Kompression von bildbasierten Szenenrepräsentationen für interaktive Umgebungen

Dissertation
der Fakultät für Informatik
der Eberhard-Karls-Universität Tübingen
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
(Dr. rer. nat.)

vorgelegt von
Dipl.-Inform. Ingmar Peter
aus Tübingen

Tübingen
2002

Tag der mündlichen Qualifikation:	6. Februar 2002
Dekan:	Prof. Dr. A. Zell
1. Berichterstatter:	Prof. Dr.-Ing. W. Straßer
2. Berichterstatter:	Prof. Dr. W. Rosenstiel

Danksagung

Viele Personen haben auf unterschiedliche Weise zum Gelingen dieser Arbeit beigetragen. Ohne alle Beteiligten namentlich nennen zu können, möchte ich meinen Eltern und meinen Geschwistern, Georg Pietrek, Stefan Gumhold, Stanislav Stoev, Michael Wand, Kirsten Terfloth, Thomas Hornung und allen Mitarbeitern am WSI/GRIS meinen besonderen Dank aussprechen. Dank auch an Prof. Dr. W. Rosenstiel für die Erstellung des Zweitgutachtens. Nicht zuletzt möchte ich mich bei Prof. Dr.-Ing. W. Straßer für die vertrauensvolle Zusammenarbeit und die Betreuung dieser Arbeit bedanken.

Zusammenfassung

Traditionelle Computergrafik beschreibt eine am Computer darzustellende Szene durch eine Menge geometrischer Primitive. Die Konstruktion einer geometrischen Szenenbeschreibung kann allerdings sehr arbeitsaufwendig und teuer sein. Die bildbasierte Modellierung und Darstellung (IBR) benutzt stattdessen Bildinformation zur Beschreibung und Darstellung einer Szene. Bildinformation ist leicht zu akquirieren und erfasst auf einfache Weise alle Eigenschaften eines beliebigen Objekts.

Theoretische Grundlage für IBR ist die plenoptische Funktion, welche die Beleuchtung einer Szene vollständig beschreibt. Ein Lichtfeld speichert eine vierdimensionale Teilmenge der plenoptischen Funktion und kann auf diese Weise Objekte mit beliebigen Materialeigenschaften und hoch komplexer Geometrie erfassen. Hierzu wird lediglich eine Menge von Bildern des Objekts benötigt. Lichtfelder können sehr effizient dargestellt werden, da der Aufwand bei der Darstellung nur linear mit der Auflösung der berechneten Ansicht wächst. Allerdings muss eine große Datenmenge in einem Lichtfeld gespeichert werden, um die plenoptische Funktion hinreichend genau abzutasten.

Die in dieser Arbeit vorgestellte Waveletstream-Datenstruktur erlaubt die progressive Speicherung, Übertragung und Darstellung von komprimierten Lichtfeld-Datensätzen. Zur Kompression der Daten wird eine Wavelet-Dekomposition durchgeführt. Die berechneten Koeffizienten werden in einem Datenstrom progressiv übertragen und ermöglichen die einfache Darstellung des Lichtfeldes in unterschiedlichen Auflösungsstufen. Zur Unterscheidung von Vorder- und Hintergrund wird die Objektsilhouette im Waveletstream gespeichert. Hierdurch wird nicht nur die Integration von bildbasierten Objekten in herkömmlich modellierte Szenen unterstützt, sondern auch die Kompression erhöht. Mit dem Waveletstream ist es möglich, Lichtfelder in einem Verhältnis von $1:100$ zu komprimieren. Nur $0,5$ Prozent der Koeffizienten sind ausreichend, um eine beliebige Lichtfeld-Ansicht mit guter Qualität aus dem Waveletstream zu rekonstruieren. Dabei wird die interaktive Dekompression und Darstellung der Lichtfelder durch den Einsatz spezieller Caching-Verfahren gewährleistet.

Außerdem werden in dieser Arbeit verschiedene Beispiele für die Integration von bildbasierten Datenstrukturen in interaktive Anwendungen vorgestellt. Die Übertragung und Darstellung von Lichtfeldern im Internet, die Integration von Lichtfeldern in eine Programmierumgebung für klassische Computergrafik und die Nutzung einer bildbasierten Datenstruktur in virtueller Realität zeigen, dass bildbasierte Objekte sinnvoll mit klassischer Computergrafik kombiniert werden können und virtuelle Welten bereichern können.

Abstract

In traditional computer graphics a scene to be rendered on a computer screen is described by a number of geometric primitives. However, construction of a geometric scene model can be a great deal of work and can be very expensive. Instead, image based modelling and rendering (IBR) employs pictorial information to describe and render a scene. Acquisition of images is straightforward and all properties of arbitrary objects are captured.

Theoretical foundation for IBR is the plenoptic function, which stores the entire illumination in a scene. A Light Field consists of a four-dimensional subset of the plenoptic function. In this way objects with arbitrary material properties and complex geometry are described. To obtain a Light Field, only a number of pictures of an object are sufficient. Light Fields can be rendered efficiently, since the cost for rendering increases only linearly with the resolution of the displayed view. Unfortunately, a large amount of data has to be stored in a Light Field to sample the plenoptic function with sufficient density.

The Waveletstream data structure presented in this work allows for progressive transmission, storage, and rendering of compressed Light Field data. For compression, the non-standard four-dimensional wavelet decomposition of the Light Field is calculated. The coefficients obtained are transmitted progressively in a data stream and allow for display of the Light Field in different resolutions. To distinguish between fore- and background, the silhouette of the object is stored in the Waveletstream. In this way, not only integration of image based objects into conventional modelled scenes is facilitated, but also the compression ratio is increased. The Waveletstream allows for compression of Light Fields at a ratio of $1:100$. Only 0.5 percent of the coefficients are sufficient to reconstruct an arbitrary view from the Waveletstream with good quality. Supplementary caching schemes allow for interactive decompression and display of Light Fields.

Furthermore, this thesis presents various examples for integration of image based data structures into interactive applications. Transmission and display of Light Fields over the internet, integration of Light Fields into a application programming interface for traditional computer graphics, and the use of image based objects in virtual reality demonstrate how image based objects can be combined with traditional computer graphics and can be used to enhance virtual worlds.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Ziel fast aller Arbeiten auf dem Gebiet der Computergrafik ist es, die Welt so, wie sie ist, oder auch so, wie wir sie uns vorstellen, abzubilden. Auf diese Weise hat sich Computergrafik in den letzten Jahrzehnten eine Vielzahl von Anwendungsfeldern erschlossen und ermöglicht in unterschiedlichsten Bereichen von Entwurf und Konstruktion, Ausbildung und Wissenschaft bis hin zur Unterhaltungsindustrie die Berechnung faszinierender, erläuternder und häufig sehr unterhaltsamer (bewegter) Bilder. Computergrafik ist das geeignete Mittel, um komplexe Zusammenhänge sowohl in der Welt der Atome als auch beim Kollaps von Sonnen anschaulich und damit verständlich darzustellen.

Bevor allerdings Bilder im Computer berechnet werden können, sind vorbereitende Schritte notwendig. Traditionell wird eine Szene, die interaktiv dargestellt werden soll, im Computer durch eine Menge von geometrischen Primitiven, meist Dreiecken, beschrieben. Zur Verbesserung des Ergebnisses können den Dreiecken verschiedene Attribute zugeordnet werden, welche die Darstellung beeinflussen. Bei der Darstellung am Bildschirm werden diese Attribute ausgewertet und wird die gegenseitige Verdeckung der Dreiecke rekonstruiert. Da für den größten Teil der hierbei benutzten Verfahren spezialisierte Hardware verfügbar ist, kann eine große Zahl von schattierten Dreiecken in kurzer Zeit am Bildschirm ausgegeben werden. Dennoch hat diese traditionelle Form der Computergrafik zwei entscheidende Nachteile.

Geometrische Szenenmodelle von beliebigen Objekten müssen durch manuelle Modellierung am Computer oder durch den Einsatz technischer Hilfsmittel, wie Entfernungsmesser oder ähnliche Technik, gewonnen werden. Doch gerade bei großen, sehr komplexen Objekten, die sich nicht oder nur schwer erfassen lassen, ist die Erstellung einer geometrischen Szenenrepräsentation mit wirtschaftlich vertretbarem Aufwand nicht möglich. Außerdem ist die Größe einer Szene, welche sich interaktiv mit einem Computer darstellen läßt, durch die Anzahl der Dreiecke, welche die Grafikhardware pro Sekunde verarbeiten kann, begrenzt. Diese Grenze kann zwar durch Einsatz spezieller Verfahren hinausgeschoben werden, doch dadurch wird die lineare Abhängigkeit zwischen Darstellungsaufwand und Anzahl der dargestellten Dreiecke nicht aufgehoben. Aus diesem Grund ist, trotz bedeutender Fortschritte auf dem Gebiet der Grafikhardware, die Behandlung und Beherrschung von Szenenkomplexität nach wie vor eines der wichtigsten Themen in der Computergrafik.

Die *bildbasierte Modellierung und Darstellung* nutzt einen gänzlich anderen Ansatz zur Erzeugung realistischer Computergrafik. Da eine Fotografie per Definition fotorealistisch ist, wird bei bildbasierten Darstellungstechniken versucht, geometrische durch bildliche Szeneninformation zu ersetzen. Die bildbasierte Szenenrepräsentation besitzt zwei entscheidende Vorteile:

1. Zur Erstellung einer bildbasierten Szenenbeschreibung eines beliebigen Objekts ist eine Menge von Fotos oder eine Videosequenz ausreichend. Damit entfällt die aufwendige und teure Erzeugung eines geometrischen Szenenmodells, wie bei herkömmlicher Computergrafik.

2. Eine bildbasierte Szenenbeschreibung ist von den Eigenschaften des abgebildeten Objekts unabhängig. Die Darstellungszeit einer bildbasiert modellierten Szene hängt nur von der Auflösung der verwendeten Quellbilder ab.

Auf Grund dieser Eigenschaften können mit bildbasierter Computergrafik leicht Ergebnisse erzielt werden, die mit klassischen Methoden gar nicht oder nur mit sehr großem Aufwand zu verwirklichen wären.

Allgemeiner betrachtet wird bei der bildbasierten Szenendarstellung versucht, beliebige Ansichten einer Szene aus einer begrenzten Zahl von Abtastwerten der plenoptischen Funktion zu rekonstruieren. Die plenoptische Funktion $P(\theta, \phi, \lambda, t, x, y, z)$ gibt die Leuchtdichte für jeden Punkt (x, y, z) im Raum, für jede Richtung (θ, ϕ), zu jedem Zeitpunkt t und auf jeder Wellenlänge λ an. Bildsynthese kann damit abstrakt als die Projektion der plenoptischen Funktion in einen zweidimensionalen Bildraum beschrieben werden. Die Organisation und der Umfang der hierzu benutzten Stichprobe der plenoptischen Funktion hängt dabei im wesentlichen von den zur Verfügung stehenden Ausgangsdaten und Ressourcen wie Rechenleistung und Speicherplatz, Annahmen über Eigenschaften der gespeicherten Szene und dem benutzten Darstellungsverfahren ab.

Eine verallgemeinerte Szenenbeschreibung auf Basis der plenoptischen Funktion kann zur Speicherung, Veränderung und Darstellung von Szenen verwendet werden. In dieser Arbeit werden Verfahren und Techniken vorgestellt, mit denen bildbasierte Szenenprimitive für die interaktive Visualisierung virtueller Welten nutzbar gemacht werden können. Hierzu wird unter anderem ein Kompressionsverfahren für Lichtfelder vorgestellt. Lichtfelder speichern eine vierdimensionale Teilmenge der plenoptischen Funktion und realisieren auf diese Weise eine sehr allgemeine Form der Szenenbeschreibung. Der größte Vorteil von Lichtfeldern ist, dass Objekte mit beliebigen Materialeigenschaften und hoch komplexer Geometrie erfasst werden können. Hierzu wird lediglich eine Menge von Bildern des Objekts benötigt. Ein anderer Vorteil von Lichtfeldern ist, dass diese sehr effizient dargestellt werden können. Da der Aufwand bei der Darstellung nur linear mit der Auflösung der berechneten Ansicht wächst, sind Lichtfelder auch auf leistungsschwachen Computern einsetzbar.

Allerdings müssen in einem Lichtfeld sehr große Datenmengen gespeichert werden. Schon bei einer Auflösung von $256 \times 256 \times 32 \times 32$ Werten und einer 24 Bit genauen Farbdarstellung hat ein Lichtfeld eine Größe von 192 MByte. Aus diesem Grund wird in dieser Arbeit mit dem *Waveletstream* ein leistungsfähiges Kompressionsverfahren speziell für Lichtfelder vorgestellt.

Die Waveletstream-Datenstruktur erlaubt die progressive Speicherung, Übertragung und Darstellung von komprimierten Lichtfeld-Datensätzen. Zur Kompression der Daten wird eine Wavelet-Dekomposition durchgeführt. Die berechneten Koeffizienten können nicht nur in einem Datenstrom progressiv übertragen werden, die Wavelet-Dekomposition ermöglicht auch die einfache Darstellung des Lichtfeldes in unterschiedlichen Auflösungsstufen. Mit dieser Eigenschaft können Darstellungsfehler bei Betrachtung des Lichtfeldes aus unterschiedlichen Abständen vermieden werden. Zur Unterscheidung von Vorder- und Hintergrund wird die Objektsilhouette im Waveletstream gespeichert. Hierdurch wird nicht nur die Integration von bildbasierten Objekten in herkömmlich modellierte Szenen unterstützt, sondern auch die Kompression erhöht. Durch weitere Verbesserungen des Waveletstreams ist es möglich, Lichtfelder in einem Verhältnis von $1 : 100$ zu komprimieren. Nur $0,5$ Prozent der Koeffizienten sind ausreichend, um eine beliebige Lichtfeld-Ansicht mit guter Qualität aus dem Waveletstream zu rekonstruieren. Dabei wird die interaktive Dekompression und Darstellung der Lichtfelder durch den Einsatz spezieller Beschleunigungsdatenstrukturen gewährleistet. Die hier vorgestellte Waveletstream-Datenstruktur ermöglicht neben hohen Kompressionsraten als einzige der bislang veröffentlichten Ansätze zur Speicherung und Darstellung von Lichtfeldern deren interaktive Dekompression und Darstellung auf unterschiedlichen Skalen. Weiterhin zeichnet den Waveletstream die Möglichkeit aus, beliebige zusätzliche Informationen und die Objektsilhouette zu speichern.

