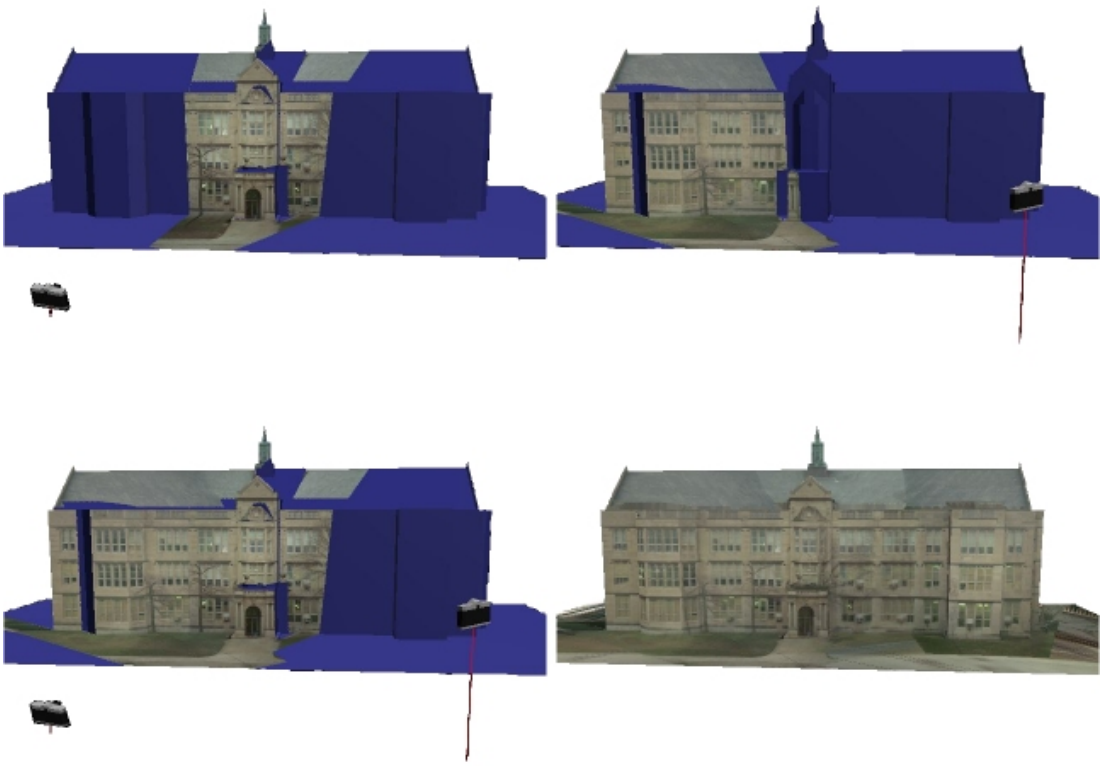


Abbildung 7.10: Die Bilder werden als Texturen auf die Geometrie projiziert. Die beiden oberen Bilder zeigen die Projektion des Bildmaterials aus ihrer berechneten Kameraposition auf die Geometrie. Das untere linke Bild zeigt die Verbindung dieser beiden Projektionen anhand einer Gewichtungsfunktion gemäss des Blinkwinkels. Das untere rechte Bild ist das Ergebnis der Verknüpfung aller zwölf zugrundeliegender Bilder und Füllung der auf den Bildern nicht sichtbaren Flächenstücke.

7.2.2 Verfeinerung der Ansichten

Um nun aus der Geometrie mit teilweise mehrfach texturierten Flächen neue überzeugende Ansichten zu generieren, werden in diesem Verfahren von Paul Debevec Techniken aus image based rendering -Verfahren verwendet.

gewichtete Mehfachtexturierung

Bei Flächen mit mehreren Texturen werden diese anhand der Blickrichtung gewichtet und überblendet. Bei der Erzeugung der Geometrie wurde zu jedem Bild, also Textur, die Kameraposition der Aufnahme bestimmt. Diese Position wird nun mit der Kameraposition der neu zu errechnenden Ansicht verglichen. Je näher der Aufnahmeblickwinkel des Texturbildes nun an dem neuen Blickwinkel auf dieses Flächestück liegt, desto mehr wirkt dieses Bild auf die letzendliche Textur des Flächestückes. Verdeutlicht wird das Prinzip in Abbildung 7.11.

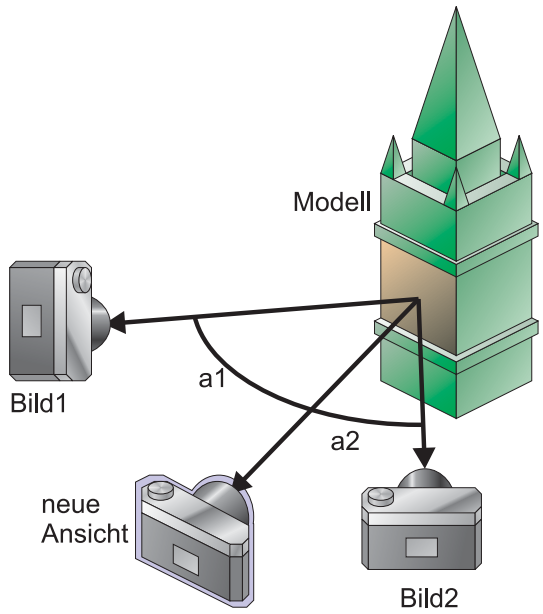


Abbildung 7.11: Die Bilder werden anhand der neuen Blickrichtung im Vergleich zur Aufnahmenblickrichtung gewichtet. Bild 2 wird bei dieser neuen Ansicht, also einen größeren Einfluss auf die Textur dieser Fläche haben als Bild1.

Aber nicht nur die Blickrichtungen werden zur Gewichtung einer Textur in Betracht gezogen. Die Qualität des Bildes selbst wird vorher zumeist von Hand ermittelt und mit eingerechnet. Hier spielen störende Einflüsse wie Bäume, Fußgänger, Lichtreflexe und ähnliches hinein, die natürlich die neuerzeugte Ansicht so wenig, wie möglich beeinflussen sollten. Eigentlich sollte für jedes Pixel einer neuen Ansicht geprüft werden, welches Bild dem aktuellen Sehstrahl am ehesten entspricht. Dies führt laut [Debevec et al. '98] jedoch bei dieser Art von Objekten zu keinem

nennenswerten Qualitätsanstieg, so dass eine flächenweise Berechnung der Gewichtungen ausreichend ist. Die Berechnung der gewichteten Texturen kann auch hier von geeigneter Grafikhardware unterstützt werden. Zu nennen wäre hier die Technik des alpha-blendings, die sich der Möglichkeit der Hardware bedient, veschiedene Duchsichtigkeitswerte darzustellen. Hierbei wird eine Fläche mehrfach in die Geometrie gesetzt. Diese Duplikate werden jeweils mit den verschiedenen Ansichten texturiert und mit Durchsichtigkeitswerten entsprechend der Gewichtung versehen. Meistens werden die Duplikate nicht an die gleiche Position gelegt, sondern ein wenig versetzt übereinandergeschichtet, um Artefakten durch Rechenungenauigkeiten vorzubeugen.

Stereoskopie - Tiefeninformationen

Um die realistische Wirkung neuer Ansichten zu verstärken, werden in den Arbeiten von [Debevec et al. '96] noch Tiefeninformationen der Texturen errechnet. Diese, in Form eines Grauwertbildes, können entweder als Höhenfeld direkt bei der neuen Ansicht berechnet werden oder über Tiefenverzerrungsalgorithmen wie in [Szeliski '94], dargestellt werden. Letztere verschieben die jeweiligen Pixel entlang der Position zur Kamera je nach Grauwert der Tiefeninformation.
Bei der Stereoskopie werden normalerweise für die Berechnungen der Tiefeninformation Bilder verwendet, deren Blickpunkt und -richtung sich nur minimal unterscheidet (-ähnlich der Anordnung der menschlichen Augen). Da sich so die einzelnen Bilder kaum unterscheiden, lassen sich zueinandergehörende Bereiche und deren Verschiebung leichter ermitteln. Um die Wahrscheinlichkeit, korrespondierende Punkte aufzufinden zu erhöhen, werden sie entlang der Epipolarlinien ermittelt. Diese ergeben sich aus dem Schnitt der Epipolarebene -aus dem betrachteten Punkt und den beiden Blickpunkten- mit den Bildebenen.

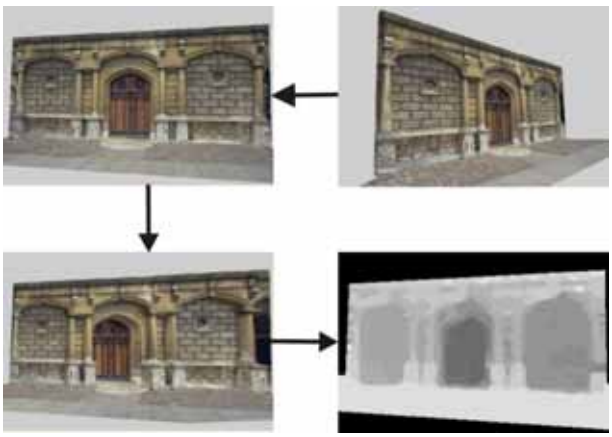


Abbildung 7.12: Die Projektion der Bilder auf eine gleiche Ansicht der Geometrie ermöglicht eine Tiefenbestimmung mittels Verfahren aus der Stereoskopie.

Wie man aber nun an den oberen beiden Bildern der Abbildung 7.12 sieht, weichen die Blickpunkte bei dem hier verwendeten Bildmaterial stark voneinander ab. Um dennoch brauchbare Tiefeninformationen zu erhalten, wird wieder auf die erstellte Geometrie zurückgegriffen. Die Bilder werden auf die Geometrie projiziert und aus dem selben Blickwinkel auf diese verglichen. In 7.12 wird also die rechte obere Ansicht auf die Geometrie wie in der linken oberen Ansicht projiziert. Die so verzerrten Bilder werden bei den hier verwendeten einfachen Objekten sehr ähnlich, wie man im linken oberen Bild und das auf dessen Geometrie verzertte Bild links unten sehen kann.

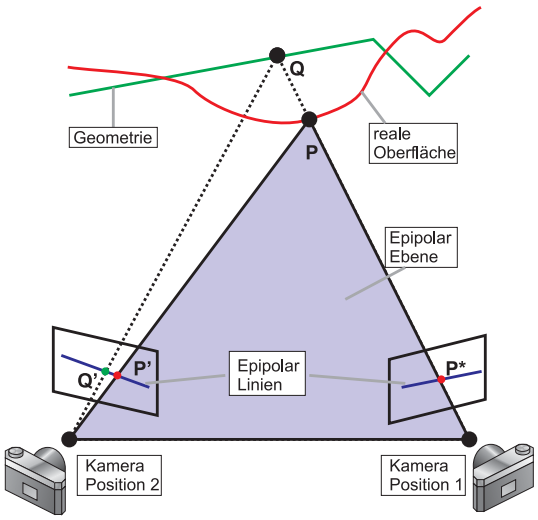


Abbildung 7.13: Der Abstand P und Q auf der Epipolarlinie in der Bildebene ergibt die Tiefe an diesem Punkt Q.