Bildbasierte Datenstrukturen können in zahlreichen Anwendungen sinnvoll eingesetzt werden. In dieser Arbeit werden als Beispiele die Übertragung und Darstellung von Lichtfeldern im Internet, die Integration von Lichtfeldern in eine Programmierumgebung für klassische Computergrafik und die Nutzung einer bildbasierten Datenstruktur in virtueller Realität vorgestellt. An letzterer Anwendung wird demonstriert, wie erst bildbasierte Techniken die Realisation eines leistungsfähigen Werkzeugs zur Navigation in künstlichen Welten ermöglichen.

1.1 Aufbau der Arbeit

Im folgenden Kapitel werden die wesentlichen Konzepte klassischer und bildbasierter Computergrafik vorgestellt und analysiert. Außerdem werden zur Einordnung der vorliegenden Arbeit die wichtigsten Veröffentlichungen auf dem Gebiet der bildbasierten Szenenmodellierung und -darstellung kurz beschrieben. Kapitel 3 enthält eine Übersicht über die grundlegenden Begriffe und Verfahren der Multi-Skalen-Analyse, so wie sie in dieser Arbeit benutzt werden. Darauf aufbauend wird die praktische Anwendung von Wavelet-Kompression bei Bildern, Volumendaten und Lichtfeldern beschrieben.

In Kapitel 4 wird mit dem Waveletstream ein Verfahren zur Wavelet-Kompression von Lichtfeldern vorgestellt. Nach einer kurzen Betrachtung bisheriger Arbeiten werden Anforderungen an ein Kompressionsverfahren für Lichtfelder formuliert und die Benutzung existierender Verfahren diskutiert. Daran schließt sich eine detaillierte Beschreibung des Waveletstreams an. Erzeugung, Struktur und interaktive Darstellung komprimierter Lichtfelder mit dem Waveletstream werden ausführlich beschrieben. Das zur Implementierung entwickelte objektorientierte Programmdesign und eine Darstellung der erzielten Resultate beenden das Kapitel.

Die Integration von bildbasierten Szenenbeschreibungen in interaktive Umgebungen sind Thema von Kapitel 5. Zunächst wird die Integration von Lichtfeldern in eine Klassenbibliothek zur Darstellung klassischer Computergrafik demonstriert. Danach wird mit dem *Multi-LDI* eine bildbasierte Datenstruktur für die Realisierung von Navigationshilfen in komplexen VR-Umgebungen vorgestellt. Die Beschreibung einer spezialisierten Implementierung für die Darstellung und Übertragung von Lichtfeldern im Internet schließt das Kapitel ab. Zuletzt werden in Kapitel 6 die erzielten Resultate diskutiert, und es wird ein Ausblick auf zukünftige Arbeiten gegeben.

Kapitel 2

Image Based Rendering

Ein Ziel im Forschungsgebiet der Computergrafik ist die interaktive Erzeugung von naturgetreuen Ansichten dreidimensionaler virtueller Szenen. Der Inhalt der dargestellten Szenen ist dabei so beliebig wie unterschiedlich: Von der Rekonstruktion wirklich existierender Orte im Computer über die Erschaffung von Fantasiewelten zur Unterhaltung, bis hin zur wissenschaftlichen Visualisierung in der Physik oder Medizin ermöglicht die Computergrafik die Erzeugung von (bewegten) Bildern.

Traditionell wird in der Computergrafik eine Szene durch eine Menge von geometrischen Primitiven modelliert und dargestellt. Dieser Ansatz hat allerdings einige Schwächen, welche im nächsten Abschnitt analysiert werden. Dagegen baut die Methode der *bildbasierten Szenenmodellierung und -darstellung (IBR)* (Abschnitt 2.3) auf einer verallgemeinerten Szenenbeschreibung durch die *plenoptische Funktion* (Abschnitt 2.2) auf. Abschließend werden in diesem Kapitel zur Einordnung der vorliegenden Arbeit die wesentlichen grundlegenden und aktuellen Arbeiten auf dem Gebiet des IBR vorgestellt.

2.1 Traditionelle Computergrafik

In der traditionellen Computergrafik wird zur Repräsentation einer Szene $\mathcal{S}$ eine Menge von geometrischen Primitiven benutzt, aus welcher die einzelnen Szenenobjekte zusammengesetzt werden. Da sich die Algorithmen zur Berechnung der notwendigen Transformationen sowie der Sichtbarkeit leicht in spezialisierte und deswegen sehr effiziente Grafikhardware gießen lassen, hat sich die Benutzung von Dreiecken als geometrisches Primitiv durchgesetzt. Zur Bestimmung der Farbe jedes Punktes $\mathbf{x} \in \mathcal{S}$ werden jedem Primitiv zusätzliche Attribute zugeordnet, die zur Auswertung eines einfachen *lokalen Beleuchtungsmodells* (z. B. [PHONG 1975, BLINN 1977]) benutzt werden. Die Parameter eines lokalen Beleuchtungsmodells bestehen zumeist aus den Flächennormalen und einer Anzahl Koeffizienten, welche die Materialeigenschaften beschreiben. Die Angabe von Flächennormalen ist sinnvoll, da gekrümmte Flächen durch Dreiecksnetze nur angenähert werden können. Durch die Angabe von Normalenrichtungen und die Verwendung eines geeigneten Interpolationsverfahrens (z. B.

Phong-Shading [PHONG 1975]) kann die Darstellung einer Oberfläche deutlich verbessert werden.

Bei der interaktiven Bilderzeugung mittels geometrischer Primitive wird vereinfachend angenommen, dass das Medium, in dem sich die Szenenobjekte befinden, keinen Einfluß auf die Bilderzeugung hat. Wenn die Leuchtdichte entlang einer Geraden im Raum somit konstant bleibt, ist es zur Berechnung eines Bildes ausreichend, für jeden Bildpixel den Teil der Geometrie zu bestimmen, welcher am nächsten zum Betrachter $\mathbf{e}$ liegt und die durch den jeweiligen Pixel definierte Sichtpyramide schneidet. Die Farbe der sichtbaren Geometrie kann mit Hilfe des lokalen Beleuchtungsmodells berechnet werden. Zur Sichtbarkeitsberechnung wird gewöhnlich der *z-Buffer* (*Tiefenwertspeicher*) benutzt. Auf Grund seiner Einfachheit läßt sich dieser Ansatz leicht in Hardware realisieren und ist ein selbstverständlicher Bestandteil moderner Grafikhardware. Aktuelle Grafikkarten ermöglichen mit diesem Verfahren die Darstellung von bis zu 60 Mio. Dreiecken/Sek.

Unglücklicherweise steigt der Aufwand für die Bilderzeugung mit dem z-Buffer linear mit der Anzahl der verarbeiteten Dreiecke. Damit ist die Größe von Szenen, die interaktiv an einem Computer dargestellt werden können, durch die Leistungsfähigkeit der vorhandenen Grafikhardware begrenzt. Es wurden zahlreiche Verfahren veröffentlicht, um diese Grenze hinauszuschieben. Diese lassen sich grob in zwei Gruppen unterteilen:

Durch **Verdeckungsrechnung (Occlusion Culling)** wird versucht, möglichst große Szenenteile zu bestimmen, die für den Betrachter nicht sichtbar sind (z. B. [GREENE und KASS 1993, ZHANG et al. 1997, BARTZ et al. 1999]). Die nicht sichtbaren Szenenanteile müssen auf diese Weise erst gar nicht mit dem z-Buffer verarbeitet werden. Erfolgreiche Anwendung von Verdeckungsrechnung setzt allerdings ein gewisses Maß an tatsächlicher Verdeckung in der Szene voraus. In speziellen Szenen mit hoher Verdeckung, wie z. B. bei Architektur- [TELLER und HANRAHAN 1993] oder Stadtszenen [WONKA und SCHMALSTEIG 1999], zeigt Occlusion Culling deshalb große Wirkung.

Ein weiteres Mittel zur Verringerung der Anzahl geometrischer Primitive, die im z-Buffer verarbeitet werden müssen, ist die **Vereinfachung der Szenengeometrie**. Bei dieser Klasse von Verfahren liegt die Szene in unterschiedlichen Auflösungsstufen (*Level of Detail (LOD)*) vor (z. B. [KLEIN 1998, KLEIN et al. 1996, HOPPE 1996]). Je nach Abstand des Betrachters zum jeweilen Szenenobjekt wird eine geeignete Auflösungsstufe gewählt und dargestellt. Auf diese Weise kann der Darstellungsaufwand an die wirklich benötigte Genauigkeit der Darstellung angepasst werden. Weiterhin werden Aliasingprobleme durch eine Unterabtastung der Geometrie vermieden. Leider ist das Maß, in dem die Szene sinnvoll vereinfacht und damit die Zahl der geometrischen Primitive verringert werden kann, stark von der Geometrie der Szenenobjekte abhängig. Auf die Geometrie eines Waldes mit zahlreichen Bäumen und einzeln modellierten Blättern ist diese Klasse von Verfahren nur schwer anwendbar.

Probleme klassischer Computergrafik

Obwohl es, wie oben beschrieben, verschiedene Ansätze zur Verbesserung des klassischen z-Buffer-Verfahrens gibt, bleiben die beiden Hauptnachteile herkömmlicher Computergrafik bis heute bestehen:

1. **Die Laufzeit des klassischen z-Buffer-Verfahrens ist linear abhängig von der Anzahl der verarbeiteten geometrischen Primitive.** Somit ist die maximale Größe einer interaktiv darstellbaren Szene immer stark von der Leistungsfähigkeit der verwendeten Grafikhardware abhängig. Erst kürzlich wurden Verfahren veröffentlicht, welche diese Abhängigkeit schwächen und die interaktive Darstellung hoch komplexer Szenen bei schwacher Abhängigkeit der Laufzeit von der Anzahl der Szenenprimitive erlauben [WAND et al. 2001, WALD et al. 2001].

2. **Jede Szene muss als geometrisches Modell vorliegen.** Die Erzeugung von großen, wirklichkeitsnahen Szenenmodellen mit verschiedenartigen Materialien und komplexer Geometrie ist trotz neuer technischer Hilfsmittel (z. B. [LEVOY et al. 2000, DEBEVEC et al. 2000]) mit wirtschaftlich vertretbarem Aufwand nicht möglich. Selbst bei beherrschbarer Szenengröße können viele in der Realität vorkommende Materialien mit den einfachen lokalen Beleuchtungsmodellen der klassischen Computergrafik nur angenähert werden.

Eine verallgemeinerte, von der Geometrie unabhängige Szenenbeschreibung ist eine mögliche Lösung für die Probleme der klassischen Computergrafik.

2.2 Die plenoptische Funktion

Obwohl die Zusammenhänge, in denen interaktive Computergrafik eingesetzt wird, sehr unterschiedlich sind, kann das visuelle Erscheinungsbild jeder Szene vollständig durch die *plenoptische Funktion* P [ADELSON und BERGEN 1991] beschrieben werden. Die plenoptische Funktion

$$P(\theta, \phi, \lambda, t, \mathbf{x}) \qquad \left[\frac{W}{m^2 sr}\right]$$

gibt für jeden dreidimensionalen Punkt $\mathbf{x}$ im Raum, jede Richtung (θ, ϕ), jeden Zeitpunkt t und jede Wellenlänge λ den Wert der Leuchtdichte (*radiance*) an. Sind eine Betrachterposition $\mathbf{e}$ und eine Kameraperspektive gegeben, kann Bildsynthese abstrakt als die Bestimmung einer dreidimensionalen Teilmenge der plenoptischen Funktion beschrieben werden[1]. Das Bild einer Szene $\mathcal{S}$ besteht somit aus Werten der plenoptischen Funktion für $\mathbf{x} = \mathbf{e}$. Zur Gewinnung des Bildes wird für jedes Bildpixel die plenoptische Funktion für die entsprechenden Richtungen (θ, ϕ), welche sich aus der Lage der Bildebene relativ zur Betrachterposition $\mathbf{e}$ ergeben, ausgewertet.

[1] Dreidimensional, weil auch die Wellenlänge λ des Lichts abgetastet werden muss.

2.3 Bildbasierte Szenenmodellierung und -darstellung (IBR)

Die *bildbasierte Szenenmodellierung und -darstellung (Image Based Modelling and Rendering (IBR))* versucht die beschriebenen Probleme der klassischen geometriebasierten Szenenmodellierung und -darstellung mit folgendem Ansatz zu lösen:

Bildbasierte Darstellungsverfahren ersetzen die herkömmliche geometrische Szenenrepräsentation durch eine Abtastung der plenoptischen Funktion und berechnen direkt aus dieser neue Ansichten der Szene.