Wie in der Abbildung 7.13 zu sehen ist, findet diese Umprojektion entlang der Epipolarebene statt. So lassen sich die Epipolarlinen ins verzerrte Bild übernehmen und zur Korrespondenzermittlung nutzen. Auf der mitverzerrten Epipolarlinie wird nun der zum Punkt P im Ausgangsbild zugehörige Punkt P' auf dem verzerrten Bild gesucht. Die Höheninformation ergibt sich nun direkt aus dem Abstand des korrespondierenden Punktes P' und dem aus der Geometrie ermittelten Punkt Q.

7.3 Fazit

Dieses Verfahren verwendet image based rendering -Verfahren, um aus nicht speziell organisierten Aufnahmen von relativ einfachen Objekten mit Hilfe von durch einen Benutzer eingegebener einfacher Geometrie neue Ansichten zu erzeugen. Zuerst wird von einem menschlichen Benutzer eine angenäherte Geometrie auf Grundlage der Bilder rechnergestützt erstellt. Aus diesen Eingaben werden die Kamerapositionen ermittelt um eine Verbindung zwischen den Bildmateri-

Abbildung 7.14: Eine neuberechnete Ansicht auf den Turm.

al und der Geometrie zu erstellen. Diese Geometrie wird erst einmal mit aus dem Bilmaterial blickwinkelabhängigen errechneten Texturen versehen. Mit Hilfe der Geometrie und den Bildern werden noch Tiefeninformationen errechnet, die den künstlich erzeugten Ansichten mehr Realismus verleihen. Diese Techniken wurde nun von [Debevec et al. '96] angewendet um den virtuellen Rundflug um den Campanile-Turm (zu sehen unter [Debevec '97]), aus aufgenommenen und berechntetn Filmmaterial zu erstellen. Die künstlich erzeugten Ansichten im Film entstanden aus insgesamt nur zwölf Fotografien. Einige sind in Abbildung 7.15 zu sehen. Die Berechnung neuer Ansichten dauerte laut [Debevec et al. '96] in diesem Projekt je nach Koplexität und Umfang der verwendeten Geometrie und Aufnahmen ein paar Sekunden bis zwei Minuten. So könnte diese Technik demnächst in Echtzeitanwendungen ihren Nutzen finden.

Literaturverzeichnis

[Debevec et al. '96] P.E. Debevec, C.J. Taylor und J. Malik. Modelling and rendering architecture from photographs: a hybrid geometry and image-based

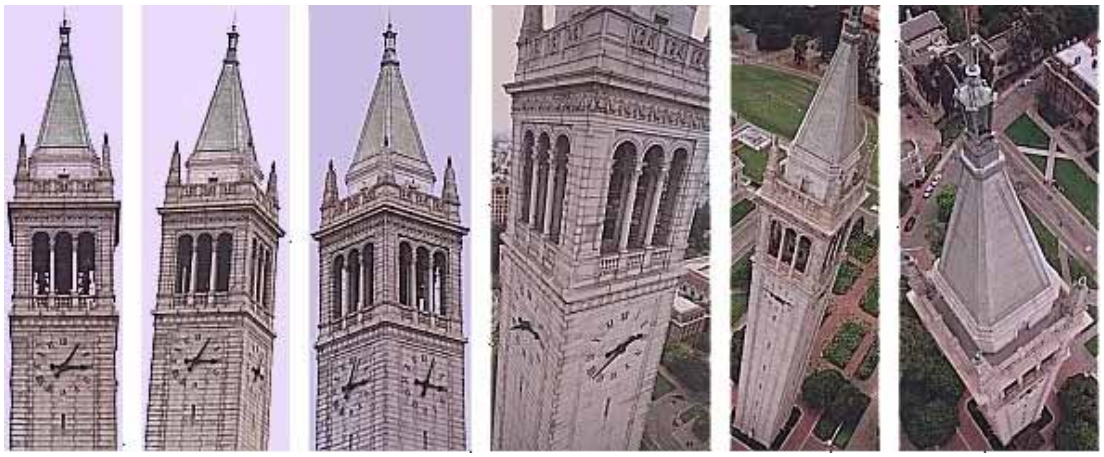


Abbildung 7.15: Verwendeten Aufnahmen für neu berechnete Ansichten.

approach. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '96)*, 1996.

[Debevec et al. '98] P.E. Debevec, Y. Yu und G. Borshukov. Efficient view-dependent image based rendering with Projective texture-mapping. Technical Report CSD-98-1003, University of California, Berkeley, 1998.

[Debevec '97] P.E. Debevec. The campanile tower. http://www.cs.berkeley.edu/~debevec/Campanile, 1997. Gesehen am 16.5.2000.

[Press et al. '92] W.H. Press, B.P. Flannery, S.A. Teukolsky und W.T. Vetterling. *Numerical receipts in C: the Art of Scientific Computing*. Cambridge University Press, 1992.

[Szeliski '94] R. Szeliski. Image mosaicing for tele-reality applications. In *IEEE Workshop on Applications of Computer Vision (WACV'94), Sarasota, Florida*, Seiten 44–53, Dec. 1994.

8 Image-Based Lighting

Armin Müller

Natürliche Bilder zeichnen sich vor allem durch komplexe Licht- und Schatteneffekte aus. Konventionell gerenderte Szenen weisen nur wenige Lichtquellen auf, die durch ein vereinfachtes physikalisches Modell repräsentiert werden. Ich werde im folgenden zeigen, wie sich der image-based Ansatz zur Generierung hochwertiger Beleuchtungseffekte einsetzen lässt. An Stelle der Lichtquellen wird eine Textur mit hohem Dynamikbereich gesetzt. Ziel ist das nahtlose und konsistente Einfügen eines synthetischen Objekts in eine reale Szene. Im wesentlichen basiert dieser Artikel auf [Debevec & Malik '97] und [Debevec '98]. Weitere Informationen und prämierte Filme liegen im Internet unter [Debevec ''].

8.1 Einleitung

Für Kinoeffekte oder Anwendungen in der Architektur ist es immer nützlich, wenn man künstliche Objekte in ein echtes Bild mogeln kann. Es gibt einige Künstler, die dieses diffizile Handwerk beherrschen: ein Bild wird ausgeschnitten und mühsam in ein anderes eingefügt. Realismus erfordert, dass Kameraposition, Lichteinfall, Filmbrillianz, Filmkörnung, Schatten, Glanzlichter und anderes übereinstimmen. Im Falle von synthetischen Objekten müssen zum einen Geometrie und Oberflächeneigenschaften der Umgebung sowie Farbe, Intensität und Positionen der Lichtquellen bestimmt werden, was sich von Hand und auch mit einem 3D-Scanner recht schwierig gestaltet.
Das hier vorgestellte Verfahren nutzt statt synthetisch definierter Lichtquellen einen foto-basierten (image-based) Ansatz. Das Licht ist global vorhanden und erlaubt die korrekte Darstellung von Reflexion, Schatten, Brechung und Streuung. Als Ausstattung ist nur durchschnittliche Hard- und Software erforderlich.
Grundsätzlich wird die Konstruktion in drei Bereiche unterteilt (Abbildung 8.1):

- Die ENTFERNTE SZENE ist der Bildhintergrund. Sie ist so weit weg, dass sie von der synthetischen Szene nicht sichtbar beeinflusst wird, deshalb ist für sie keine weitere Modellierung zu unternehmen. In Einzelfällen, z. B. der spiegelnden Glasfassade eines Hochhauses, wird die ferne Szene doch tangiert und muss deshalb in die nächste Kategorie eingeordnet werden! Weiterhin wollen wir die ferne Szene als globale Lichtquelle mit echten

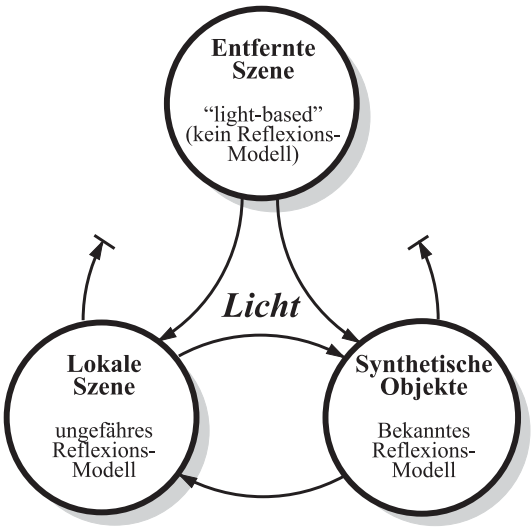


Abbildung 8.1: Die Vorgehensweise. Entnommen aus [Debevec '98]

Leuchtwerten missbrauchen, zwischen aktivem und passivem Bildbereich liegen 6 Größenordnungen an Helligkeit. Dazu mehr in Kapitel 8.3.

- Die LOKALE SZENE ist von der Präsenz der synthetischen Objekte betroffen. Auf sie werden Reflexionen und Schatten der synthetischen Objekte fallen und umgekehrt. Um sie ins Beleuchtungsmodell aufnehmen zu können, müssen Geometrie und Oberflächeneigenschaften bekannt sein, wenigstens näherungsweise (Kapitel 8.2.1). In den meisten Fällen ist die nahe Szene nur eine Ebene auf welcher die Objekte plaziert werden und damit trivial, Bäume oder ähnliches sind hingegen unmöglich zu modellieren.

- Die SYNTHETISCHEN OBJEKTE sind in ihren Eigenschaften durch ihren Entwurf schon vollständig definiert. Form und Material sind keiner Einschränkung unterworfen – auch Glas, Wasser und Leuchtkörper sind erlaubt.

Am Ende erfolgt die Synthese aller drei Teilszenen zu einer einzigen mittels eines raffinierten Differenzverfahrens.

8.2 Die Arbeitsweise des Raytracing

Die Raytracing-Komponente dieses Verfahrens wird mit dem frei erhältlichen Programm *„Radiance“* bewältigt [Larson '].
Raytracing (Strahlverfolgung) ist im Grunde eine möglichst akkurate Simulation eines Teilbereiches der Physik, der Optik. Grundlage des Bildes ist eine geometrisch definierte Szene aus dem Auge und der Bildebene des Betrachters, mehreren Objekten sowie Lichtquellen (Abbildung 8.2). Jede Lichtquelle und jedes Objekt

besitzen eine Grundfarbe, Objekte können auch mit einer Bildprojektion (der Textur, also einer Farbmatrix) bedeckt sein. Die Berechnung (das Rendern) des Bildes geschieht rückwärts, indem man Sehstrahlen vom Auge durch die einzelnen Punkte (Pixel) der Bildebene schickt. Jeder Sehstrahl wird mit den Objekten der Szene geschnitten und erhält die Farbe des getroffenen Objekts, die folglich dem Pixel zugeordnet wird. Schiesst der Sehstrahl ins Leere erhält er die Farbe des Hintergrundes. Aber welche Farbe hat ein Objekt?