Zur Berechnung einer neuen Szenenansicht müssen somit die benötigten Werte der plenoptischen Funktion aus einer (vollständigen oder unvollständigen) Stichprobe rekonstruiert und dargestellt werden. Bildbasierte Verfahren überwinden (oder verringern zumindest) die beiden Hauptnachteile klassischer Computergrafik mit folgender Strategie:

- **Verringerung der Anzahl der geometrischen Primitive.** Idealerweise besteht die gesamte Szeneninformation aus Bilddaten. Hierdurch kann die Zeit, die zur Darstellung einer Szene benötigt wird, höchstens mit der Größe der verwendeten Stichprobe der plenoptischen Funktion wachsen. Die Komplexität oder Größe der Szene hat keinen Einfluss auf die Zeit, die zu deren Darstellung benötigt wird.
- **Vereinfachung der Szenenmodellierung.** Da die Fotografie einer Szenerie per Definition „fotorealistisch“ ist, umgeht die direkte Verwendung von Bildinformation als Modellierungsprimitiv den zeitaufwendigen und fehlerträchtigen Modellierungsprozess, welcher zudem nur eine Näherung der realen Szene hervorbringt.

2.4 Bisherige Arbeiten

IBR ist ein sehr aktives Forschungsgebiet, auf dem eine Vielzahl von Publikationen erschienen sind, die für diese Arbeit relevant sind. Die in den letzten Jahren veröffentlichten Ansätze unterscheiden sich zum Teil sehr in

- den benutzten Ausgangsdaten,
- den Anforderungen an Rechenleistung und Speicherplatz,
- der Allgemeinheit bzw. dem Anwendungsfeld eines Verfahrens.

Dies macht eine systematische Klassifizierung der Arbeiten durch wenige Merkmale schwierig. Die nun folgende Übersicht über den aktuellen Stand der Forschung ordnet die bisherigen Arbeiten in erster Linie entsprechend deren Nutzung von Geometrie- und Bildinformation ein. Einen guten Überblick über das Gebiet der bildbasierten Modellierung und Darstellung geben außerdem [MÜLLER 1999, MCMILLAN und GORTLER 1999, GUSTAFSSON und TURBELL 1997].

2.4.1 Texturen

Eines der ältesten und einfachsten Verfahren zur Verringerung der geometrischen Szenenkomplexität ist die Benutzung von *Texturen* [BLINN und NEWELL 1976, HECKBERT 1986]. Um von den geometrischen Details eines Objekts zu abstrahieren, werden auf die Seitenflächen eines einfachen Grundkörpers Bilder des modellierten Objekts aufgebracht. Auf diese Weise kann beispielsweise die detaillierte Geometrie einer Hauswand (Abb. 2.1 (b)) oder das Vorhandensein einer feinen Oberflächenstruktur vorgetäuscht werden. Jedem Parameter eines lokalen Beleuchtungsmodells kann eine Textur zugeordnet werden, die den jeweiligen Parameter verändert. Texturen, welche die Oberflächennormalen beeinflussen, werden als *Bump Maps* [BLINN 1978] bezeichnet und täuschen geringe Höhenveränderungen auf einer Oberfläche vor.

Allerdings wird die Verwendung von Texturen offensichtlich, wenn sich der Betrachter zu weit von der Position entfernt, von welcher das für die Textur benutzte Bild erzeugt wurde: Die gegenseitige Verdeckung von auf der Textur abgebildeten geometrischen Details wird nicht dargestellt (Abb. 2.1 (c)). Die Verwendung von Bump Maps kann an der Silhouette eines Objekts erkannt werden, da diese die Form der Grundgeometrie beibehält. Mit beiden Verfahren kann außerdem die gegenseitige Verdeckung der dargestellten Detailgeometrie nicht korrekt modelliert werden.

(a) (b) (c)

Abbildung 2.1: Texturen können den Realismus eines Modells deutlich erhöhen. (a) Geometriemodell. (b) Texturiertes Geometriemodell. (c) Detailansicht einer einzelnen Hauswand aus einem flachen Winkel: Die gegenseitige Verdeckung geometrischer Details, wie Fenstererker, wird falsch dargestellt, weil lediglich eine Textur auf eine ebene Grundgeometrie aufgebracht wurde. (Modelle von [VirTü])

Trotz der beschriebenen Nachteile ist die Verwendung von Texturen in vielen Situationen eine sehr wirkungsvolle Möglichkeit, die Modellierung und Darstellung komplexer Szenen zu vereinfachen. Insbesondere aus Computerspielen sind texturierte Modelle heute nicht mehr wegzudenken. Wohl auch deshalb hat sich in den letzten Jahren die Größe des *Texturspeichers* vervielfacht. Dieser spezielle, direkt auf der Grafikkarte untergebrachte Speicher, ermöglicht eine stark beschleunigte Darstellung texturierter Szenenobjekte, da die Zugriffszeit auf die Texturdaten extrem gering ist.

Die Erzeugung texturierter Szenenbeschreibungen für reale Szenen kann durch Methoden der Bildverarbeitung stark vereinfacht werden. In [DEBEVEC et al. 1996, TARINI et al. 2000] werden Systeme zur Erstellung von Architekturmodellen beschrieben. Hierzu werden in [DEBEVEC et al. 1996] vom Benutzer in Bildern die Eckpunkte einfacher geometrischer Grundkörper wie Quader oder Pyramiden markiert. Jeder Fläche des konstruierten geometrischen Grundkörpers kann auf diese Weise leicht aus dem jeweiligen Quellbild eine Textur zugewiesen und die ungefähre Aufnahmeposition der einzelnen Quellbilder bestimmt werden.

Ein ähnlicher Ansatz wird in [POULIN et al. 1998] verfolgt, wobei an Stelle einer expliziten Modellierung der Szenengeometrie durch den Benutzer nur geometrische Bedingungen definiert werden. In einem iterativen Prozess werden Szenengeometrie und Kamerapositionen rekonstruiert, Texturen aus den Eingabebildern extrahiert und zu einem dreidimensionalen Modell der Szene zusammengefügt. Durch die Benutzung kalibrierter Kameras ermöglicht das Verfahren von Matusik et al. [MATUSIK et al. 2000] sogar die vollkommen automatische Rekonstruktion texturierter Modelle in Echtzeit [MATUSIK et al. 2001]. Aus den Silhouetten (bewegter) Objekte werden hierbei einfache polygonale Modelle rekonstruiert und texturiert.

Um die Darstellungsqualität texturierter Modelle zu verbessern, wird in [DEBEVEC et al. 1998] immer diejenige Textur bestimmt und zur Darstellung verwendet, die aus einer Richtung aufgenommen wurde, die am ehesten mit der Blickrichtung des Betrachters übereinstimmt. Im Gegensatz hierzu speichert Pulli et al. [PULLI et al. 1997] zu jedem aufgenommenen Bild ein vereinfachtes Geometriemodell und vereinigt Geometrie und Textur erst während der Darstellung zu einem einheitlichen Modell.

Verwendung von Tiefeninformation

Mit Hilfe von *Tiefeninformation* kann die perspektivische Verzerrung und gegenseitige Verdeckung von Szenenteilen, die auf einem Bild zu sehen sind, bei Bewegung des Betrachters korrekt berechnet werden. Ein Bild, welches für jedes Pixel zusätzlich zu seiner Farbe auch Tiefen- oder äquivalent *Disparitätsinformation* speichert, wird als *Tiefenbild (Depth Picture)* bezeichnet. Durch die Verwendung von Tiefenbildern kann die Genauigkeit texturierter Szenendarstellungen erhöht werden.

Die Darstellung von Texturen mit zusätzlicher Tiefeninformation wird als *Displacement Mapping* [COOK 1984] oder *Relief Texturing* [OLIVEIRA et al. 2000] bezeichnet. Diese Verfahren erlauben im Gegensatz zu herkömmlichen Texturen die richtige Darstellung von Objektsilhouetten und der gegenseitigen Verdeckung. Geometrische Details, wie z. B. Fenstervorsprünge oder Markisen an Hauswänden, können perspektivisch korrekt abgebildet werden. Weiterhin erlauben Displacement Maps und Relief Textures nach einer einfachen Rekonstruktion der Oberflächengeometrie die schattierte Darstellung der Geometrie mit Verfahren herkömmlicher Computergrafik [OLIVEIRA und BISHOP 1998]. Schaufler et al. beschreibt in [SCHAUFLER und PRIGLINGER 1999] die interaktive Darstellung von Displacement Maps mit bildbasierten Verfahren.

In [GUMHOLD und HÜTTNER 1999, DOGGETT et al. 2001] wird spezialisierte Hardware zur Darstellung von Displacement Maps vorgestellt. Das Darstellungsverfahren für Relief Textures ist besonders einfach, da diese auf eine ebene Fläche aufgebracht werden [OLIVEIRA und BISHOP 1999a]. Deshalb bietet sich eine Implementation in Hardware an [POPESCU et al. 2000].

2.4.2 Imposters

In der bisherigen Diskussion war davon ausgegangen worden, dass Texturen Bilder sind, welche Detailinformationen in Bezug auf eine einfache Grundgeometrie enthalten. Es ist aber ebenso sinnvoll, einfache, zweidimensionale Bildinformation **ohne** zugehörige „Trägergeometrie“ zu benutzen, um die Darstellung komplexer geometriebasierter Szenen zu verbessern.

In [SHADE et al. 1996] werden zur Darstellung der weiter entfernten Teile einer durchwanderten Szene texturierte Flächen (*Imposters*) benutzt. Hierdurch müssen lediglich kleine Teile der Szenengeometrie nach jeder Bewegung des Betrachters erneut mit Hilfe des z-Buffers dargestellt werden. Ein spezielles Fehlermaß stellt sicher, dass das Bild eines Imposters neu berechnet oder dieser durch eine Geometriedarstellung ersetzt wird, sobald die Abweichung von der korrekten Szenenansicht zu groß wird.

Eine spezialisierte Anwendung von Imposters in Stadtszenen wird in [SILLION et al. 1997, WIMMER et al. 1999] beschrieben. Aliaga et al. schlägt in [ALIAGA und LASTRA 1997] die Benutzung von *Portal Textures* in Architekturmodellen vor. Die Portal Textures werden in Öffnungen, wie Fenster oder Türen, eingesetzt, um die dahinter liegenden Räume darzustellen. Anstelle einer Unterteilung der Szene mittels eines BSP-Baums wie in [SHADE et al. 1996], beschreibt die Arbeit von Maciel und Shirley [MACIEL und SHIRLEY 1995] das „Verpacken“ von komplexen Objekten in texturierte Hüllvolumen. Diese ersetzen bei entsprechendem Abstand des Betrachters die exakte geometrische Darstellung des jeweiligen Szenenobjektes. Da die Hüllvolumen für eine Vielzahl unterschiedlicher Betrachtungsrichtungen und Auflösungen erzeugt werden müssen, benötigt dieser Ansatz allerdings eine große Menge Speicherplatz.

2.4.3 Rein bildbasierte Verfahren

Eine Verallgemeinerung der bislang geschilderten Ideen führt zu Szenenrepräsentationen, welche ausschließlich auf Bildinformationen beruhen. Die im folgenden beschriebenen Verfahren benutzen eine Stichprobe der plenoptischen Funktion P in Form von Bildern, um neue Ansichten einer Szene zu berechnen.

QuickTime VR [CHEN 1995] speichert eine Szene als Menge von zylindrischen Panoramabildern, die aus herkömmlichen Fotografien mit identischem Projektionszentrum gewonnen werden können [SZELISKI und SHUM 1997]. Der Betrachter kann die Blickrichtung innerhalb eines Panoramas frei wählen, Bildteile stufenlos vergrößern und zwischen den unterschiedlichen Panoramen wechseln. Allerdings darf jedes Panorama immer nur von einer festen Position im Zentrum des Zylinders betrachtet werden. Außerdem wird der Rand des Panoramas sichtbar, wenn die Kamera zu weit nach oben oder unten geneigt wird.

2.4.4 Kombination von Bild- und Disparitätsinformation

Um aus existierenden Bildern neue Ansichten einer Szene für beliebige Betrachterpositionen berechnen zu können, wird zusätzliche Information benötigt. Wie schon bei den Texturen, ermöglicht Tiefen- bzw. Disparitätsinformation die korrekte Berechnung der perspektivischen Verzerrung und der gegenseitigen Verdeckung von Szenenelementen.

In [CHEN und WILLIAMS 1993] werden aus existierenden Bildpaaren mit Hilfe von Tiefeninformation zusätzliche Bilder für neue Betrachterpositionen erzeugt (*Image Warping*). Hierbei werden die Pixel der Quellbilder mit Hilfe der Tiefeninformation auf die Bildebene der neuen Betrachterposition projiziert. In [MARK et al. 1997] wird Image Warping angewendet, um die interaktive Darstellung komplexer Szenen zu beschleunigen. Hierbei wird nur ein geringer Teil der benötigten Ansichten traditionell mit dem z-Buffer berechnet. Die Zwischenbilder werden mit Hilfe der Tiefeninformation interpoliert. Rafferty et al. [RAFFERTY et al. 1998a] schlägt die Verwendung von Image Warping zur beschleunigten Darstellung von Architekturmodellen vor. Räume, die durch Fenster oder Türen zu sehen sind, werden durch Tiefenbilder dargestellt. Der Einsatz von Tiefenbildern ermöglicht hierbei im Vergleich zu [ALIAGA und LASTRA 1997] eine verbesserte Darstellung.

Laveau und Faugeras [LAVEAU und FAUGERAS 1994] speichern eine Szene als eine Menge von Bildern mit zugehörigen Transformationen, die zur Berechnung neuer Ansichten benutzt werden. Auf ähnliche Weise werden in [MCMILLAN und BISHOP 1995, ALIAGA und CARLBOM 2001] neue Ansichten aus einer Menge von Panoramen berechnet. Gemeinsam repräsentieren diese eine vierdimensionale Abtastung der plenoptischen Funktion.