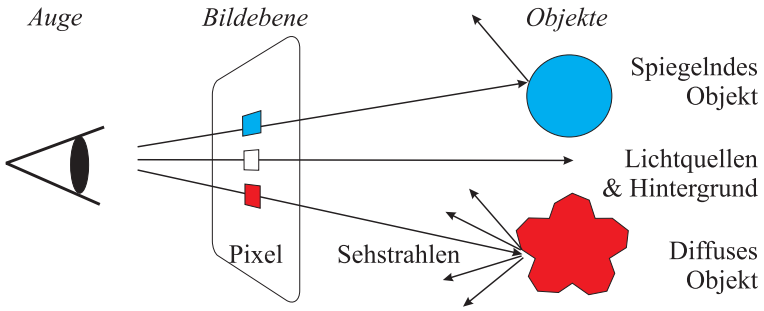


Abbildung 8.2: Raytracing schematisch

8.2.1 Die Reflexionsfunktion

Jede Oberfläche hat ein bestimmtes Reflexionsverhalten. Da die Reflexion vom Auge zur Lichtquelle oder umgekehrt gleich ist, heisst die zugehörige Funktion „Bidirectional Reflectance Distribution Function“ (BRDF). Sie besteht meistens aus einer spiegelnden und einer diffusen Komponente (Absorption, Emission und Brechung seien vernachlässigt). Trifft der Sehstrahl im Winkel α auf eine spiegelnde Oberfläche, wird er im gleichen Winkel α weitergeschickt und die Suche beginnt rekursiv von neuem. Diffuse (rauhe) Oberflächen sammeln Licht aus allen einfallenden Richtungen mit einer charakteristischen Winkelabhängigkeit. Da man mitnichten unendlich viele Strahlen verfolgen kann, muss man die diffuse Reflexion durch ein Bündel stochastisch erzeugter neuer Sehstrahlen nachahmen. Beim herkömmlichen Raytracing wählt man in guter Näherung nur die Strahlen, die auf eine Lichtquelle zeigen. Auf der Radiance Map repräsentiert jeder Pixel eine Lichtquelle, hier muß man Kompromisse eingehen. Wichtig ist die BRDF an dieser Stelle zur Modellierung der Oberflächeneigenschaften der lokalen Szene. Die Textur der lokalen Szene ist unwichtig, da sie durch ein Differenzverfahren nachträglich eingefügt wird. Als Farbe wählt man einen repräsentativen Mittelwert. Interessant sind nun noch Reflexions- und Diffusionsvermögen. Man hat mehrere Möglichkeiten, die unterste wurde bei den Beispielbildern gewählt:

- Fotogrammetrisch messen
- Iterativ schätzen, rendern, vergleichen, verbessern...
- Grob annehmen

8.2.2 Environment Mapping

Jede Raytracing-Szene muss aus Flächen, Oberflächeneigenschaften, Lichtquellen und mehr zusammengestellt werden. Eine gängige Methode mit wenig Modellieraufwand viele Details einzubringen, ist das sogenannte Environment Mapping. Man stellt eine Textur zur Verfügung, die kubisch, zylindrisch oder sphärisch einen Gegenstand umgibt und den Hintergrund ersetzt. Trifft ein Sehstrahl auf kein weiteres Objekt mehr, wird er auf die Environment Map abgebildet und erhält den Farbwert des Schnittpunktes. Environment Maps wurden 1976 von Jim Blinn eingeführt. Die aktive Helligkeit wird von synthetischen (punktförmigen) Lichtquellen geliefert. In Abbildung 8.3 rechts ist der klassische Utah-Teapot mit Environment Map gerendert, links ist eine grob von Hand gezeichnete Environment Map, die als Umgebung der Kanne dient.

Abbildung 8.3: Einfache Environment Map, gerendertes Ergebnis. Entnommen aus [Debevec ']

In der Filmindustrie wurde das Verfahren weiterentwickelt. Die gezeichnete Textur wurde durch echte Bilder ersetzt. Bekannt wurden z. B. „Flight of the Navigator“ (1986) und „Terminator II“ (1991) in Abbildung 8.4. Weil die extraterrestrischen Wesen meist silbrig erscheinen, wurde das Verfahren in Reflection Mapping umbenannt. Es kann nämlich nur spiegeln, nicht Licht und Schatten werfen. Erst vor kurzem wurde die Technik wiederentdeckt für das spiegelnde Raumschiff der Naboo in „Star Wars: Episode I“ (1999).

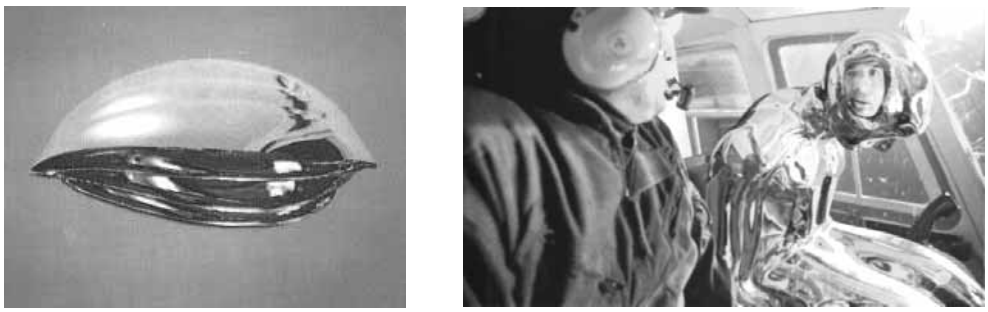


Abbildung 8.4: Der Flug des Navigators, Terminator II. Entnommen aus [Debevec ']

Wir wollen nun die simple Environment Map durch eine mit hohem Dynamikbereich (Kontrast) belichtete Fotoaufnahme aus der Realität ersetzten (HDR =

High Dynamic Range). Diese Aufnahme ist in der Lage, alle synthetischen Lichtquellen zu ersetzten. Deshalb heisst die Environment Map ab hier Radiance Map, sie enthält nicht nur die Farbinformation, sondern auch Helligkeit (Abbildung 8.5).

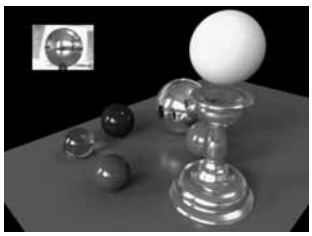


Abbildung 8.5: Mit Radiance Map gerendert. Entnommen aus [Debevec ']

8.3 Fotografie mit hohem Dynamikbereich

Herkömmliche Fotos als Environment Map sind für unseren Zweck zu kontrastarm. Eine Digitale CCD-Kamera liefert einen Kontrast von nur 1:100, eine Kleinbildkamera sogar bis zu 1:10.000, was beim Digitalisieren (Filmscanner mit 3 Kanälen zu je 8 Bit) allerdings wieder verloren geht. In natürlicher Umgebung beträgt der Kontrast aber bis zu 1:100.000, ein einfaches Extrembeispiel ist der Blick aus einem düsteren Raum ins Freie. Diesen Wert erreicht man durch Montage einer Reihe unterschiedlich belichteter Einzelbilder. In diesem Abschnitt wird eine Methode vorgestellt, die selbstkalibrierend ist bis auf einen konstanten Faktor, d. h. es sind keine Fotogrammetrische Messung und keine Kalibrierkurven notwendig.

8.3.1 Empfindlichkeit des Filmes

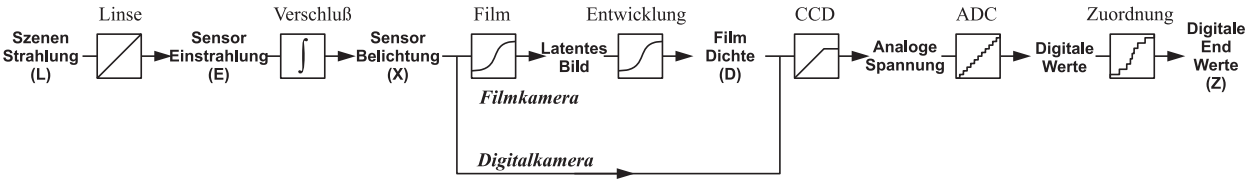


Abbildung 8.6: Der fotografische Prozess. Entnommen aus [Debevec & Malik '97]

Abbildung 8.6 zeigt die Bilderfassungskette für Film- und CCD-Kameras. In jedem Schritt führen Nichtlinearität, Sättigung, Digitalisierung etc. zu Abweichungen der gemessenen Helligkeit von der vorhandenen Helligkeit. Wir betrachten im nächsten Abschnitt vor allem die Einstrahlung E, die Sensorbelichtung X und den resultierenden digitalen Pixelwert Z.

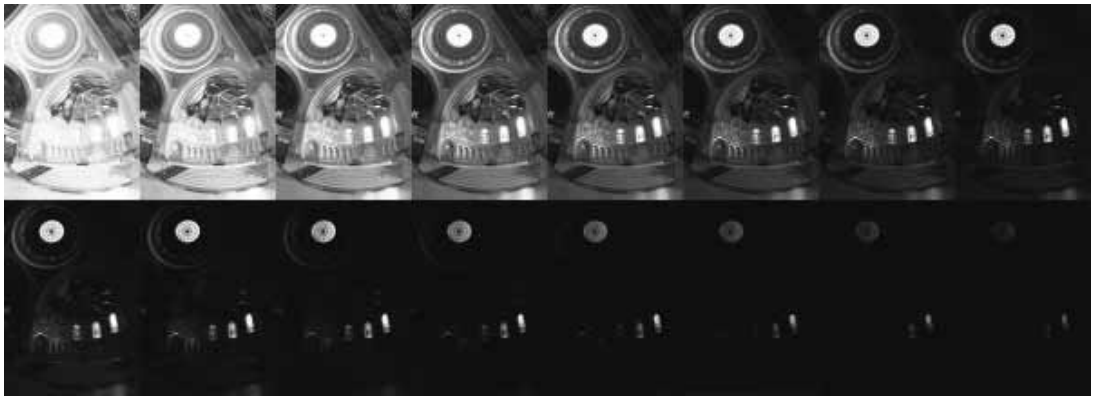


Abbildung 8.7: Belichtungsreihe mit variierender Verschlusszeit. Entnommen aus [Debevec & Malik '97]

8.3.2 Rückgewinnung der Filmempfindlichkeit

Ausgangspunkt des Algorithmus ist eine Reihe digitalisierter Fotos (Abbildung 8.7) mit gleichem Standpunkt und Fokus sowie verschiedenen bekannten Belichtungszeiten Δt_j. Die Blendenöffnung schliesst auf $f/8$ oder kleiner, um eine Abschattung der Ränder zu vermeiden. Es wird angenommen, dass die Szene statisch ist und die Bilder so zügig aufgenommen werden, dass die Ausleuchtung konstant bleibt. Beim Entwickeln und Scannen analoger Filme muss die automatische Qualitätsregelung ausgeschaltet werden.
Bilder einer CCD-Kamera kann man direkt in das Verfahren speisen, hingegen positionieren selbst vollautomatische Filmscanner (z. B. Kodak PhotoCD) den Filmstreifen nicht exakt. Gescannte Bilder müssen vorher durch ein Autokorrelationsverfahren zur Deckung gebracht werden.

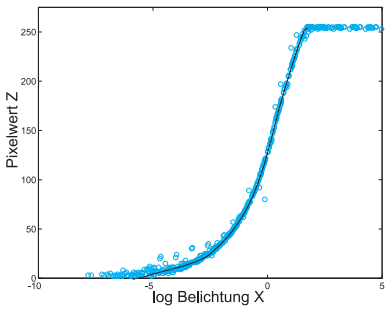


Abbildung 8.8: Charakteristische Kurve einer Kodak DCS460. Entnommen aus [Debevec & Malik '97]

Die Empfindlichkeit eines Filmes auf Belichtung wird die charakteristische Kurve genannt (oder nach dem Entdecker auch Hurter-Driffield Kurve, Abbildung 8.8). Sie ist der Graph der optischen Dichte D (der Schwärzung) des entwickelten Filmes über dem Logarithmus der Belichtung X, der ihr zugrunde liegt. Eigent-

lich kann man die optische Dichte nur (mehr oder minder genau) messen, wir betrachten also gleich den digitalen Endwert Z. Die Belichtung X ist das Produkt der Bestrahlung E des Filmes und der Belichtungszeit Δt (Einheit $\frac{J}{m^2}$). Das Schlüsselkonzept liegt nun darin, dass bei konstantem Produkt $X = E\Delta t$ sich die optische Dichte D nicht ändert („Reziprozität"). Verdoppelt man E und halbiert Δt, bleibt die optische Dichte D unverändert. Filmabzüge und CCD-Chips folgen diesem Gesetz zwischen 10 Sekunden und 1/10.000 Sekunden.
Anstelle der optischen Dichte D betrachten wir den diskreten Pixelwert Z, eine nichtlineare Funktion der Belichtung X: $Z = f(X)$, f sei monoton. Auf jedem Serienbild j der Belichtungszeit Δt_j ist für jeden (örtlichen) Pixel i die Einstrahlung E_i konstant:

$$Z_{ij} = f(E_i \Delta t_j) \tag{8.1}$$

Umkehren von f (da monoton) und Logarithmieren (da positiv) liefert

$$\ln f^{-1}(Z_{ij}) = \ln E_i + \ln \Delta t_j$$

Wir substituieren $g = \ln f^{-1}$ und erhalten die Gleichungsmenge:

$$g(Z_{ij}) = \ln E_i + \ln \Delta t_j \tag{8.2}$$

mit i über alle Pixel N und j über alle Belichtungszeiten (Aufnahmen) P. Die Einstrahlung E_i ist unbekannt, wir fordern g soll glatt und monoton sein. Wir haben nur diskrete Pixelwerte Z, deshalb ist g eine Funktion auf diskretem Definitionsbereich. Wir bestimmen g nach der Methode der kleinsten Quadrate, indem wir $\mathcal{O}$ minimieren:

$$\mathcal{O} = \sum_{i=1}^{N} \sum_{j=1}^{P} \left[g(Z_{ij}) - \ln E_i - \ln \Delta t_j \right]^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} g''(z)^2 \tag{8.3}$$

Der erste Term fordert die Erfüllung des Reziprozitätssatzes 8.1, der zweite sichert die Glätte (und damit Monotonie) der Funktion. Der Skalar λ wiegt die beiden Terme gegeneinander ab und muss je nach Bildrauschanteil angemessen gewählt werden. Die Skalierung der Pixelwerte Z_i ist beliebig, deshalb ordnen wir sie symmetrisch an: $g(Z_{mid}) = 0$, mit $Z_{mid} = \frac{1}{2}(Z_{min} + Z_{max})$.
Ein technisches Problem ist noch, dass selbst unbelichtete Punkte einen gewissen Pixelwert liefern. Überbelichtete Punkte hingegen sind gesättigt und deshalb auch nicht aussagekräftig. Als Lösung führen wir eine einfache Gewichtungsfunktion ein, die im Arbeitsbereich des Fotomediums schwerer wiegt als an den Problemzonen (Abbildung 8.9):

$$w(z) = \begin{cases} z - Z_{min} & \text{wenn} \quad z \leq \frac{1}{2}(Z_{min} + Z_{max}) \\ Z_{max} - z & \text{wenn} \quad z > \frac{1}{2}(Z_{min} + Z_{max}) \end{cases} \tag{8.4}$$

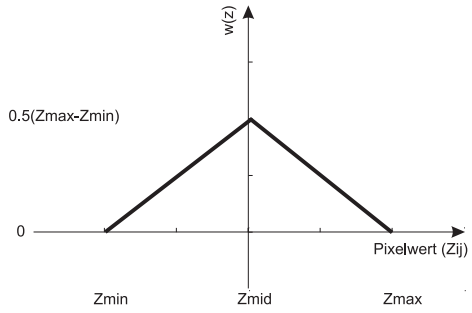


Abbildung 8.9: Graph der Gewichtungsfunktion

Gleichung 8.3 sieht nun so aus:

$$\mathcal{O} = \sum_{i=1}^{N} \sum_{j=1}^{P} \{w(Z_{ij}) [g(Z_{ij}) - \ln E_i - \ln \Delta t_j]\}^2 + \lambda \sum_{z=Z_{min}+1}^{Z_{max}-1} [w(z)g''(z)]^2$$

Schliesslich muss man nicht alle 10^6 Pixel betrachten, das Gleichungssystem ist um Grössenordnungen überbestimmt. Für 256 Pixelwerte und 11 Serienbilder sind 50 Pixel erfahrungsgemäss mehr als genug. Nur muss man bei der Auswahl darauf achten, dass sie örtlich und in der Helligkeit etwa gleich verteilt liegen sowie in Gebieten niedriger Varianz. Abbildung 8.10 zeigt fünf Stichproben (Pixel) aus drei unterschiedlich belichteten Aufnahmen und die daraus rekonstruierte Umkehrfunktion des Films.

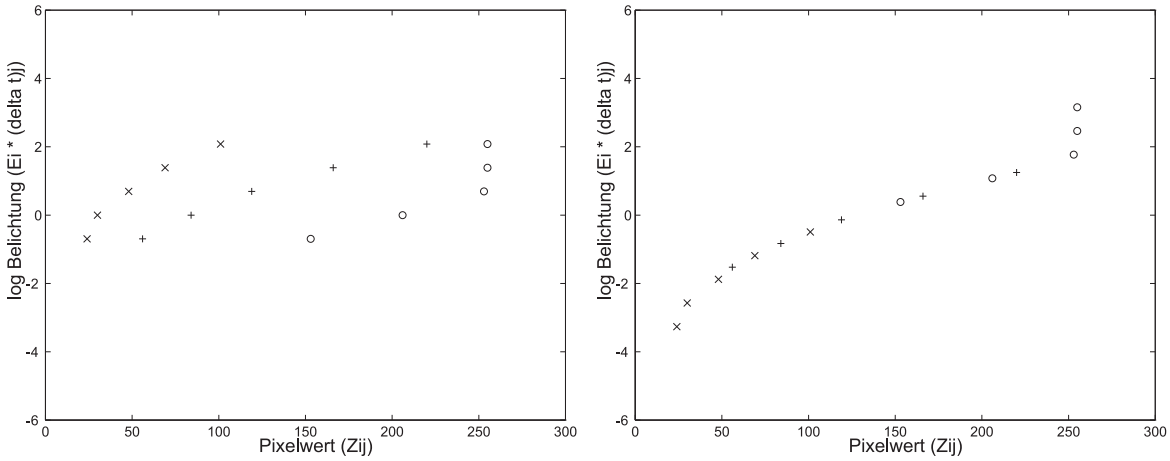


Abbildung 8.10: Rückgewinnung der Einstrahlung aus mehreren Fotos. Entnommen aus [Debevec & Malik '97]

8.3.3 Konstruktion der Radiance Map

Wurde die Abbildung g des fotografischen Prozesses einmal ermittelt, kann man damit schnell Pixelwerte in Einstrahlung umrechnen, sofern man die Belichtungs-

zeiten Δt_j kennt. Wir stellen Formel 8.2 um nach:

$$\ln E_i = g(Z_{ij}) - \ln \Delta t_j$$

Um numerisch stabil zu bleiben und gleichzeitig Helligkeitswerte mit großem Dynamikbereich zu erhalten, lassen wir die Pixel aller Aufnahmen in die Berechnung mit eingehen. Wir verwenden nochmals die Gewichtsfunktion 8.4, die uns vor gesättigten Pixeln schützt:

$$\ln E_i = \frac{\sum_{j=1}^{P} w(Z_{ij})(g(Z_{ij}) - \ln \Delta t_j)}{\sum_{j=1}^{P} w(Z_{ij})}$$

Bis jetzt wurden nur Schwarz-Weiss-Fotos betrachtet. Farbbilder entstehen durch lichtempfindliche Chemikalien bzw. CCD-Elemente, die auf unterschiedliche Wellenlängen mehr oder weniger einheitlich ansprechen. Farbbilder müssen deshalb in ihre Farbkanäle aufgeteilt und separat bearbeitet werden. Zusätzlich ist ein Weissabgleich nötig, da jeder Kanal neu skaliert wird.
Die Belichtungswerte sind relativ, denn sie wurden während der Rekonstruktion auf einer logarithmischen Skala willkürlich um den Mittelwert 0 zentriert. Wenn man absolute Helligkeit erfordert, muss man entweder die Empfindlichkeit des Filmes (ASA) überschlagen oder eine Kalibrierung mittels Aufnahme einer Lichtquelle bekannter Intensität durchführen.
Bemerkenswert ist, dass die Abbildung g für alle Bilder des Prozesses anwendbar ist, und man bei unterschiedlichen Prozessen p mit zugehörigem g_p vergleichbare Ergebnisse erhält, egal wie physikalisch verschieden sie auch sind (es ist also eine Art Farbkalibrierung inbegriffen).
Die fertigen Farbbilder mit hohem Dynamikbereich werden in einem kompakten Gleitkomma-Bildformat abgespeichert, welches jeweils 8 Bit für die Mantissen Rot, Grün, Blau und einen gemeinsamen Exponenten verwendet. Diese Dateien können z. B. von dem Render-System „Radiance“ [Larson '] gelesen werden.

8.3.4 Hoher Dynamikbereich im Vergleich

Abbildung 8.11 zeigt den Unterschied zwischen Radiance Mapping (a-e, mit der HDR-Messung aus Abbildung 8.14) und normalem Environment Mapping (unten). Die Materialien sind Spiegel, rauhes Gold, diffuses Grau, glänzend grünes Plastik und stumpfes orangenes Plastik. (a) und (f) unterscheiden sich wenig, weil das Wiedergabemedium zu schwach ist. Bei (g) bis (j) verwischen sich durch die diffuse Oberfläche die Glanzlichter. Man kann sich die Oberfläche als Gaußschen Weichzeichner vorstellen, der das einfallende Licht mit der Gaußfunktion faltet und danach wieder abstrahlt. Die beschnittene Environment Map wird dabei völlig ausgebügelt. (h), (i) und (j) wurden nachträglich um Faktor 8 aufgehellt, um eine qualitative Beurteilung zu ermöglichen. Es wird jetzt klar, dass ein normales Foto zu wenige Informationen für eine echte optische Simulation liefert.

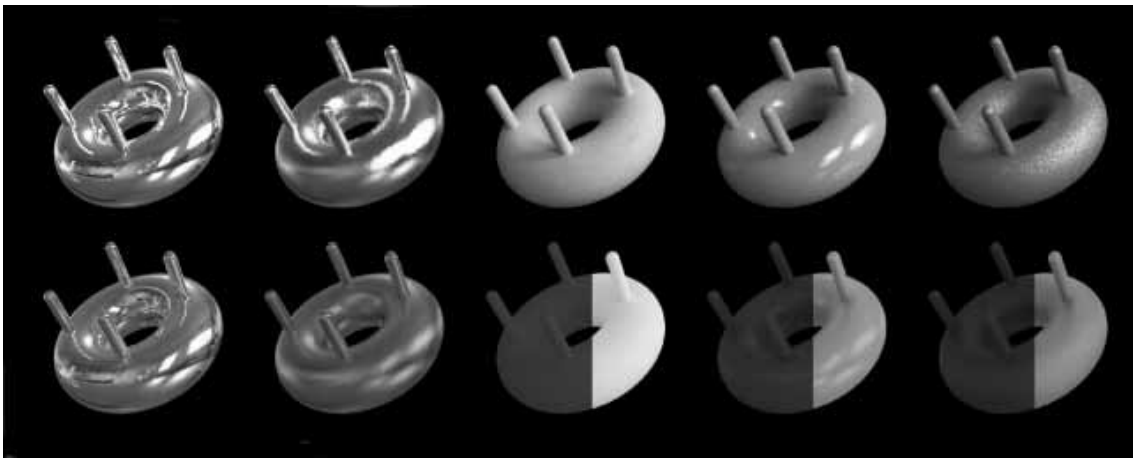


Abbildung 8.11: Synthetisches und natürliches Licht verglichen. oben: a,b,c,d,e unten: f,g,h,i,j. Entnommen aus [Debevec & Malik '97]

Bemerkenswert ist die fleckige Natur von Objekt (e), die auf die stochastische Strahlverfolgung zurückzuführen ist. Das Artefakt kann durch eine höhere Anzahl von Suchstrahlen verringert werden, jedoch steigert dieser Parameter die Berechnungszeit exponentiell.