Die beschriebenen Verfahren benötigen korrekte Tiefenwerte oder äquivalente Angaben. Diese sind aber für reale Szenen oft schwierig oder nur mit großem technischen Aufwand zu bestimmen [MCALLISTER et al. 1999]. Um die verschiedenen Szenenansichten gegenseitig in Bezug zu setzen, werden in [LAVEAU und FAUGERAS 1994] und [MCMILLAN und BISHOP 1995] eine Anzahl von Punktkorrespondenzen für alle Eingabebilder benötigt. In [CHAN et al. 1999] wird deshalb ein Verfahren zur Segmentierung und dem automatischen Finden von Korrespondenzen zwischen Bildpaaren beschrieben. Aliaga und Carlbom [ALIAGA und CARLBOM 2001] vermeiden diesen Schritt durch die automatische Erzeugung einer ausreichend dichten Repräsentation der plenoptischen Funktion.

Eine andere Möglichkeit, Tiefenwerte zu bestimmen, sind Ansätze, bei denen ganzen Gruppen von Bildpunkten vom Benutzer eine gemeinsame Tiefe zugewiesen wird. In [HORRY et al. 1997, OH et al. 2001, KANG et al. 2001] wird das Quellbild vom Benutzer in die Abbildungen der unterschiedlichen Objekte zerlegt. Jedes Segment bekommt danach (manuell) eine Tiefe zugewiesen und wird im virtuellen Bildraum, ähnlich einer Theaterkulisse, „aufgestellt“. Der Bildraum kann nach Ergänzung von eventuell fehlender Information über den Bildhintergrund vom Benutzer frei durchwandert werden.

(a)

(b)

(c)

Abbildung 2.2: Tiefen- oder Disparitätsinformation erlaubt die Rekonstruktion neuer Ansichten einer Szene durch Projektion der Pixel der Quellbilder auf die Bildebene der neuen Ansicht. (a) Quellbild. (b) Die neue Ansicht der Szene weist „Löcher“ in Bereichen auf, die auf dem Quellbild verdeckt sind. (c) Vollständige Bildrekonstruktion mit Hilfe von Informationen aus zusätzlichen Tiefenbildern.

Bildrekonstruktion beim Image Warping

Ein wesentliches Problem beim Image Warping ist die Veränderung der Sichtbarkeitsverhältnisse in der betrachteten Szene durch die Veränderung des Standpunkts des Betrachters. Zum einen können Szenenteile, die vormals verdeckt waren und unter Umständen auf keinem der Eingabebilder enthalten sind, sichtbar werden (Abb. 2.2 (b)); zum anderen können mehrere Pixel aus einem Quellbild auf ein einziges Pixel im Zielbild abgebildet werden. Indem eine geeignete Verarbeitungsreihenfolge der Quellbilder gewählt [MCMILLAN und BISHOP 1995], ein z-Buffer benutzt wird (z. B. [MARK et al. 1997, HORRY et al. 1997]) oder spezialisierte Verfahren zur korrekten Darstellung der Verdeckung verwendet werden [FU et al. 1999] ist letzteres Problem einfach zu lösen. Ein größeres Problem stellen Bildbereiche dar, zu deren Aussehen keine Information vorliegt („Löcher"). Sind diese Bereiche nicht zu groß, können sie mit Hilfe einfacher Heuristiken aus bekannten benachbarten Bildbereichen durch Interpolation überbrückt werden [CHEN und WILLIAMS 1993, MARK et al. 1997]. Eine andere Strategie zur Vermeidung von Löchern ist eine geeignete Auswahl der Aufnahmepositionen der Quellbilder [FLEISHMAN et al. 1999].

Allgemein kann das geschilderte Problem aber nur durch die Verwendung einer dichten und möglichst vollständigen Stichprobe der plenoptischen Funktion P vermieden werden. Dies bedeutet bei den bislang beschriebenen Verfahren die Benutzung einer womöglich hohen Anzahl von Quellbildern. Die Laufzeit eines Image Warping-Verfahrens ist allerdings linear von den Gesamtzahl der zu transformierenden Pixel abhängig [MCMILLAN und BISHOP 1995, CHEN und WILLIAMS 1993]. Die Anzahl der Pixel, die interaktiv transformiert werden können, ist begrenzt. Daher ist es wünschenswert, dass jeder Oberflächenpunkt bzw. Abtastwert der plenoptischen Funktion in einer Szene während der Bilderzeugung höchstens einmal behandelt wird.

Die *Delta Tree* Datenstruktur [DALLY et al. 1996] enthält eine Anzahl von Tiefenbildern, welche die Szene aus unterschiedlichen Winkeln zeigen. Um jeden Oberflächenpunkt der Szene nur einmal zu speichern, werden nur Differenzbilder in Bezug auf die jeweils direkten Nachbarn gespeichert. Andere sparsame bildbasierte Szenenrepräsentationen werden in [MAX und OHSAKI 1995, RADEMACHER und BISHOP 1998, HANSON und WERNERT 1998] beschrieben. Zur Erzeugung von *Multiple-Center-of-Projection (MCOP) Images* [RADEMACHER und BISHOP 1998] wird eine Schlitzkamera entlang einer kontinuierlichen Bahn durch die Szene bewegt. An jeder Kameraposition wird dem MCOP Bild eine Spalte zusammen mit Disparitätsinformation für jedes Pixel in der jeweiligen Spalte hinzugefügt. Auf diese Weise soll jeder Oberflächenpunkt der Szenen nur einmal im MCOP Bild gespeichert werden. Die in [HANSON und WERNERT 1998] beschriebene Datenstruktur vereint im Stil kubistischer Bilder die Information unterschiedlicher Betrachterpositionen in einer einheitlichen Darstellung. Max und Ohsaki schlagen in [MAX und OHSAKI 1995] erstmals die Verwendung eines *Multi-Layered Z-Buffers* vor. In diesem können in jedem Pixel eines Bildes mehrere Oberflächenpunkte mit unterschiedlichen Tiefen abgelegt werden. Diese Idee für eine sparsame Repräsentationen der plenoptischen Funktion wird später in [SCHAUFLER 1998] und [SHADE et al. 1998] für die interaktive Darstellung komplexer Objekte aufgegriffen.

2.4.5 Layered Imposters und geschichtete Tiefenbilder

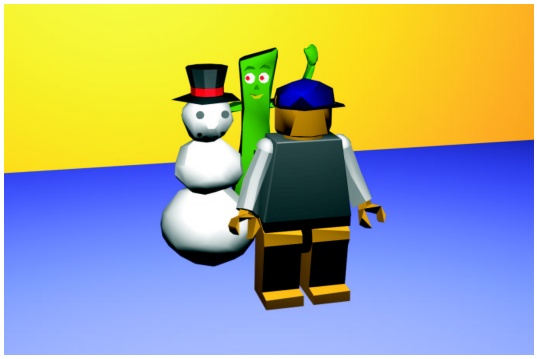


Abbildung 2.3: Einfache Beispielszene mit drei Spielzeugfiguren.

Layered Imposters [SCHAUFLER 1998] und *geschichtete Tiefenbilder (Layered Depth Images (LDI))* [SHADE et al. 1998] speichern im Gegensatz zu den oben beschriebenen Datenstrukturen alle Abtastwerte der plenoptischen Funktion in einem einheitlichen Bezugssystem ab. Alle Werte beziehen sich auf nur eine Betrachterposition. Dies ermöglicht bei den LDIs deren Darstellung mit einem besonders effizienten Algorithmus und bei den Layered Imposters eine sehr gute Ausnutzung der Funktionalität von Grafikhardware.

Layered Imposters und geschichtete Tiefenbilder unterscheiden sich hauptsächlich in der verwendeten Datenstruktur und dem darauf beruhenden Verfahren zur Darstellung. Layered Imposters benutzen eine Schar paralleler Ebenen, die mit dem darzustellenden Objekt geschnitten werden (Abb. 2.4 (a)). Punkte auf der Objektoberfläche werden jeweils senkrecht auf die am nächsten liegende „Tiefenebene“ projiziert. Auf diese Weise kann ein Objekt durch eine Menge von teiltransparent texturierten Ebenen interaktiv mittels Grafikhardware sehr einfach dargestellt werden. Eine geringe Überlappung der Bilder zwischen den verschiedenen Ebenen verhindert das Auftreten von Lücken bei der Darstellung. Nachteil dieser Vereinfachung ist, dass jedem Bildpunkt einer von endlich vielen verschiedenen Tiefenwerten zugeordnet werden muss. Wird eine zu geringe Anzahl an Tiefenebenen gewählt, können bei der Darstellung Verzerrungen auftreten. Werden zu viele Tiefenebenen angelegt, wird die Darstellung unter Umständen zu langsam. Weiterhin kann eine Selbstverdeckung von Objekten eventuell nicht korrekt dargestellt werden. In [DECORET et al. 1999] werden Layered Imposters zur Verbesserung der Visualisierung komplexer Stadtszenen [SILLION et al. 1997] eingesetzt.

Im Gegensatz zu den Layered Imposters ist ein geschichtetes Tiefenbild (LDI) ein Tiefenbild, bei dem jedes Pixel eine beliebige Anzahl von Farb- mit zugehörigen Tiefenwerten (*Tiefenpixel*) enthalten kann. Auf diese Weise werden in einem LDI auch verdeckte Szenenteile gespeichert und können bei einer Veränderung der Betrachterposition korrekt dargestellt werden (Abb. 2.4 (b)).

Ein LDI hat eine feste, zur Zeit der Erstellung gewählte Auflösung. In vielen Fällen ist es aber sinnvoll und nützlich, die Auflösung an die Bedürfnisse der jeweiligen An-

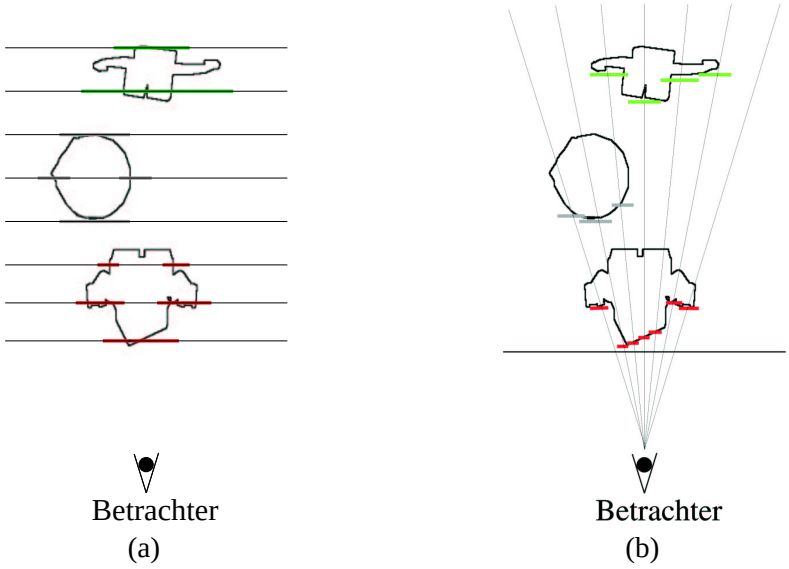


Abbildung 2.4: Schematische Darstellung der Funktionsweise von Layered Imposters und geschichteten Tiefenbildern (LDI), anhand des Modells aus Abbildung 2.3: (a) Layered Imposters bestehen aus einer Folge von teiltransparenten texturierten Flächen, welche die Bildpunkte des Objekts mit zur jeweiligen Fläche ähnlicher Tiefe speichern. (b) LDIs speichern Objekte als eine geordnete Menge von Tiefenpixeln, welche an der regulären Struktur eines Tiefenbildes ausgerichtet sind.

wendung anzupassen. In [CHANG et al. 1999] wird der *LDI Tree* beschrieben, der ein LDI in unterschiedlichen Auflösungsstufen speichert. Leider kann der LDI Tree nicht in einer Zeit dargestellt werden, die ausreichend kurz für interaktive Anwendungen ist.

Auf Anwendungen und auf die effiziente Darstellung von LDIs wird in Abschnitt 5.2 näher eingegangen.

2.4.6 Punkte als Darstellungsprimitive

In den letzten Jahren sind zahlreiche Arbeiten veröffentlicht worden, welche die Verwendung von Punkten an Stelle von Dreiecken als geometrische Primitive bei der Darstellung computergrafischer Szenen beschreiben. Diese Entwicklung wurde einerseits von den praktischen Bedürfnissen der bildbasierten Verfahren (z. B. [SHADE et al. 1998]) getrieben, andererseits führen auch theoretische Überlegungen wie in [WAND et al. 2001] zur Verwendung einer *Point Sample-Repräsentation* für die Szenendarstellung.

Auf Grund der enormen Leistungssteigerung der Grafikhardware in den letzten Jahren ist es heute möglich, Szenen mit einem sehr hohen Detailierungsgrad selbst auf Grafikkarten des unteren Preissegments interaktiv darzustellen. Wegen der hohen Zahl von Dreiecken in einer Szene ist dabei die durchschnittliche projizierte Fläche eines Dreiecks auf dem Bildschirm kontinuierlich gesunken. Deshalb werden häufig mehrere Dreiecke auf eine Fläche projiziert, die weniger als einem Bildpixel entspricht.

In diesen Fällen ist es wenig effizient, für alle drei Eckpunkte eines Dreiecks eine vollständige perspektivische Transformation durchzuführen, da im Ergebnis alle Eckpunkte auf dasselbe Pixel projiziert werden.

Die Benutzung von Punkten als Darstellungsprimitiv ist ein geeigneter Ausweg, um die Darstellung hoch detaillierter Szenen weiter zu beschleunigen. Pro Punkt muss die perspektivische Transformation nur noch einmal durchgeführt werden. Schon 1974 wurde von E. Catmull in [CATMULL 1974] erkannt, dass die sukzessive Anwendung von Unterteilungsalgorithmen schließlich zur Verwendung von Punkten führen muss. Praktisch wurden Punkte erstmals 1985 von Levoy und Whitted [LEVOY und WHITTED 1985] zur Darstellung von kontinuierlichen, differenzierbaren Flächen eingesetzt.

Aber auch Verfahren aus dem Bereich der bildbasierten Szenendarstellung, wie den oben beschriebenen geschichteten Tiefenbildern (LDI), motivieren die Nutzung von Punkten als Darstellungsprimitive. Wenn, wie bei den LDIs, die Materialeigenschaften einer Szene auf rein diffuse Reflexion eingeschränkt sind, so speichert ein Tiefenpixel im LDI den Wert der plenoptischen Funktion für alle Positionen und Richtungen, von denen der Punkt sichtbar ist. Ein weiterer Vorteil des Einsatzes von Punkten in bildbasierten Verfahren ist, dass eine punktbasierte Darstellung leicht mit dem z-Buffer einer vorhandenen Grafikhardware dargestellt werden kann.

In jüngster Zeit sind eine Reihe von Verfahren, die Punkte als Darstellungsprimitiv verwenden, veröffentlicht worden [GROSSMAN und DALLY 1998, SHADE et al. 1998, PFISTER et al. 2000, WAND et al. 2001, STAMMINGER und DRETTAKIS 2001, CHEN und NGUYEN 2001]. Einige von diesen werden im Folgenden vorgestellt.

Abbildung 2.5: Ansicht einer Landschaft, die mit dem randomisierten z-Buffer berechnet wurde. Durch die Darstellung von Dreiecken mit großer projizierter Fläche (z. B. die Blätter am Baum im Vordergrund) mit herkömmlicher Grafikhardware wird das Verfahren niemals langsamer als das herkömmliche z-Buffer Verfahren. Das dargestellte Bild wurde in nicht interaktiver Zeit durch Mittelung über mehrere Einzelbilder erzeugt.

2.4.7 Der randomisierte z-Buffer

Das *randomisierte z-Buffer Algorithmus* [WAND et al. 2001] erlaubt die interaktive Darstellung von Szenen, die aus bis zu 130 Milliarden Dreiecken bestehen (Abb. 2.5). Damit ist dieses Verfahren um vier bis fünf Größenordnungen besser als die beste verfügbare Grafikhardware heute.

Diese hohe Leistung wird durch die **dynamische** Erzeugung einer *Point Sample-Repräsentation* erreicht, welche die sichtbare Szenengeometrie vollständig abdeckt. Hierzu werden die Dreiecke der Szenengeometrie in unterschiedliche *Flächeninhaltsklassen* unterteilt: Für Gruppen von Dreiecken wird deren projizierte Fläche am Bildschirm abgeschätzt. Mit diesem Wert wird die Anzahl der Punkte berechnet, die auf den Oberflächen der Dreiecke zufällig zur Darstellung derselben ausgewählt werden. Durch eine gezielte Überabtastung [WAND et al. 2001] der Dreiecke wird verhindert, dass „Löcher“ entstehen und eine korrekte Darstellung der Szene gewährleistet ist. Weiterhin ist die Anzahl der Punkte durch die dynamische Erzeugung immer der Bildschirmauflösung angepasst. Die gewählten Punkte werden anschließend mit Hilfe der Grafikhardware dargestellt, wobei ein intelligentes Caching-Schema interaktive Darstellungszeiten erlaubt.

Der randomisierte z-Buffer Algorithmus ermöglicht die Auswahl und Darstellung der gewählten Punkte in *output-sensitiver* Zeit [SUDARSKY und GOTSMAN 1996]. Die Laufzeit zur Berechnung eines Bildes ist $O(a \cdot \log n)$, wobei n die Anzahl der Dreiecke in der Szene und a die projizierte Gesamtfläche aller Dreiecke ist. Dies bedeutet, dass die Laufzeit nur schwach von der Anzahl der geometrischen Primitive und stark von der Bildschirmauflösung abhängt. Es gibt allerdings Hinweise, dass die Laufzeit des Verfahrens nochmals deutlich auf rein lineare $O(a)$ verringert werden kann.

2.4.8 Andere Point Sample-Repräsentationen

Im Gegensatz zum oben beschriebenen randomisierten z-Buffer, dessen Entwicklung theoretischen Überlegungen folgte und so zu einem überlegenen Laufzeitverhalten führte, wurde die Entwicklung der im Folgenden beschriebenen Datenstrukturen von Problemen aus der Praxis motiviert.

Ein Verfahren für prozedurale und komplexe Geometrie

Stamminger und Drettakis stellen in [STAMMINGER und DRETTAKIS 2001] eine Methode zur Darstellung von prozeduralen und komplexen natürlichen Objekten, wie beispielsweise Landschaften mit Bäumen, vor. Hierbei wird, ähnlich wie in [WAND et al. 2001], dynamisch eine Point Sample-Repräsentation erzeugt. Im Gegensatz zum randomisierten z-Buffer arbeitet das Verfahren von Stamminger et al. jedoch nicht bei allen Szenenobjekten mit einem randomisierten Ansatz. Zur Darstellung von Geländemodellen wird eine reguläre Abtastung verwendet. Dabei auftretende Lücken werden detektiert und durch zusätzliche Punkte gefüllt. Obgleich die reguläre Abtastung das Auftreten von stochastischem Rauschen verhindert, konnte nicht gezeigt werden, wie die bei der regulären Abtastung auftretenden Alias-Effekte bei Unterschreitung des Nyquist-Limits [SHANNON 1948] verhindert werden sollen.

QSplat

Das *QSplat*-Verfahren [RUSINKIEWICZ und LEVOY 2000] wurde von S. Rusinkiewicz und M. Levoy entwickelt, um hoch detaillierte geometrische Modelle, die mittels Laser-Scannern gewonnen wurden, interaktiv darzustellen. Hierzu wird auf einer Menge von Oberflächenpunkten eine Hierarchie von Hüllkugeln aufgebaut. Im so konstruierten Baum werden an jeden Knoten Oberflächeneigenschaften wie Normalenrichtungen oder Materialkoeffizienten gespeichert. Diese Attribute werden als Mittelwert ihrer Nachfolger im Baum berechnet. Zur Darstellung wird der Baum traversiert, bis die projizierte Größe einer Hüllkugel einen vorher festgelegten Grenzwert unterschreitet oder ein Blatt erreicht ist. Der jeweilige Knoten wird durch einen Punkt, dessen Größe durch die Hüllkugel und dessen Farbe durch die gespeicherten Oberflächeneigenschaften festgelegt ist, dargestellt.

In [CHEN und NGUYEN 2001] wird die schon in [WAND et al. 2001] vorgestellte Optimierung für die Darstellung „großer“ Dreiecke auch für das QSplat Verfahren eingeführt: An Stelle von Oberflächenpunkten speichert der *POP*-Ansatz an den Blättern des Baums Dreiecke. Hierdurch können Dreiecke mit einer großen projizierten Fläche direkt mit Hilfe von Grafikhardware dargestellt werden. Dies ist in der Regel effizienter als eine Darstellung aus einzelnen Punkten und erhöht die Bildqualität, da die Geometrie in ihrer ursprünglichen Form dargestellt werden kann.

Surfels

Die von Pfister et al. [PFISTER et al. 2000] benutzte Szenenrepräsentation speichert eine Menge von Oberflächenpunkten in einer Hierarchie von geschichteten Tiefenbildern. Die Oberflächenpunkte werden in einem Ray Casting-Schritt gewonnen und in drei paarweise orthogonal zueinander stehende geschichtete Tiefenbilder (LDIs), einem *Layered Depth Cube (LDC)* [LISCHINSKI und RAPPOPORT 1998], gespeichert. Aus den LDCs werden danach mittels spezialisierter Filterverfahren Point Sample-Repräsentationen mit niedrigeren Auflösungen erzeugt. Auf diese Weise kann für die Darstellung am Bildschirm immer die optimale Auflösung gewählt werden. Ein verbessertes Darstellungsverfahren [ZWICKER et al. 2001] erlaubt die anisotrope Texturfilterung, Entfernung verdeckter Flächen und die korrekte Darstellung transparenter Flächen, unabhängig von der Reihenfolge ihrer Darstellung.

Der Nachteil von QSplat und Surfels ist bei beiden Verfahren, dass die Auflösung der Datenstrukturen a priori festgelegt ist, da die verwendeten Point Sample-Datenstrukturen in einem Vorverarbeitungsschritt erzeugt werden. Hierdurch ist die beste erreichbare Bildqualität immer durch die höchste verfügbare Auflösung der Daten begrenzt. Um auch für Betrachterpositionen nahe am Objekt eine gute Bildqualität zu erzielen, muss die Auflösung entsprechend hoch gewählt und damit eine große Menge an Daten gespeichert werden. Eine dynamische Erzeugung der Point Sample-Repräsentation wie in [WAND et al. 2001] oder [STAMMINGER und DRETTAKIS 2001] vermeidet dieses Problem.

2.4.9 Lichtfelder

In [LEVOY und HANRAHAN 1996, GORTLER et al. 1996] werden Datenstrukturen beschrieben, die, betrachtet man die Skala der heute verfügbaren bildbasierten Datenstrukturen und Darstellungsverfahren, das Konzept einer Szenenbeschreibung als Stichprobe der plenoptischen Funktion am konsequentesten umsetzen. *Light Field* bzw. *Lumigraph* besitzen im Gegensatz zu den meisten der oben beschriebenen Datenstrukturen keinerlei Einschränkungen in Bezug auf Geometrie, Material oder optische Eigenschaften der gespeicherten Objekte. Beide Datenstrukturen erlauben die Rekonstruktion neuer Ansichten beliebiger Objekte für fast beliebige Betrachterpositionen. Dabei ist der Aufwand für die Berechnung eines Bildes unabhängig von der Geometrie und damit der Komplexität der dargestellten Szene.

Light Field bzw. Lumigraph werden in Abschnitt 4.1 näher beschrieben.

2.5 Weiterführende Arbeiten

Bildbasierte Datenstrukturen speichern eine Teilmenge der plenoptischen Funktion. Diese wiederum beschreibt die Leuchtdichte in einer Szene für jeden Ort, jede Richtung, jeden Zeitpunkt und jede Wellenlänge. Eine aktive Veränderung des so beschriebenen Zustandes ist in diesem Konzept nicht vorgesehen, in vielen Fällen aber wünschenswert und sinnvoll. Ziel weiterführender Arbeiten ist deswegen die effiziente Erfassung, Speicherung und Verarbeitung der Reflexionseigenschaften von synthetischen und realen Objekten. Dies erlaubt beispielsweise die (interaktive) Veränderung von Szenen, die ursprünglich nur als Fotografie vorlagen.

Messung von Beleuchtung und Hinzufügen von Objekten

In [DEBEVEC und MALIK 1997] wird ein Verfahren zur Messung der Umgebungsbeleuchtung vorgestellt. Hierzu wird mit unterschiedlicher Belichtung eine Anzahl von Fotografien von spekular reflektierenden Kugeln angefertigt. Diese können danach genutzt werden, um Computer-generierte Objekte nachträglich in reale Umgebungen einzufügen [DEBEVEC 1998].

Veränderung der Beleuchtung

In einem Lichtfeld werden gewöhnlich Bilder, die durch die Bewegung einer Kamera in zwei Freiheitsgraden gewonnen wurden, gespeichert. Katayama et al. [KATAYAMA et al. 1999b, KATAYAMA et al. 1999a] speichert statt dessen Szenenansichten, die sich bei Bewegung des Betrachters und einer Lichtquelle mit jeweils einem Freiheitsgrad ergeben. Mit diesem einfachen Ansatz kann das gespeicherte Objekt mit unterschiedlichen Beleuchtungen dargestellt werden. Der Speicherbedarf wächst, wie beim herkömmlichen Lichtfeld, mit $O(n^4)$. Ein ähnlicher Ansatz wird in [MAGDA et al. 2000] beschrieben.

Verallgemeinerte Beschreibungen

Polynomial Texture Maps [MALZBENDER et al. 2001] speichern im Gegensatz zu herkömmlichen Texturen das Bild einer Objektoberfläche für unterschiedliche Beleuchtungssituationen. Hierzu werden für jeden Pixel die Koeffizienten eines Polynoms, das die Veränderung der reflektierten Helligkeit in Abhängigkeit von der Richtung und Intensität des Lichteinfalls beschreibt, gespeichert. Diese sparsame Repräsentation erlaubt allerdings nur die Rekonstruktion von Texturen für einen festen Betrachterstandort.

Allgemein werden die Reflexionseigenschaften eines Punktes durch die *Bidirectional Reflection Distribution Function (BRDF)* beschrieben. Diese ist abhängig von den Richtungen des ein- und ausfallenden Lichts, sowie von den Materialeigenschaften. Somit werden zur Speicherung von Reflexionseigenschaften in einer allgemeinen Form höherdimensionale Datenstrukturen benötigt. Sind die Reflexionseigenschaften bekannt, können diese genutzt werden, um die Beleuchtung von beliebigen Szenen nachträglich zu ändern [LOSCOS et al. 2000, BOIVIN und GAGALOWICZ 2001] oder komplexe Materialien fotorealistisch darzustellen [XU et al. 2001].

Zahlreiche Arbeiten beschreiben Verfahren zur Rekonstruktion der BRDF mit Hilfe spezieller Vorrichtungen [SCHIRMACHER et al. 1999a, DEBEVEC et al. 2000, MALZBENDER et al. 2001] oder aus Fotografien [YU und MALIK 1998, YU et al. 1999, RAMAMOORTHI und HANRAHAN 2001, BOIVIN und GAGALOWICZ 2001, LIU et al. 2001]. Bei den letztgenannten Verfahren wird immer vorausgesetzt, dass die (approximierte) Geometrie der Szene bekannt ist. In [WONG et al. 1997b, WONG et al. 1997a, WONG et al. 2001] wird ein Verfahren zur Veränderung der Beleuchtung in bildbasiert modellierten Szenen beschrieben, das ohne Angabe zur Szenengeometrie auskommt. Wong et al. fassen die Bildebene selbst als Szenenobjekt auf und speichern für jeden Punkt der Bildebene eine einfache Repräsentation der gemessenen BRDF.