8.4 Synthese von Raytracing und HDR Fotografie

Für die Produktion des Trickbildes benötigen wir zunächst ein normales Foto des Hintergrundes als Leinwand (Abbildung 8.13) und eine örtliche Helligkeitsverteilung als Ausleuchtung für die synthetische Szene. Danach kann gerendert und alles zusammengefügt werden.

8.4.1 Erfassung einer räumlichen Lichttextur (Radiance Map)

Mit dem Werkzeug aus dem vorhergehenden Kapitel wollen wir nun eine Radiance-(leuchtende Environment-) Map aufnehmen, siehe Abbildung 8.15. Wir plazieren an der Stelle der Szenerie eine spiegelnde Metallkugel (auch „Lichtsonde" genannt) auf einer Kugelschreiberspitze, an der die Objekte erscheinen sollen. Davon nehmen wir wie in Kapitel 8.3 beschrieben eine Reihe Fotos auf und fügen sie zu einem Bild mit hohem Dynamikbereich zusammen. Das Ergebnis zeigt Abbildung 8.14. Allerdings ist das Wiedergabemedium so schwach, daß drei Auszüge notwendig sind. Die quantitativen Angaben in Abbildung 8.14 sind die gemessenen RGB-Werte. Man kann erkennen, dass weder in hellen noch dunklen Bereichen Sättigung eintritt. Die Glühbirne strahlt rötliches Licht aus, das Tageslicht ist eher blau. Für ein geblendetes Auge sehen beide weiss aus, eine optische Täuschung.
Position, Höhe und Durchmesser der Kugel müssen bekannt sein, damit wir später

Abbildung 8.12: *links:* Bestmöglich belichtetes Einzelfoto. Helle und dunkle Zonen sind trotzdem schwarz bzw. weiss gesättigt. Die Fenstersprossen sind ausgeblutet.*rechts:* Falschfarbendarstellung des Bildes mit hohem Dynamikbereich. Jeder Dynamikbereich besitzt seine Details. Entnommen aus [Debevec & Malik '97]

Abbildung 8.13: Aufnahme des Hintergrundes. Entnommen aus [Debevec '98]

die Reprojektion durchführen können. In diesem Beispiel wird einfach ein Papier mit Ausrichtungsmarken unter die Kugel gelegt. Die Kugel befindet sich senkrecht über dem Schnittpunkt der beiden Diagonalen in vorliegender Höhe. Die Geometrie der lokalen Szene, in diesem Fall der Schreibtisch unterhalb des Papiers, berechnet man aus den Ausrichtungsmarken und ihrer bekannten Anordnung.

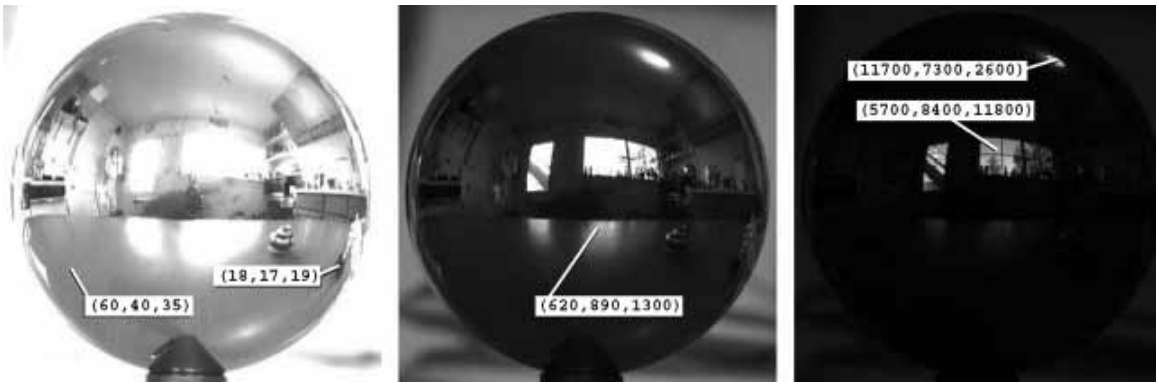


Abbildung 8.14: 360° Aufnahme mit einer Kugel. Entnommen aus [Debevec '98]

Hilfreich ist eine kalibrierte Kamera, da man ohne Nachmessen die Kameraposition, -drehung und -brennweite bestimmen kann.
Die fiktive Reduzierung der Radiance Map auf eine Projektion vom Mittelpunkt der Kugel auf ihre Sphäre ist zulässig, weil die synthetische Szene im Verhältnis zum globalen Beleuchtungsmodell relativ klein ist. Die Lichtstrahlen, die auf die Kugel fallen, sind beim Raytracing ungefähr gleich den Sehstrahlen, die von der synthetischen Szene in den Raum schiessen. An dieser Stelle tritt die Verwandtschaft mit den plenoptischen Verfahren zu Tage. Es werden immer nur die nächstliegenden Sehstrahlen beachtet. Von den sieben Dimensionen der plenoptische Funktion benutzen wir nur zwei Variablen, nämlich $P(\Theta, \phi)$.

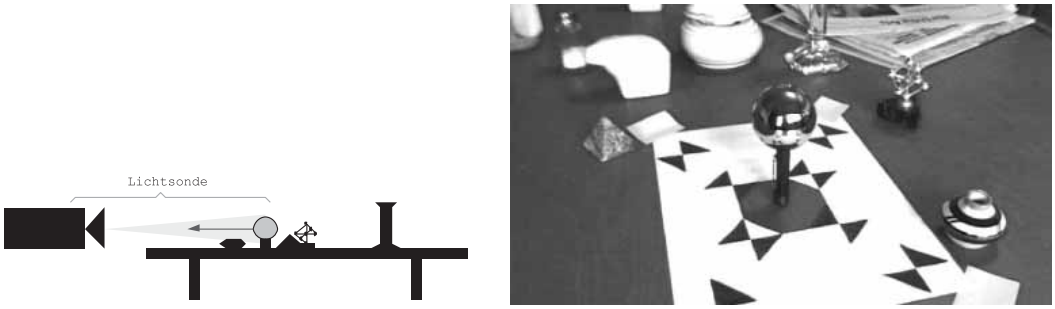


Abbildung 8.15: Aufnahme des Umgebungslichtes. Entnommen aus [Debevec '98]

Grundsätzlich liefert die Kugel eine 360°-Aufnahme. Leider verdeckt sie einen kleinen Teil der Szene, die Kamera spiegelt sich darin, die Ränder sind nicht sehr detailliert und zusätzliche Artefakte wie z. B. der Kugelschatten können erscheinen. Effektiv retuschiert man zwei Aufnahmen der Kugel zusammen, die um 90° versetzt aufgenommen wurden, was ein gutes Ergebnis bringt.
Man kann weiter das Bild der Kugel mit hohem Dynamikbereich auf einen Würfel projizieren wie in Abbildung 8.16 und erhält sechs Flächen, die als Hintergrund-

textur in das Programm „Radiance“ eingebunden werden. Flächen lassen sich zum einen leicht plazieren und sind beim Rendervorgang schneller zu berechnen.

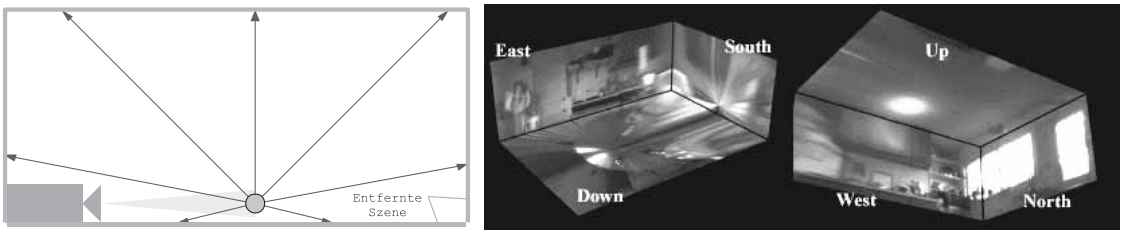


Abbildung 8.16: Die entferne Szene als sechs HDR-Raumtexturen. Entnommen aus [Debevec ’98]

8.4.2 Komposition der Teilbilder

Als nächstes kann man die lokale Szene zusammen mit den synthetischen Objekten und der Radiance Map berechnen lassen. Die Einstellungen der Kameraposition, Drehung und Brennweite sollten selbverständlich mit denen des Hintergrundfotos identisch sind. Das gerenderte Bild kann man an der richtigen Stelle in den Hintergrund (Abbildung 8.13) einfügen und erhält damit Abbildung 8.17 (rechts). Die Beleuchtung stimmt bereits genau überein, aber man sieht sofort, dass die Holzmaserung des Tisches verloren ist.

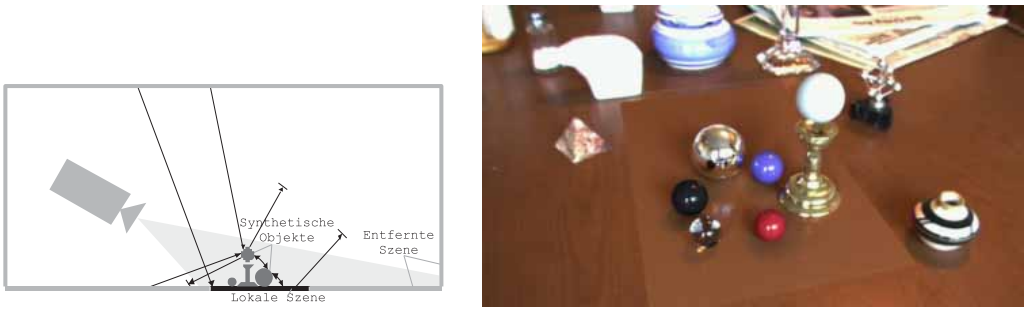


Abbildung 8.17: Kombination des globalen Beleuchtungsmodells. Entnommen aus [Debevec ’98]

Zur Verfeinerung kommt ein Differenzverfahren zum Einsatz, das die verlorenen Details wieder herstellt. Man berechnet die lokale Szene noch einmal, jetzt ohne die synthetischen Objekte, aber wiederum mit der Radiance Map (Abbildung 8.18 links).
Zusätzlich erzeugt man die Objektschnittmaske der synthetischen Szene, die angibt in welchen Teilen des Bildes die synthetischen Objekte im Vordergund stehen (Abbildung 8.18 rechts).
Das nächste Bild ist die Differenz aus Abbildung 8.17 und Abbildung 8.18 (links) mit ausmaskierten synthetischen Objekten. Es gibt an, wie sich die lokale Szene

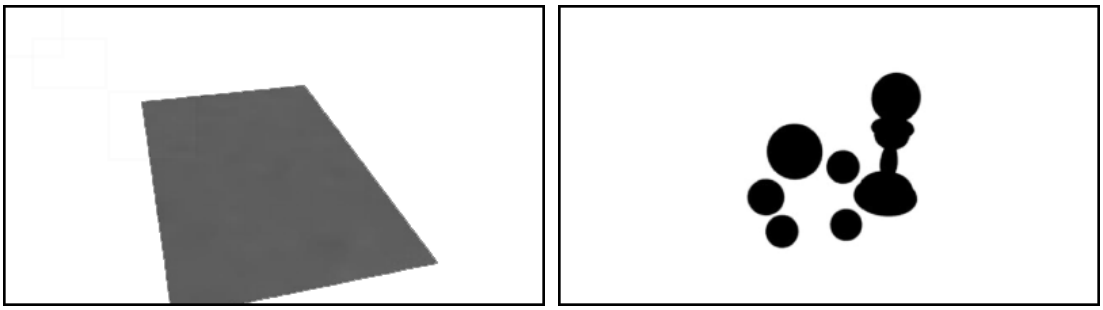


Abbildung 8.18: lokale Szene, Objektschnittmaske. Entnommen aus [Debevec '98]

durch die Anwesenheit der synthetischen Objekte verändert hat. Mittleres grau bedeutet keine Änderung, hell ist Reflexion, dunkel ist Schatten. Die schwarzen Bereiche sind ohne Auswirkung. Man beachte, dass in dem Beispiel durch das Tageslicht andere Merkmale sichtbar werden als durch die Glühbirne.