Kapitel 3

Wavelet Kompression

Das in Kapitel 4 beschriebene Kompressionsverfahren für Lichtfelder basiert auf Wavelets. Deshalb werden nun die notwendigen Konzepte und Begriffe zur Approximation von Funktionen mittels hierarchischer Funktionsbasen erläutert. Nach einer allgemeinen Einführung in die Multi-Skalen-Analyse (MSA) im ersten Abschnitt, werden die wichtigsten Eigenschaften einer MSA erklärt (Abschnitt 3.2). Da ein Lichtfeld eine vierdimensionale Datenstruktur ist, wird in Abschnitt 3.3 ein Verfahren zur Konstruktion mehrdimensionaler Wavelets beschrieben. Abschließend wird ein Überblick über aktuelle Arbeiten zur Kompression von Bild- und Volumendaten sowie von Lichtfeldern mit Wavelets gegeben.

Wavelets sind Gegenstand zahlreicher Arbeiten in der Mathematik und können auf vielen Teilgebieten der Computergrafik nutzbringend eingesetzt werden. Hierzu gehören die Darstellung von Kurven und Freiformflächen, globale Beleuchtungsrechnung und nicht zuletzt die Kompression von Bilddaten. Ein guter Einstieg in dieses Thema ist in [STOLLNITZ et al. 1995a, STOLLNITZ et al. 1995b] enthalten. Anwendungen von Wavelets aus dem Bereich der Computergrafik werden in [STOLLNITZ et al. 1996, PIETREK 2001] beschrieben. Die in diesem Kapitel verwendeten Grundbegriffe aus der linearen Algebra sind in jedem Standardwerk (z. B. [FISCHER 1986, LANG 1997]) erklärt.

3.1 Multi-Skalen-Analyse

Die *Multi-Skalen-Analyse (MSA)* einer Funktion f erlaubt deren Darstellung in unterschiedlichen Auflösungen oder *Skalen*. Die Skalen werden mit 0 beginnend aufsteigend nummeriert. Skala 0 bezeichnet dabei die gröbste Auflösung. Zahlreiche im Folgenden definierte Begriffe und Bezeichnungen werden einer bestimmten Skala j zugeordnet. Dies wird durch den hochgestellten Index j ausgedrückt und darf nicht mit einer Potenzierung verwechselt werden.

3.1.1 Basen

Eine Folge verschachtelter Funktionsräume

$$V^0 \subset V^1 \subset V^2 \ldots \subset V^j \ldots$$

bildet die Basis jeder MSA. Aus der gegenseitigen Verschachtelung der Funktionsräume ergibt sich, dass die Auflösung m^j und somit die Dimension der Funktionsräume mit ansteigender Skala j wächst

$$m^0 < m^1 < m^2 \ldots < m^j \ldots$$

wobei m^j die Dimension von V^j bezeichnet.

Die Basisfunktionen $\phi_i^j, i = 0, \ldots, m^j - 1$ von V^j werden als *Skalierungsfunktionen* bezeichnet. Die *Wavelet-Räume* W^j bilden jeweils das Komplement von V^j in V^{j+1} in Bezug auf ein gewähltes inneres Produkt.

$$V^{j+1} = V^j \oplus W^j$$

Das bedeutet, dass jede Funktion $f \in V^{j+1}$ als Linearkombination von Funktionen aus V^j und W^j dargestellt werden kann. Die Wavelet-Räume werden von Basisfunktionen $\psi_i^j, i = 0, \ldots, n^j - 1$ aufgespannt, wobei n^j die Dimension des Wavelet-Raums W^j ist. Die Basisfunktionen ψ_i^j von W^j werden auch als *Waveletfunktionen* oder kurz *Wavelets* bezeichnet.

Aus den bisherigen Definitionen folgt

$$m^{j+1} = m^j + n^j$$

3.1.2 Beziehungen zwischen den Basen

Um die Darstellung zu vereinfachen und eine kompakte Schreibweise mathematischer Beziehungen zu ermöglichen, wird eine Matrixschreibweise für die Skalierungs- und Waveletfunktionen eingeführt. Die Basisfunktionen Φ^j und Ψ^j von V^j bzw. W^j werden als Zeilenvektoren

$$\begin{aligned} \mathbf{\Phi}^j &:= [\phi_0^j \cdots \phi_{m^j-1}^j] \\ \mathbf{\Psi}^j &:= [\psi_0^j \cdots \psi_{n^j-1}^j] \end{aligned}$$

der Skalierungs- bzw. Waveletfunktionen ϕ_i^j und ψ_i^j geschrieben.

Aufgrund der gegenseitigen Verschachtelung der Funktionsräume V^j kann jede Skalierungsfunktion $\phi^{j-1} \in V^{j-1}$ als Linearkombination von Basisfunktionen aus V^j dargestellt werden. Somit existiert eine Matrix $\mathbf{P}^j$ mit

$$\mathbf{\Phi}^{j-1}(x) = \mathbf{\Phi}^j(x)\, \mathbf{P}^j$$

$\mathbf{P}^j$ ist eine $m^j \times m^{j-1}$ Matrix.

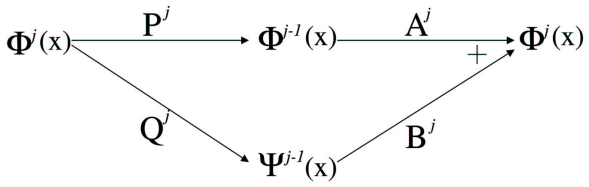


Abbildung 3.1: Darstellung der Zusammenhänge zwischen den verschiedenen Basen einer MSA.

Da W^{j-1} ein Unterraum von V^j ist, kann jedes Wavelet $\psi^{j-1} \in W^{j-1}$ ebenso als Linearkombination von Basisfunktionen aus V^j dargestellt werden. Somit existiert eine Matrix $\mathbf{Q}^j$ mit

$$\mathbf{\Psi}^{j-1}(x) = \mathbf{\Phi}^j(x)\,\mathbf{Q}^j$$

$\mathbf{Q}^j$ ist eine $m^j \times n^{j-1}$ Matrix.

Da V^{j-1} und W^{j-1} gemeinsam den Raum V^j aufspannen, läßt sich schließlich jede Skalierungsfunktion $\phi^j \in V^j$ als Linearkombination von Basisfunktionen $\phi_i^{j-1} \in V^{j-1}$ und $\psi_i^{j-1} \in W^{j-1}$ darstellen. Also existieren Matrizen $\mathbf{A}^j$ und $\mathbf{B}^j$, so dass gilt

$$\mathbf{\Psi}^j(x) = \mathbf{\Phi}^{j-1}(x)\,\mathbf{A}^j + \mathbf{\Psi}^{j-1}(x)\,\mathbf{B}^j$$

$\mathbf{A}^j$ ist eine $m^{j-1} \times m^j$- und $\mathbf{B}^j$ eine $n^{j-1} \times m^j$-Matrix.

Abbildung 3.1 zeigt noch einmal schematisch den Zusammenhang zwischen den Basen einer MSA. Insgesamt erfüllen die Matrizen $\mathbf{P}^j$, $\mathbf{Q}^j$, $\mathbf{A}^j$ und $\mathbf{B}^j$ folgende Bedingung

$$\left[\begin{array}{c}\mathbf{A}^j \\ \hline \mathbf{B}^j\end{array}\right] = [\mathbf{P}^j|\mathbf{Q}^j]^{-1}$$

Bei der benutzten *Block Matrix Notation* bilden die Matrizen über- bzw. nebeneinander geschrieben eine größere Matrix.

3.1.3 Dekomposition

Zu einer Funktion f kann nun mit Hilfe einer MSA eine Approximation $\tilde{f}^j$ konstruiert werden.

$$f(x) \approx \tilde{f}^j(x) = \sum_{i=0}^{m^j-1} c_i^j \phi_i^j(x)$$

Fasst man die Koeffizienten c_i^j in einem Spaltenvektor $\mathbf{c}^j$ zusammen

$$\mathbf{c}^j := \left[\begin{array}{c} c_0^j \\ \vdots \\ c_{m^j-1}^j \end{array}\right]$$

kann die Basisdarstellung auch kurz

$$\tilde{f}^j(x) = \mathbf{\Phi}^j(x)\mathbf{c}^j$$

geschrieben werden.

Die Koeffizienten $\mathbf{c}^{j-1}$ der nächst gröberen Skala $j-1$ der MSA können jetzt durch eine einfache Matrixmultiplikation berechnet werden.

$$\mathbf{c}^{j-1} = \mathbf{A}^j \mathbf{c}^j$$

Da die Dimension von m^{j-1} von V^{j-1} kleiner als die von V^j ist, geht bei diesem Berechnungsschritt zwangsläufig Information verloren. Wegen $V^j = V^{j-1} \oplus W^{j-1}$ können diese Detailinformationen aber in W^{j-1} dargestellt werden. Werden die entsprechenden *Detailkoeffizienten* $d_i^{j-1}, i = 0, \ldots, n^{j-1}-1$ als Vektor

$$\mathbf{d}^{j-1} := \begin{bmatrix} d_0^{j-1} \\ \vdots \\ d_{n^{j-1}-1}^{j-1} \end{bmatrix}$$

geschrieben, können die notwendigen Berechnungen wiederum in Matrixschreibweise dargestellt werden.

$$\mathbf{d}^{j-1} = \mathbf{B}^j \mathbf{c}^j$$

Insgesamt kann mit Hilfe der berechneten Skalierungs- und Detailkoeffizienten eine Darstellung von $\tilde{f}^j$ mit den Skalierungs- und Waveletfunktionen der Skala $j-1$ berechnet werden.

$$\begin{aligned} \tilde{f}^j(x) &= \sum_{i=0}^{m^j-1} c_i^j \phi_i^j(x) \\ &= \sum_{i=0}^{m^{j-1}-1} c_i^{j-1} \phi_i^{j-1}(x) + \sum_{i=0}^{n^{j-1}-1} d_i^{j-1} \psi_i^{j-1}(x) \\ &= \mathbf{\Phi}^{j-1}(x) \mathbf{c}^{j-1} + \mathbf{\Psi}^{j-1}(x) \mathbf{d}^{j-1} \end{aligned}$$

Die oben beschriebene Aufspaltung der Koeffizienten $\mathbf{c}^j$ in eine niedrig aufgelöste Version $\mathbf{c}^{j-1}$ und eine Menge von Detailkoeffizienten $\mathbf{d}^{j-1}$ wird *Wavelet-Dekomposition* oder kurz *Analyse* genannt. $\mathbf{A}^j$ und $\mathbf{B}^j$ heißen deswegen auch *Analyse-Filter*.

Die Dekomposition der Koeffizienten $\mathbf{c}^{j-1}$ kann rekursiv fortgesetzt werden, bis schließlich nur noch *Skalierungskoeffizienten* c_i^0 und eine Anzahl Detailkoeffizienten d_i^j übrig sind. Auf diese Weise erhält man die *Wavelet-Transformation* von $\tilde{f}^j$.

$$\begin{aligned} \tilde{f}^j(x) &= \sum_{i=0}^{m^0-1} c_i^0 \, \phi_i^0(x) + \sum_{k=0}^{j-1} \sum_{i=0}^{n^k-1} d_i^k \, \psi_i^k(x) \\ &= \mathbf{\Phi}^0(x) \, \mathbf{c}^0 + \sum_{k=0}^{j-1} \mathbf{\Psi}^k(x) \, \mathbf{d}^k \end{aligned}$$

3.1.4 Rekonstruktion

Die Wavelet-Transformation einer Funktion ist verlustfrei und umkehrbar, da es sich im Prinzip nur um einen Basiswechsel handelt. Deshalb können die Koeffizienten $\mathbf{c}^j$ leicht aus den Koeffizienten $\mathbf{c}^{j-1}$ und $\mathbf{d}^{j-1}$ rekonstruiert werden.

$$\mathbf{c}^j = \mathbf{P}^j \mathbf{c}^{j-1} + \mathbf{Q}^j \mathbf{d}^{j-1}$$

Dieser Vorgang wird als *Synthese* oder *Rekonstruktion* bezeichnet. Entsprechend heissen die verwendeten Matrizen $\mathbf{P}^j$ und $\mathbf{Q}^j$ *Synthese-Filter*.

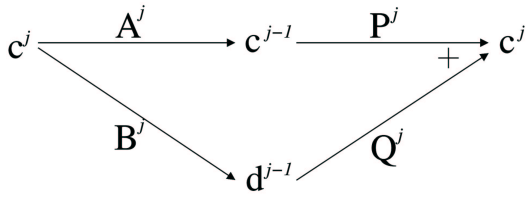


Abbildung 3.2: Berechnung der Koeffizienten während des Analyse- (linke Seite) und Syntheseschritts (rechte Seite).

Mittels Rekonstruktion können aus den während der Dekomposition gewonnenen Koeffizienten $\mathbf{c}^0$, $\mathbf{d}^0$, $\mathbf{d}^1$, …, $\mathbf{d}^j$ die Koeffizienten $\mathbf{c}^k$ einer Funktionsdarstellung $\tilde{f}^k$ für jede Skala $k, 0 \leq k \leq j$ berechnet werden. Hierzu wird der Rekonstruktionsschritt jeweils rekursiv auf die gerade gewonnenen Koeffizienten $\mathbf{c}^k$ und $\mathbf{d}^k$ angewandt. Abbildung 3.2 stellt Analyse- und Syntheseschritt nochmals schematisch dar.

3.2 Eigenschaften von Wavelets

Aus der Wavelet-Transformation einer Funktion können wie beschrieben unterschiedlich fein aufgelöste Repräsentationen gewonnen werden. Hierbei werden zu einem Grundsignal $\mathbf{c}^0$ sukzessive Detailinformationen, die in $\mathbf{d}^k$ gespeichert sind, hinzugefügt. Im Unterschied zur Fourier-Analyse enthält eine Wavelet-Transformation zusätzlich zu den Informationen über das Frequenzspektrum eines Signals auch Ortsinformation über das Auftreten der Frequenzen. Die Ortsinformation kann dabei an den Indizes i der $d_i^k \in \mathbf{d}^k$ abgelesen werden.

Die Eigenschaft einer Multi-Skalen-Analyse (MSA), gleichzeitig Orts- und Frequenzinformation darzustellen, wird durch die Benutzung von Basisfunktionen mit kompakten Trägern erreicht. Im Unterschied dazu werden in der Fourier-Analyse periodische Sinus- bzw. Cosinus-Funktionen mit unendlichem Träger als Basisfunktionen verwendet.

In der Regel kommen nicht alle Frequenzen an allen Stellen im Signal vor. Das bedeutet, dass viele der Detail-Koeffizienten d_i^k einen sehr geringen Wert haben oder Null sind. Dieser Umstand kann für eine (verlustbehaftete) Kompression des Signals ausgenutzt werden, denn durch Weglassen von Koeffizienten mit (geringem oder) einem

Wert gleich Null kann häufig ein großer Teil der Koeffizienten eingespart werden. Der auftretende Qualitätsverlust ist gering, da meist nur wenig Detailinformation verloren geht.

In einer MSA für Funktionen auf $\mathbb{R}$ haben alle Wavelets im Grunde die gleiche Form. Jede Waveletfunktion ψ_i^j geht aus einem einzigen *Mutterwavelet* ψ durch Verschiebung und Skalierung hervor

$$\psi_i^j(x) = \psi(2^j x - i)$$

Bei einer MSA auf einem endlichen Intervall müssen die Wavelets, deren Träger die Intervallgrenzen überlappen würden, speziell konstruiert werden, damit die oben definierten Eigenschaften einer MSA eingehalten werden.

Jede MSA, welche die oben gegebene Definition erfüllt, kann als Grundlage für eine Wavelet-Darstellung von Funktionen dienen. Bei der Konstruktion einer MSA gibt es bestimmte Freiheitsgrade, die genutzt werden können, um eine für die jeweilige Anwendung günstige MSA zu erhalten.

1. Die **Skalierungsfunktionen** $\mathbf{\Phi}^j$ legen die Eigenschaften der Funktionsräume V^j und die Synthese-Filter $\mathbf{P}^j$ fest.
2. Das benutzte **innere Produkt** wird zur Bestimmung der Orthogonalität und der Norm benutzt. Da mit Hilfe der Norm auch der Fehler der Approximation gemessen wird, kann es sinnvoll sein, das innere Produkt an die jeweilige Anwendung anzupassen.
3. Die **Wavelet-Funktionen** $\mathbf{\Psi}^j$ vervollständigen die MSA. Durch $\mathbf{\Psi}^j$ werden ausserdem die Synthese-Filter $\mathbf{Q}^j$ festgelegt.

Je nach Anwendung der MSA können unterschiedliche Eigenschaften von Bedeutung sein.

- Ein **kompakter Träger** der Basisfunktionen verringert den Aufwand bei Analyse und Synthese. Je kleiner die Träger der Skalierungs- und Waveletfunktionen sind, desto dünner sind meist die Analyse- und Synthesematrizen besetzt. Hierdurch sinkt der Aufwand einer Wavelet-Transformation.
- Die **Glattheit** der Basisfunktionen entscheidet über die Qualität der Approximation. Insbesondere wird durch glatte Basisfunktionen das Auftreten von störenden Artefakten, wie beispielsweise Machbandeffekten, vermieden. Allerdings wächst mit steigender Glattheit meist auch die Größe des Trägers.
- Die **Symmetrie** der Waveletfunktionen ist bei manchen Anwendungen gewünscht, um Verzerrungen zu vermeiden.
- Die Approximationsgenauigkeit eines Wavelets wird durch dessen **verschwindendes Moment (Vanishing Moment)** charakterisiert. Ein Wavelet $\psi(x)$ besitzt n verschwindende Momente, wenn $\int \psi(x)x^k = 0$ für $k = 0, \ldots, n-1$ aber nicht für $k = n$. Je höher das verschwindende Moment, desto besser können glatte Funktionen approximiert werden.

Wavelets können anhand ihrer Eigenschaften in Klassen eingeteilt werden.

3.2.1 Semiorthogonale Wavelets

Semiorthogonale Wavelets besitzen viele günstige Eigenschaften. Von diesen auch als *pre-wavelets* bezeichneten Basisfunktionen wird gefordert, dass alle Skalierungsfunktionen und Wavelets einer Skala paarweise senkrecht aufeinander stehen.

$$\langle \phi_i^k, \psi_j^k \rangle = 0 \qquad \text{für } i = 0, \ldots, m^k - 1 \qquad j = 0, \ldots, n^k - 1$$

Bei semiorthogonalen Wavelets ist bei Benutzung des Standard-Skalarprodukts die Darstellung auf einer Skala $i < j$ eine L^2-Approximation der Darstellung auf Skala j.

Die *Spline Wavelets* [FINKELSTEIN und SALESIN 1994] sind ein bekanntes Beispiel für semiorthogonale Wavelets. Als Skalierungsfunktionen werden die offenen B-Splines auf dem Intervall $[0; 1]$ mit äquidistantem Knotenvektor und vielfachen Endknoten verwendet. Die Wavelets werden so gewählt, dass sie einen minimalen Träger haben und die Synthesefilter $\mathbf{Q}^j$ somit möglichst dünn besetzt sind. Der Grad der verwendeten B-Splines kann frei bestimmt werden und hängt von der gewünschten Glattheit der Approximation ab. Allerdings wächst mit steigender Glattheit auch die Größe des Trägers und damit der Aufwand der Wavelet-Dekomposition.

Wie oben beschrieben ist die Anzahl der Einträge ungleich Null in den Analyse- und Synthesefiltern eine der wichtigsten Eigenschaften einer MSA, da der Aufwand von Analyse und Synthese linear mit der Anzahl der Einträge ungleich Null steigt. Nachteil der semiorthogonalen Konstruktion ist deshalb, dass die Analysefilter im Gegensatz zu den Synthesefiltern häufig nicht dünn besetzt sind und der Aufwand bei der Analyse unter Umständen quadratisch mit der Anzahl der Koeffizienten der analysierten Funktion wächst.

3.2.2 Biorthogonale Wavelets

Die *biorthogonale Wavelet-Konstruktion* erfordert statt der Orthogonalität von Wavelets und Skalierungsfunktionen einer Skala dieselbe Bedingung für die dualen Basen $\tilde{\mathbf{\Phi}}^j$ und $\tilde{\mathbf{\Psi}}^j$.

$$\langle \phi_i^k, \tilde{\psi}_j^k \rangle = 0 \qquad \text{für } i = 0, \ldots, m^k - 1 \qquad j = 0, \ldots, n^k - 1$$
$$\langle \psi_i^k, \tilde{\phi}_j^k \rangle = 0 \qquad \text{für } i = 0, \ldots, n^k - 1 \qquad j = 0, \ldots, m^k - 1$$

Diese Bedingung führt dazu, dass im Gegensatz zur semiorthogonalen Konstruktion sowohl die Analyse- als auch die Synthese-Filter dünn besetzt sind. Allerdings sind die Darstellungen zwischen den verschiedenen Skalen keine L^2-Approximationen mehr. Aus diesem Grund sind biorthogonale Wavelets für Anwendungen, welche die Darstellung eines Signals auf unterschiedlichen Auflösungsstufen benutzen, nur bedingt geeignet.

3.2.3 Orthonormale Wavelets

Die orthonormalen Wavelets zeichnen sich dadurch aus, dass alle Basisfunktionen normiert sind und paarweise zueinander senkrecht stehen.

$$\begin{aligned}
\langle \phi_i^k, \phi_j^k \rangle &= \delta_{i,j} && \text{für } i = 0, \ldots, m^k - 1 && j = 0, \ldots, m^k - 1 \\
\langle \psi_i^k, \psi_j^k \rangle &= \delta_{i,j} && \text{für } i = 0, \ldots, n^k - 1 && j = 0, \ldots, n^k - 1 \\
\langle \phi_i^k, \psi_j^k \rangle &= 0 && \text{für } i = 0, \ldots, m^k - 1 && j = 0, \ldots, n^k - 1
\end{aligned}$$

Die orthonormale Wavelet-Konstruktion ist somit ein Spezialfall der semiorthogonalen Wavelet Konstruktion.

Ein großer Vorteil dieser Klasse von Wavelets ist die einfache Berechnung des Approximationsfehlers, wenn Wavelet-Koeffizienten weggelassen werden. Er beträgt

$$\|f(x) - \tilde{f}(x)\|_2^2 = \sum_l d_l \tag{3.1}$$

wobei die d_l die weggelassenen Wavelet-Koeffizienten sind. Um eine Wavelet-Transformation, für die orthonormale Wavelets benutzt wurden, (verlustbehaftet) zu komprimieren, ist es am günstigsten Koeffizienten mit geringem Wert zu entfernen. Hierzu werden die d_i^j aufsteigend sortiert und so lange entfernt, bis eine festgelegte Fehlermarke überschritten wurde.

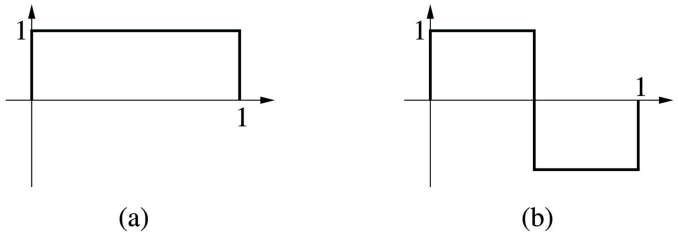


Abbildung 3.3: Haar-Skalierungsfunktion (a) und Haar-Wavelet (b).

Die Haar-MSA und ihre trivialen Variationen sind einige der wenigen orthonormalen Wavelet-Basen. Die Haar-MSA ist dabei die einzige, die auch symmetrisch ist. Sie besteht aus stückweise konstanten Funktionen und besitzt einen sehr kompakten Träger. Weiterhin ermöglicht die einfache Struktur der Haar-Wavelets in der praktischen Anwendung die Berechnung der Koeffizienten lediglich durch Additionen, Subtraktionen und Divisionen durch 2. All diese Operationen können sehr performant implementiert werden, da nur Integer-Arithmetik benötigt wird. Praktisch gesehen besteht die Berechnung einer Wavelet-Transformation mit der Haar-Basis darin, paarweise Mittelwerte und die Differenzen zu den Mittelwerten zu berechnen.

Tree-Wavelets sind eine spezielle Untermenge der orthonormalen Wavelets, bei denen sich die Träger aller Waveletfunktionen einer Skala nicht überlappen. Die Verwendung von Tree-Wavelets ist günstig, wenn nur bestimmte Werte aus einer Wavelet-Transformation rekonstruiert werden sollen. Im Gegensatz zu den im folgenden Abschnitt beschriebenen Daubechies-Wavelets sind Haar-Wavelets auch Tree-Wavelets.

Der kompakte Träger und die einfache Form der Haar-Wavelets (Abb. 3.3) bewirken, dass diese nicht glatt sind und nur ein verschwindendes Moment besitzen. Daubechies [DAUBECHIES 1988] entwickelte deshalb mit den *Daubechies-Wavelets* D eine Familie orthonormaler glatter Wavelets mit kompaktem Träger. Die D_4 Wavelet-Basis beispielsweise besitzt zwei verschwindende Momente und hat einen sehr kompakten Träger, so dass die Analyse- und Synthesefilter jeweils nur vier Koeffizienten ungleich Null pro Zeile bzw. Spalte enthalten. Die D_4 Skalierungs- und Waveletfunktionen sind asymmetrisch und stetig, aber an keiner Stelle differenzierbar. Ihre fraktale Form entsteht durch Anwendung eines Unterteilungsalgorithmus. Da mit der Anzahl der verschwindenden Momente eines Daubechies-Wavelets auch die Größe des Trägers wächst, überlappen sich im Unterschied zur Haar-Wavelet-Basis die Träger von Daubechies-Wavelets auf einer Skala gegenseitig.

3.2.4 Multiwavelets

Im Unterschied zu den bislang beschriebenen MSA, werden *Multiwavelets* [ALPERT 1993, ALPERT et al. 1993] nicht aus einer einzelnen Skalierungs- und Waveletfunktion konstruiert, sondern aus jeweils einem Vektor von Skalierungs- und Waveletfunktionen. Die hierdurch geschaffenen zusätzlichen Freiheitsgrade ermöglichen die Konstruktion von Wavelet-Basen mit nützlichen Eigenschaften wie Orthogonalität, einem kompakten Träger oder einer bestimmten Anzahl von verschwindenden Momenten. Wie oben beschrieben ist die Haar-Wavelet-Basis die einzig orthonormale und gleichzeitig symmetrische Wavelet-Basis. Haar-Wavelets besitzen allerdings nur ein verschwindendes Moment. Multiwavelets bieten hier einen Ausweg.