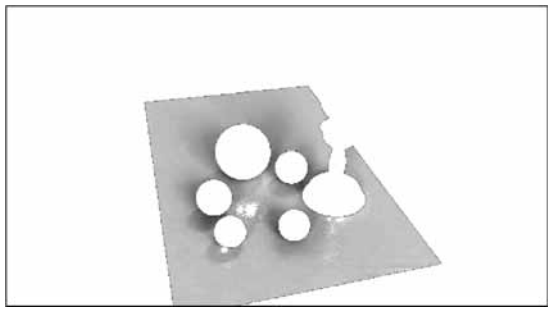


Abbildung 8.19: Differenzbild der lokalen Szene. Entnommen aus [Debevec '98]

Schliesslich sind wir soweit, ein korrektes Bild zu erhalten. Wir kopieren in den Hintergrund 8.13 die Differenz der lokalen Szene 8.19 (also die Änderung des Hintergrundes im sichtbaren Bereich der lokalen Szene) und die ausmaskierten synthetischen Objekte (welche Hintergrund und lokale Szene Überdecken). Das Produkt ist in Abbildung 8.20 dargestellt. Hier ist zu bemerken, dass der Tisch (die lokale Szene) im Spiegelbild der Metallkugel (synthetisch) trotzdem keine Textur aufweist, sie ist leider nur über Raytracing und nicht per Differenzbild berechenbar – eine klare Grenze der Kompositionstechnik. Realistisch ist dafür der Brennglasffekt der Glaskugel auf dem Tisch.

Konsistent sind die Erscheinung der synthetischen Objekte, ihre diffuse und glänzende Schattierung als auch Richtung und Farbe der Schatten. Die Grobkörnigkeit liegt an der begrenzten Anzahl der stochastisch verschickten Strahlenbündel, die diffuses Licht nachahmen.

Abbildung 8.20: Das Resultat. Entnommen aus [Debevec '98]

Literaturverzeichnis

[Debevec '] Paul Debevec. Paul Debevec Homepage. http://www.cs.berkeley.edu/ debevec.

[Debevec & Malik '97] Paul E. Debevec und Jitendra Malik. Recovering High Dynamic Range Radiance Maps from Photographs. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '97)*, 1997.

[Debevec '98] Paul Debevec. Rendering Synthetic Objects into Real Scenes: Bridging Traditional and Image-based Graphics with Global Illumination and High Dynamic Range Photography. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '98)*, 1998.

[Larson '] Gregory Ward Larson. Radiance WWW Server. http://radsite.lbl.gov/radiance/.

9 Der Delta Tree

Jörg Tovar

Der Delta Tree ist eine Datenstruktur, welche ein Objekt durch eine Reihe von Referenzaufnahmen repräsentiert. Die algorithmische Auswertung des Delta Tree erzeugt beliebige Projektionen des Objektes. Die zentrale Idee ist es jeden Oberflächenpixel des Objektes nur einmal zu speichern und auszuwerten. Insgesamt bietet der Delta Tree damit eine effiziente Speicherung und gute Performance bei der Generierung neuer Ansichten.
Als objektorientierter Ansatz ermöglicht der Delta Tree den Aufbau von komplexen Szenen aus mehreren einzelnen Objekten. Damit stellt der Delta Tree eine Erweiterung des Plenoptic Modeling dar.
Dieses Kapitel über den Delta Tree basiert auf der Ausarbeitung „The Delta Tree: An Object-Centered Approach to Image-Based Rendering“ von William J. Dally, Leonard McMillan, Gary Bishop and Henry Fuchs [Dally et al. '96].

9.1 Motivation

Das Plenoptic Modeling [McMillan & Bishop '95] bietet die Möglichkeit sich in einer räumlichen Szene frei zu bewegen. Man erhält Innenansichten von beliebigen Positionen innerhalb des Raumes. Allerdings möchte man auch Objekte darstellen und sich um diese Objekte herum frei bewegen. Solche Außenansichten sind mit dem Plenoptic Modeling nicht möglich. Der Delta Tree erweitert das Plenoptic Modeling in dieser Hinsicht.
Der Delta Tree ist ein objektorientiertes IBR-Verfahren, bei dem das Zusammenstellen komplexer Szenen aus einzelnen Objekten möglich ist. Eine Reihe von Referenzansichten werden in einer Baumstruktur, dem Delta Tree gespeichert. Bei der Durchquerung des Delta Trees lassen sich neue Ansichten des Objektes generieren. Bei den Ansichten ist es möglich beliebige Betrachterpositionen außerhalb der konvexen Hülle des Objektes einzunehmen. Sollen Ansichten innerhalb der konvexen Hülle erzeugt werden, muß das Objekt in mehrere Teilobjekte zerlegt werden. Für die Teilobjekte wird jeweils ein eigener Delta Tree aufgebaut. Diese werden zu einer Szene zusammengesetzt.
Im Delta Tree wird angestrebt, jeden Oberflächenpixel des Objektes nur einmal zu speichern. Dies bedeutet ein minimales Datenvolumen. Bei der Bilderzeugung wird jeder Pixel nur einmal gelesen und reprojiziert. Eine Interpolation zwischen mehreren Werten findet nicht statt. Die in den Knoten des Delta Tree gespei-

cherten Referenzansichten werden deshalb um alle Pixel reduziert, welche bereits bei einem Vorgänger des Baumes dargestellt sind.

9.2 Begriffsdefinitionen

Die **Hüllkugel** oder **Sampling Sphere** ist eine gedachte Umgebungskugel mit festem Radius, welche das gesamte Objekt umschließt. Auf der Oberfläche der Hüllkugel liegen die Kamerapositionen der Referenzaufnahmen.
Ein **Pixel** kennzeichnet die Orts- und Oberflächeninformationen eines Punktes der Objektoberfläche. Die Ortsinformationen der Pixel können zum Beispiel in Form von Disparity Maps gespeichert werden. Als Oberflächeninformationen werden zunächst nur die Farbwerte aus den Referenzaufnahmen verwendet. Weitere Informationen über die Objektoberfläche wie Transparenz, Spiegelung, Glanz u.ä. sind denkbar.
Eine **Ansicht** oder **view** V_P enthält die Bildinformationen des Objektes vom Blickpunkt P. Die Daten einer Ansicht sind in den gespeichert.
Für einen rechteckigen Ausschnitt einer Ansicht verwende ich den Begriff **Bildausschnitt** oder **subview**.

9.3 Sampling, Aufnahmephase

Beim Sampling werden die Referenzansichten des Objektes erzeugt. Die Referenzaufnahmen werden mit Pixelkorrespondenzen in Form von Disparity Maps erweitert und später als Referenzansicht im Delta Tree gespeichert. Als Referenzaufnahmen können sowohl digitale Aufnahmen, als auch am Computer generierte Aufnahmen dienen.
Wie bei jedem IBR-Verfahren stellt sich die Frage was für Referenzansichten man benötigt. Wieviele und welche Referenzaufnahmen müssen für die Reprojektion ausgewählt werden?

9.3.1 Nutzung von Referenzansichten zur Reprojektion

Reprojiziert man eine Ansicht auf einen neuen Blickpunkt, so sind in der Regel nicht alle benötigten Bildpunkte in der Ausgangsansicht vorhanden. Dies ist in Abbildung 9.1 illustriert. Das Bild in der Mitte ist aus der Reprojektion der zu dem linken Bild gehörenden Ansicht entstanden. Es fehlen Informationen von der verdeckten Oberfläche hinter dem Ausguß und von der Rückansicht der Teekanne. Außerdem sind am äußeren Rand der Teekanne nur wenig Bildinformationen in der linken Aufnahme vorhanden. Dies begründet sich dadurch, dass die Strahlen vom Projektionszentrum der Kamera in einem sehr spitzen Winkel auf die Oberfläche der Teekanne fallen. Für die Darstellung von dem neuen Blickpunkt

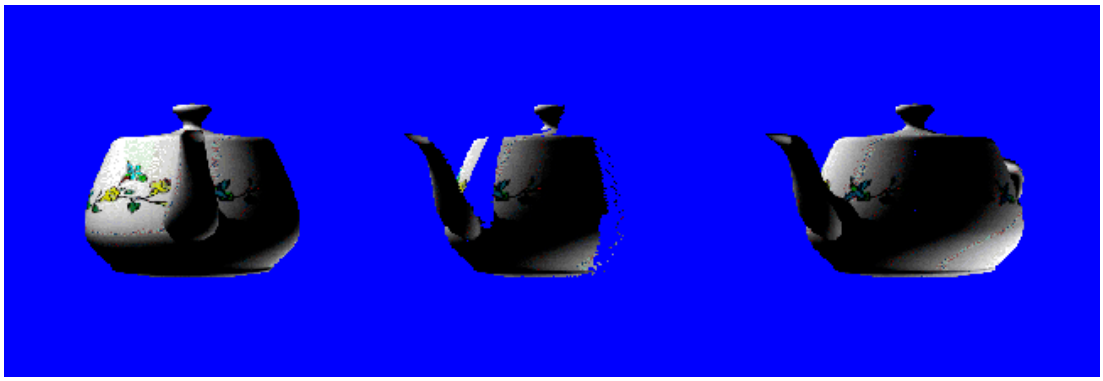


Abbildung 9.1: Reprojektion der linken Aufnahme in der Mitte und Vervollständigung der Reprojektion rechts entnommen aus „The Delta Tree: An ...“ [Dally et al. '96]

aus werden weitere Bildpunkte benötigt. Daraus folgt, dass man für die Reprojektion mehrere Ansichten benötigt. Das rechte Bild in Abbildung 9.1 wurde aus insgesamt 5 Referenzansichten reprojiziert. Die Projektionszentren der zur Reprojektion verwendeten Referenzansichten müssen den neuen Blickpunkt möglichst dicht umgeben. Eine Referenzansicht von der Rückansicht des neuen Blickpunktes liefert keine brauchbaren Informationen. Um eine neue Ansicht eines Objektes im Dreidimensionalen zu reprojizieren, werden im Allgemeinen mindestens drei umliegende Referenzansichten benötigt.

9.3.2 Referenzansichten des Delta Trees

Bei der Erstellung von Referenzaufnahmen für den Delta Tree ist es Vorraussetztung, dass alle Projektionszentren der aufnehmenden Kamera auf einer vorher festgelegten Hüllkugel liegen. Jede Position eines Projektionszentrums auf der Hüllkugel läßt sich eindeutig durch zwei Winkel (θ, ϕ) beschreiben. Die Kenntnis des Radius' ist nicht erforderlich. Nun wird die Oberfläche der Hüllkugel mit einem regelmäßigen Netz von solchen Positionen überzogen. Die benötigte Dichte des Netzes hängt unter anderem von Vertiefungen im Objekt ab. Je nach dem wie tief und schmal diese Vertiefungen sind, muß man die Dichte erhöhen, um alle Oberflächenpixel des Objektes durch mindestens eine Referenzansicht zu erreichen.

9.4 Aufbau des Delta Tree

9.4.1 Struktur des Baumes

Ein Teil der Hüllkugeloberfläche mit den darauf diskret verteilten Kamerapositionen, ist in Abbildung 9.2 links dargestellt. Horizontal und vertikal sind die

Winkel θ und ϕ von 0 bis 45 Grad aufgetragen. Jeder schwarze Punkt kennzeichnet eine Kameraposition auf der Hüllkugeloberfläche, bei der eine Referenzansicht erstellt wurde. Rechts in der Abbildung sieht man einen Teil des zugehörigen Delta Trees, der als Quad Tree aufgebaut ist. Ein kompletter Delta Tree enthält für jede der sechs Grundansichten (oben, unten, vier Seitenansichten) einen solchen Quad Tree. Jeder Knoten im Baum speichert eine Referenzansicht und es ist

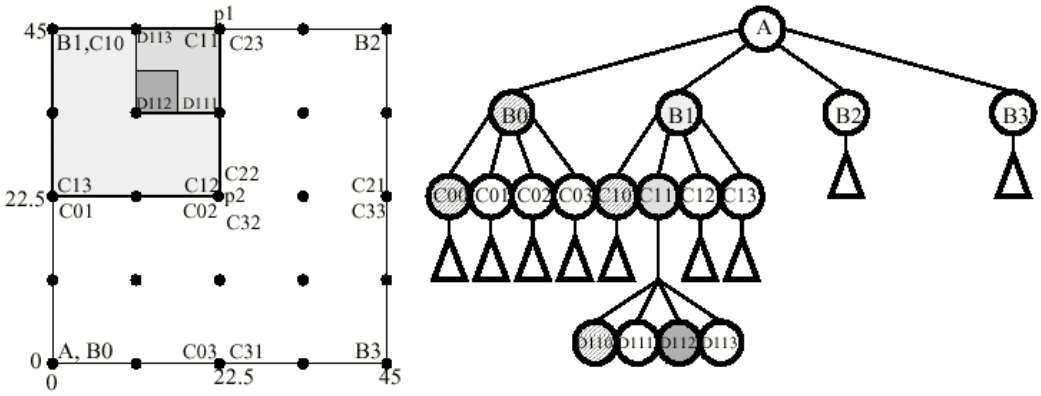


Abbildung 9.2: Ausschnitt der Hüllkugeloberfläche mit zugehörigem Teil des Delta Trees entnommen aus „The Delta Tree: An ...“ [Dally et al. '96]

ihm eine Region auf der Hüllkugeloberfläche zugeordnet. Im Beispiel aus Abbildung 9.2 speichert der Wurzelknoten A die Referenzansicht an der Stelle (0,0) und es ist ihm die gesamte abgebildete quadratische Region von (0,0) bis (45,45) zugeordnet. Jeder der vier Nachfolgeknoten in dem Baum beinhaltet jeweils eine Referenzansicht von einer Regionsecke des Vaterknotens. Als zugeordnete Region erhält er ein Viertel der Region des Vaterknotens. Dies setzt sich bis zu den Blättern des Baumes fort. So sind zum Beispiel die Regionen des Knotens D112 und dessen Vorgänger grau gekennzeichnet.

9.4.2 Vermeidung von Redundanz

Zur Reduktion der Anzahl der zu speichernden Pixel, werden in jedem Knoten alle Pixel der Referenzansicht weggelassen, welche durch Vorgängerknoten im Baum bereits dargestellt wurden. Somit speichert jeder Knoten (abgesehen von der Wurzel) nur noch einen Teil einer Referenzansicht. Manche Knoten sind ganz leer, da der Vaterknoten dieselbe Referenzansicht bereits enthält. Diese Knoten, zum Beispiel B0, C00, ... sind im Baum aus Abbildung 9.2 schraffiert. Werden Pixel einer Referenzansicht in mehreren Knoten benötigt, teilen sich die Knoten diese Pixel. Das heißt die Pixel sind nur einmal im Speicher abgelegt und es wird von mehreren Knoten darauf verwiesen. Dieses Teilen von Pixeln einer Referenzansicht tritt in den folgenden Situationen auf:

- Einer der vier Knotennachfolger teilt die gesamte Referenzansicht mit dem aktuellen Knoten. Im Beispiel sind dies die schraffierten Knoten in dem Baum.

- Referenzansichten am äußeren Rand der Wurzelregion werden von verschiedenen Quad Trees geteilt.

- Mehrere Knoten, welche im Baum Cousins sind, greifen gemeinsam auf Pixel einer Referenzansicht zu. Dies ist zum Beispiel in Abbildung 9.2 bei den Punkten $P1$ und $P2$ zu erkennen. Dort teilen sich die Knoten C11 und C23 beziehungsweise C12, C22, C02 und C32 jeweils die Pixel einer Referenzansicht.

Die Abbildung 9.3 zeigt für das obige Beispiel die Teil-Referenzansichten des Quad Trees bis zur Tiefe zwei. Dabei befindet sich die Teilansicht jeweils an der Ecke der zugehörigen Region. Links unten ist also die gesamte Referenzansicht des Wurzelknotens A zu sehen. In der Ecke links oben befindet sich die Teilansicht des Knotens B1, rechts oben vom Knoten B2 und so weiter. Man erkennt sehr gut, dass die Knoten weiter unten im Baum wesentlich weniger Pixel speichern müssen als die Knoten oben im Baum. Knoten A enthält 34.000 Objektpixel, hingegen speichert ein Knoten der Tiefe zwei durchschnittlich nur noch 3.800 Objektpixel.

9.5 Resampling, Wiedergabephase

Beim Resampling wird der Delta Tee von der Wurzel bis zu den Blättern durchlaufen und dabei eine neue Ansicht des Objektes generiert.

9.5.1 Resampling mit Blickpunkt auf der Hüllkugel

Eine Ansicht von einem Blickpunkt P auf der Hüllkugel erhält man, indem man den Quad Tree von der Wurzel bis zu dem Blatt durchläuft, in dessen Region P liegt. Zusätzlich besucht man noch alle Geschwister des Blattknotens. In jedem Knoten wird die gespeicherte Teilansicht auf den neuen Blickpunkt P reprojiziert und mit Hilfe eines z-Buffer mit den reprojizierten Ansichten der Vorgänger zusammengesetzt. Den Weg auf der Hüllkugeloberfläche illustriert die Abbildung 9.4 für einen Punkt P, der in der Region von D112 liegt. Dabei werden die Knoten A, B1, C11, D111, D112, D113 im Baum besucht und deren Teilansichten zur Generierung der neuen Ansicht verwendet. Der Besuch der Geschwister auf Blattebene gewährleistet, dass die Zielansicht mit Hilfe der vier Referenzansichten auf der Blattebene erzeugt wird. Die vier Projektionszentren dieser Referenzansichten liegen P am nächsten und auf der Hüllkugeloberfläche liegt P in der konvexen Hülle der Projektionszentren.

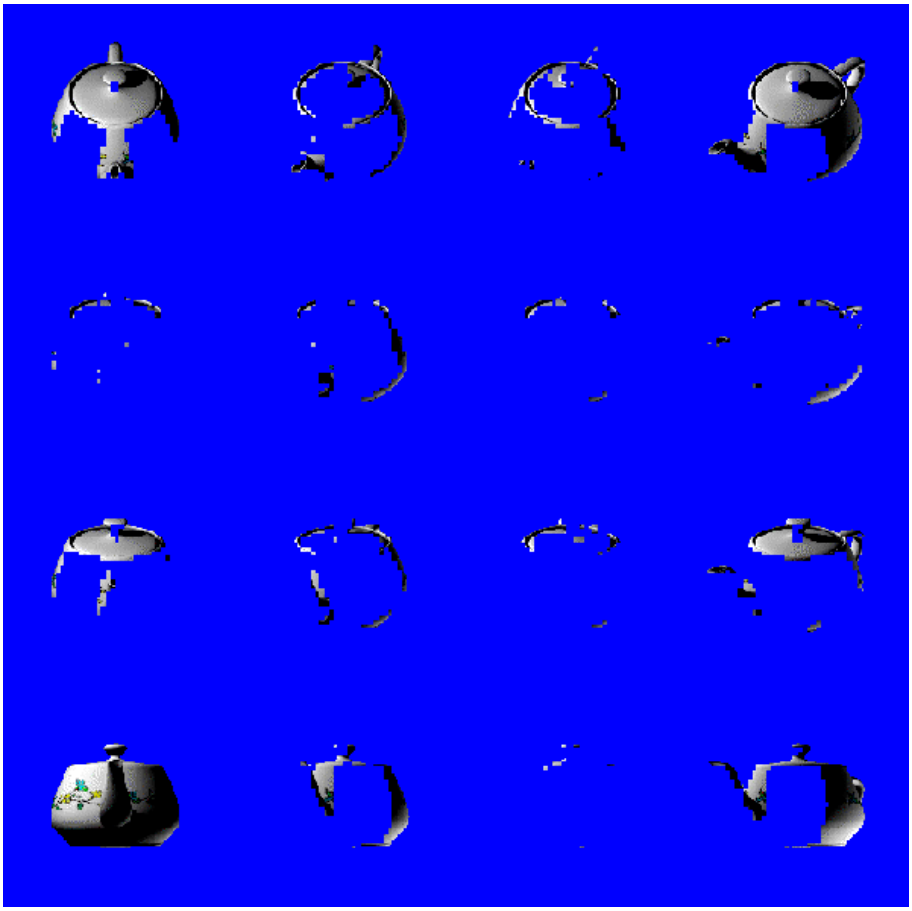


Abbildung 9.3: Teilreferenzansichten des Delta Tree-Ausschnittes entnommen aus „The Delta Tree: An ...“ [Dally et al. '96]

9.5.2 Resampling mit beliebigen Blickpunkt

Wird eine beliebige Betrachterposition innerhalb oder außerhalb der Hüllkugel gewählt, muß ein ganzer Bereich zur Generierung der neuen Ansicht verwendet werden. Für einen Blickpunkt r außerhalb der Hüllkugel und einen Blickpunkt s innerhalb der Hüllkugel zeigt dies Abbildung 9.5. Dabei ist die zur Generierung benötigte Menge auf der Hüllkugel fett markiert. Die neue Ansicht wird durch den folgenden rekursiven Algorithmus in kleinere Bildausschnitte aufgeteilt und erzeugt.

Generiere (Knoten K, Blickpunkt P, Ansicht V)
Anteil der Teil-Referenzansicht von K in V finden
Anteil durch P reprojizieren und mit V kombinieren
falls K Nachfolger hat
V in Bildausschnitte entlang Regionsgrenzen teilen

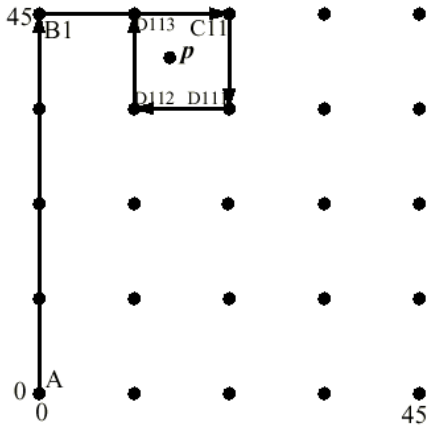


Abbildung 9.4: Weg beim Durchqueren des Delta Trees entnommen aus „The Delta Tree: An ...“ [Dally et al. '96]

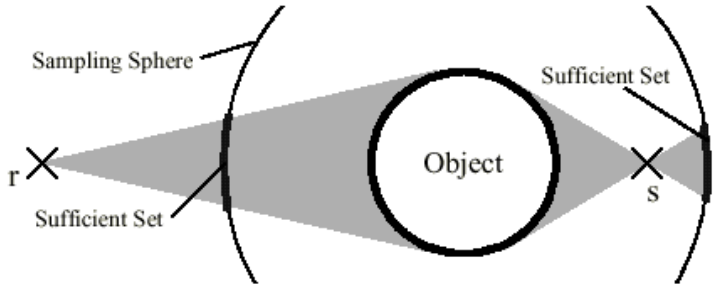


Abbildung 9.5: Finden der benötigeten Bereiche auf der Hüllkugel entnommen aus „The Delta Tree: An ...“ [Dally et al. '96]

für jeden Bildausschnitt B
Auswählen des Nachfolgers bezüglich B
Generiere (Nachfolger, P, B)

Im Wesentlichen werden die Regionsgrenzen des Delta Tree durch den Blickpunkt P auf die Bildebene V projiziert und die Bildebene an diesen Stellen aufgeteilt. Dies ist in Abbildung 9.6 dargestellt.
Bei der weiteren Bearbeitung werden die so entstandenen Bildausschnitte mit der entsprechenden Region auf der Hüllkugel erzeugt. In Abbildung 9.6 wird der Bildausschnitt V_1 der Bildebene V durch die Region R_1 erzeugt und mit Hilfe der Region R_2 wird der Bildausschnitt V_2 generiert. Die Reprojektion auf den neuen Blickpunkt erfolgt mit dem Plenoptic Modeling für Ebenen.

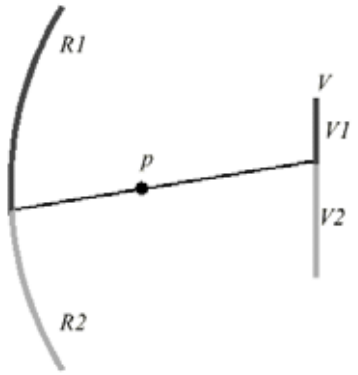


Abbildung 9.6: Projektion der Regionsgrenzen auf die Bildebene entnommen aus „The Delta Tree: An ...“ [Dally et al. '96]

Die Abbildung 9.7 zeigt eine Generationsfolge einer Ansicht. In der ersten Zeile sieht man die verwendeten Teilreferenzansichten, welche man bei der Durchquerung des Delta Tree auffindet. In der zweiten Zeile sind die Teilreferenzansichten auf den neuen Blickpunkt reprojiziert. In der dritten Zeile erkennt man wie sich die neue Ansicht durch Kombination der darüber liegenden reprojizierten Teilansichten aufbaut.

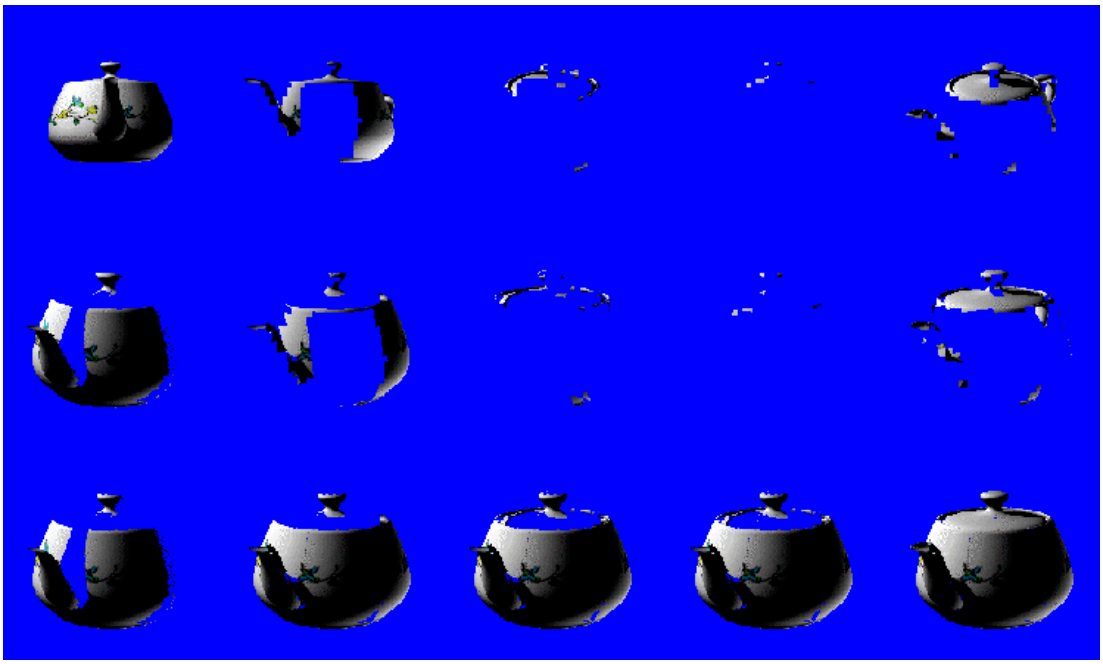


Abbildung 9.7: Generieren einer neuen Ansicht – verwendete Teilansichen, reprojezierte Teilansichten, Aufbau der neuen Ansicht entnommen aus „The Delta Tree: An ...“ [Dally et al. '96]

9.6 Implementierung des Delta Tree

9.6.1 Übergang von Pixel auf Patches

Die Teilansichten in dem Delta Tree sind dünn besetzte Pixelmatrizen. Die Pixel werden zum Teil von mehreren Knoten geteilt und müssen einzeln zugreifbar sein. Um auf die einzelnen Pixel effizient zuzugreifen, müßte jeder mit zusätzlichen Verwaltungsinformationen versehen werden. Diese würden das insgesamt benötigte Speichervolumen inakzeptabel aufblähen. Eine einfache Speicherung in Form eines Arrays ist zu inflexibel und schränkt die algorithmische Auswertung stark ein. Deshalb wurde vom Pixel weg auf Patches der Größe 8x8 Pixel übergegangen. Dadurch ist der Anteil der Verwaltungsinformationen wesentlich kleiner und eine an das JPEG-Verfahren angelehnte Komprimierung sorgt für eine weitere Reduzierung des Speichervolumens.

9.6.2 Redundanz im Delta Tree, Komprimierung

Das Ziel jeden Pixel der Objektoberfläche nur einmal zu speichern ist nicht erreicht. Durch den Übergang von Pixel auf Patches gelangt Pixel-Redundanz in den Delta Tree. Sind zwei Patches abgesehen von einem Pixel gleich, muß man beide Patches speichern. Weitere Redundanzen entstehen durch Verfeinerungen innerhalb der Referenzansichten. Wird zum Beispiel ein Oberflächenabschnitt des Objektes in der Wurzel nur sehr ungenau dargestellt, verfeinern nachfolgende Knoten diesen Abschnitt durch zusätzliche Bildinformationen in ihren Referenzansichten. Letztlich ist auch die Aufspürung bereits dargestellter Bildinformationen recht schwierig. Es werden genaue Ortsinformationen benötigt und teilweise kann man die Redundanzen nur unzureichend aufspüren.
Dally, McMillan, Bishop und Fuchs [Dally et al. '96] erzielten Speicherkompressionen der Ursprungsdaten von circa 9:1 nur durch Weglassen bereits dargestellter Bildinformation und gemeinsame Nutzung von Referenzansichten.

9.6.3 Reprojektion auf den neuen Blickpunkt

Die Reprojektionen erfolgen auf Basis des Plenoptic Modeling für Ebenen. Damit die Reprojektion durchgeführt werden kann, muß in den Ansichten geometrische Information z. B. in Form von Disparity-Maps vorhanden sein. Diese werden in der Samplingphase über die Suche nach Pixelkorrespondenzen für die Referenzaufnahmen erstellt.
Dazu seien zwei Referenzaufnahmen von den Blickpunkten A und B gegeben und gesucht sei eine Referenzansicht mit geometrischen Daten vom Blickpunkt A. Zuerst werden die Disparity-Werte von der Aufnahme A bezüglich der Aufnahme von B berechnet. Diese werden zusammen mit der Aufnahme A und den Koordinaten von A und B als Referenzansicht V_A gespeichert.

Bei der Reprojektion wird mit dem Punkt P über eine perspektivische Abbildung aus der Ansicht V_A die Ansicht V_P berechnet. Genaueres findet man in der Ausarbeitung von McMillan und Bishop [McMillan & Bishop '95].

9.7 Schlußbetrachtung

Wie bei anderen IBR-Verfahren sind die Darstellungsmöglichkeiten des Objektes durch das eingesetzte Bildmaterial beschränkt. Bei der vorhandenen Implementierung wurden vorerst keine visuellen Effekte wie zum Beispiel Glanz, Belichtung und Schatten realisiert. Diese können genau wie bei herkömmlich erzeugten Bildern nachträglich in die generierten Bilder eingebracht werden. Ebenso haben die Autoren von [Dally et al. '96] vorerst keine Transparenz innerhalb der Objektoberfläche berücksichtigt. Dies ist mit einer entsprechenden Erweiterung des Verfahrens ebenfalls zu lösen.
Problematisch sind die vorausgesetzten geometrischen Daten über das Objekt. Hat man Bilder von realen Objekten sind genaue geometrische Daten für jeden einzelnen Punkt auf der Objektoberfläche in den allermeisten Fällen nicht vorhanden. Eine andere Möglichkeit implizit an diese Daten zu kommen, ist die Suche nach Pixelkorrespondenzen in zwei Aufnahmen von verschiedenen Blickpunkten auf das Objekt. Für dieses Problem gibt es aber bisher keine zufriedenstellende algorithmische Lösung. Des weiteren ist die Bildqualität der generierten Ansicht an einigen Stellen noch unzureichend. So erkennt man fehlende Pixel an Übergängen im Objekt. Zum Speicheraufwand des Delta Trees findet man nur Angaben über die benötigten Patches. Der Delta Tree der vorgestellten Grundansicht der Teekanne besteht zum Beispiel aus knapp 7000 Patches. Den daraus resultierenden Speicherverbrauch in Bytes bleiben Dally, McMillan, Bishop und Fuchs dem Leser schuldig.

Literaturverzeichnis

[Dally et al. '96] William J. Dally, Leonard McMillan, Gary Bishop und Henry Fuchs. The Delta Tree: An Object-Centered Approach to Image-Based Rendering. MIT AI Lab Technical Memo 1604, May 1996.

[McMillan & Bishop '95] L. McMillan und G. Bishop. Plenoptic Modeling: An Image-Based Rendering System. In *Computer Graphics Proceedings, Annual Conference Series (Proc. SIGGRAPH '95)*, Seiten 39–46, 1995.