Wegen ihrer günstigen Eigenschaften werden Multiwavelets im *Wavelet Radiosity*-Verfahren [GORTLER et al. 1993] eingesetzt. Die dort benutzten *Flatlets* $\mathcal{F}_M$ [SCHRÖDER et al. 1994] und die auf Basis der ersten M Legendre Polynome konstruierten Multiwavelets $\mathcal{M}_M$ sind orthonormal. Für $M = 1$ sind $\mathcal{F}_1$ und $\mathcal{M}_1$ mit der Haar-Wavelet-Basis identisch. Multiwavelets können so konstruiert werden, dass sie eine beliebige Anzahl M von verschwindenden Momenten besitzen. Für $M > 1$ ermöglichen Multiwavelets daher eine glatte Approximation und sind gleichzeitig orthonormal. Die Flatlet-Basis $\mathcal{F}_M$ ist im Unterschied zu $\mathcal{M}_M$ aus stückweise konstanten Boxfunktionen zusammengesetzt (Abb. 3.4 (b)).

In Abbildung 3.4 sind die Skalierungs- und Waveletfunktionen von $\mathcal{M}_2$ und $\mathcal{F}_2$ dargestellt. Um die unterschiedlichen Funktionen $\phi_{i,k}^j$ und $\psi_{i,k}^j$ in den Vektoren der Skalierungs- und Waveletfunktionen zu kennzeichnen, wird ein zusätzlicher Index k verwendet. Wie bei der Haar-Wavelet-Basis überschneiden sich die Träger der Waveletfunktionen einer Skala für unterschiedliches k nicht gegenseitig und sind wie die Haar-Wavelets Tree-Wavelets.

3.3 Mehrdimensionale Wavelet-Konstruktion

Für die Erweiterung einer Multi-Skalen-Analyse auf mehrdimensionale Funktionen wird meist der Tensorprodukt-Ansatz gewählt. Zur Bildung der notwendigen Kombi-

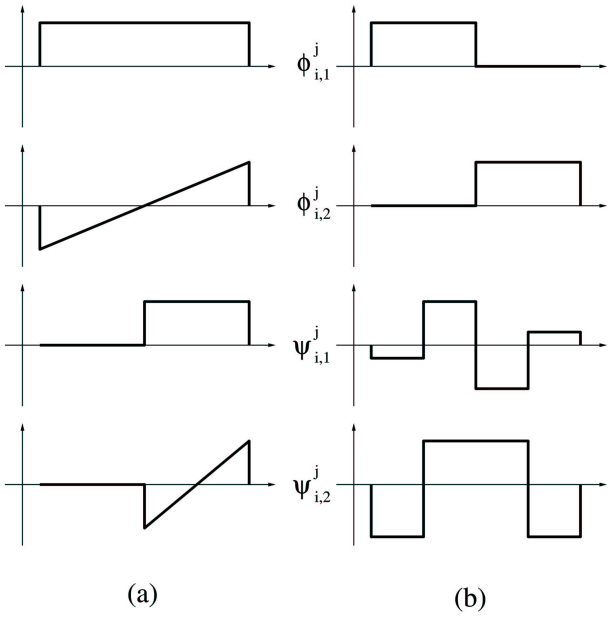


Abbildung 3.4: Skalierungs- und Waveletfunktionen der $\mathcal{M}_2$ Multiwavelets (a) und der Flatlets $\mathcal{F}_2$ (b).

nationen aus Skalierungs- und Waveletfunktionen einer eindimensionalen MSA stehen unterschiedliche Ansätze wie die *Line-*, *Quincunx-*, *Uniform-*, *Standard-* oder *Non-Standard*-Konstruktion [SALOMON 2000] zur Verfügung. Die Non-Standard-Konstruktion (auch *Pyramid*-Konstruktion) ist eine der populärsten Methoden, da die Qualität der Dekomposition in den meisten praktischen Anwendungen sehr hoch, die Darstellung symmetrisch und die mathematische Beschreibung einfach ist.

3.3.1 Non-Standard-Konstruktion

Bei der Non-Standard-Konstruktion einer n-dimensionalen MSA wird zunächst die Skalierungsfunktion definiert

$$\phi(x_0, \ldots, x_{n-1}) := \phi(x_0) \cdots \phi(x_{n-1})$$

Die n-dimensionalen Mutterwavelets werden durch systematische Kombination der eindimensionalen Skalierungsfunktion ϕ und des Mutterwavelets ψ gewonnen. Dabei müssen die kombinierten MSA nicht zwingend gleich sein. Das Mutterwavelet muss in jeder Kombination mindestens einmal vorkommen.

Zur Verdeutlichung wird nun die Non-Standard-Konstruktion an Hand der Konstruktion einer zweidimensionalen MSA beschrieben. Zunächst werden die zweidimensionale Skalierungsfunktion

$$\phi(x_0, x_1) := \phi(x_0)\phi(x_1)$$

und die drei Mutterwavelets definiert

$$\phi\psi(x_0, x_1) \quad := \quad \phi(x_0)\psi(x_1)$$

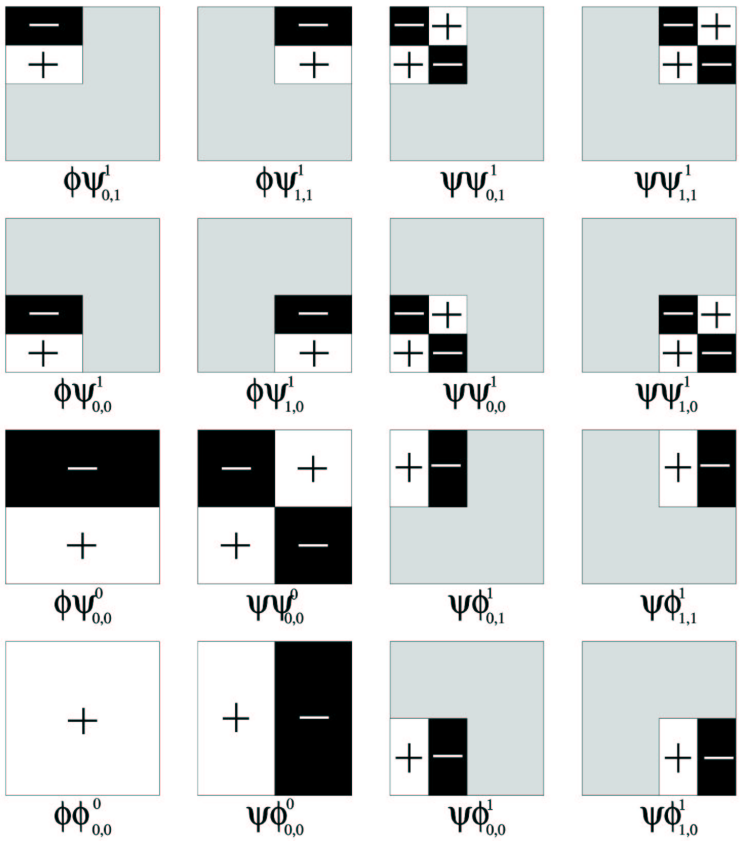


Abbildung 3.5: Skalierungs- und Waveletfunktionen der zweidimensionalen Haar-MSA für die Skalen 0 und 1 bei Non-Standard-Konstruktion.

$$\begin{aligned} \psi\phi(x_0, x_1) &:= \psi(x_0)\phi(x_1) \\ \psi\psi(x_0, x_1) &:= \psi(x_0)\psi(x_1) \end{aligned}$$

Alle Waveletfunktionen können nun durch Translation und Skalierung aus den Mutterwavelets konstruiert werden.

$$\begin{aligned} \phi\psi^j_{k,l}(x_0, x_1) &:= 2^j \phi\psi(2^j x_0 - k, 2^j x_1 - l) \\ \psi\phi^j_{k,l}(x_0, x_1) &:= 2^j \psi\phi(2^j x_0 - k, 2^j x_1 - l) \\ \psi\psi^j_{k,l}(x_0, x_1) &:= 2^j \psi\psi(2^j x_0 - k, 2^j x_1 - l) \end{aligned}$$

Die Faktoren 2^j normalisieren hierbei die Wavelets. In Abbildung 3.5 sind die Skalierungs- und Waveletfunktionen der durch Non-Standard-Konstruktion gewonnenen zweidimensionalen Haar-MSA für die Skalen 0 und 1 dargestellt.

Seien nun $c^j_{k,l}$ die Koeffizienten einer Wavelet-Transformation $\tilde{f}^j$ auf Skala j, für welche die konstruierten zweidimensionalen MSA benutzt wurden. Weiterhin sei $\mathbf{C}^j := [c^j_{k,l}]$ die zugehörige Koeffizientenmatrix. Zur Berechnung der Koeffizienten

der Skala $j-1$ wird auf jeder Zeile von $\mathbf{C}^j$ der eindimensionale Synthesefilter

$$\left[\frac{\mathbf{A}^j}{\mathbf{B}^j}\right]$$

angewandt. Die berechneten Skalierungs- und Detailkoeffizienten werden wiederum spaltenweise in eine Matrix geschrieben, auf die dann der Synthesefilter nochmals zeilenweise angewendet wird. Die so berechnete Koeffizientenmatrix enthält eine $m^{j-1} \times m^{j-1}$ große Untermatrix aus Skalierungskoeffizienten. Die restlichen Einträge enthalten die Detailkoeffizienten der Wavelet-Transformation. Zur Berechnung der vollständigen Wavelet-Transformation wird die beschriebene Transformation auf der Untermatrix mit den Skalierungskoeffizienten wiederholt rekursiv ausgeführt.

3.4 Kompression von Bilddaten mit Wavelets

Wavelets werden schon lange zur Kompression von Bilddaten eingesetzt. Entsprechende Verfahren sind Gegenstand zahlreicher Veröffentlichungen. Einen guten Überblick über den Stand der Technik geben [DAVIS und NOSRATINIA 1998, SAHA 2000]. Neben der Quantisierung der Wavelet-Koeffizienten ist die Kodierung der Information, welche der Wavelet-Koeffizienten wo im Datensatz gespeichert sind, eine der Hauptprobleme beim Entwurf eines Wavelet-basierten Kompressionsschemas. Auf die Quantisierung der Koeffizienten soll hier nicht weiter eingegangen werden.

Im Folgenden werden die wichtigsten auf Wavelets basierenden Kompressionsverfahren für herkömmliche zweidimensionale Bilddaten, Volumendatensätze und schließlich für Lichtfelder kurz besprochen. Die Vor- und Nachteile existierender Verfahren motivieren schließlich die Neuentwicklung des im nächsten Kapitel detailliert beschriebenen Kompressionsverfahrens für Lichtfelder.

3.4.1 Zweidimensionale Bilddaten

Der *SPIHT-Algorithmus (Set Partitioning on Hierarchical Trees)* von Said und Pearlman [SAID und PEARLMAN 1996] basiert auf dem *Embedded Zerotree Wavelet (EZW) Coding* [SHAPIRO 1993] und ist eines der bekanntesten Verfahren zur Speicherung und Übermittlung von Koeffizienten aus Wavelet-komprimierten Bilddaten.

Beim SPIHT-Verfahren erlaubt eine progressive Übertragung der Daten eine kontinuierliche Verbesserung der Bildqualität. Hierzu werden alle Wavelet-Koeffizienten entsprechend ihrer Betragsgröße sortiert und in absteigender Reihenfolge übermittelt. Durch die Sortierung der Wavelet-Koeffizienten entsprechend ihrer Größe kann deren Selbstähnlichkeit bei der Kodierung ausgenutzt werden.

Jede Bitebene der Wavelet-Koeffizienten wird einzeln, beginnend mit dem höchstwertigsten Bit (msb), übertragen. Für Koeffizienten, deren höchstwertigstes Bit sich auf der aktuellen Bitebene befindet, muss nur die Gesamtzahl der ‘1’-Bits und das Vorzeichen übertragen werden (Abb. 3.6). Für diejenigen Koeffizienten, deren höchstwertigstes Bit sich oberhalb der aktuellen Bitebene befindet, werden die Bitwerte der ak-

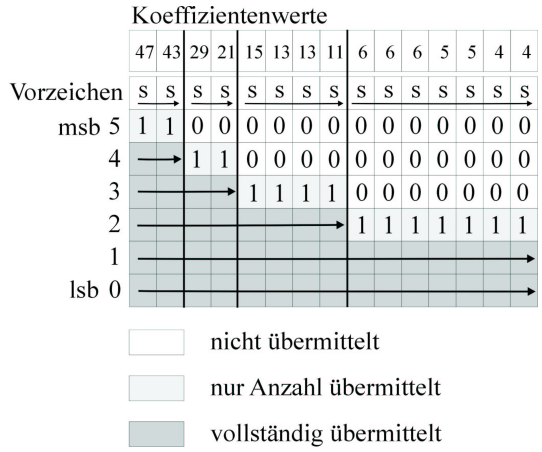


Abbildung 3.6: Schematische Darstellung der Kodierung von Wavelet-Koeffizienten im SPIHT-Verfahren: Beginnend mit dem höchstwertigsten Bit werden die Bitebenen einzeln übertragen. Bei Koeffizienten, deren höchstwertigstes Bit mit der aktuellen Bitebene übereinstimmt, werden nur die Anzahl der ‘1’-Bits und die Vorzeichen kodiert.

tuellen Bitebene sequentiell übertragen. Zusätzlich zu den eigentlichen Koeffizienten-Werten müssen die Koordinaten der Koeffizienten übertragen werden, da die Position der Koeffizienten im Bitstrom von deren betragsmäßigem Wert abhängt.

Um die Information über die Sortierung der Koeffizienten effizient zu kodieren, wird der SPIHT-Algorithmus benutzt. Dieser unterteilt die Koordinaten in verschiedene Mengen, mit denen die Sortierung durchgeführt wird. Zur Sortierung werden eine Anzahl von Fallunterscheidungen gemacht, deren Ergebnisse im Datenstrom übertragen werden. Zur Dekodierung des Bitstroms wird ein passender komplementärer Algorithmus verwendet. Der SPIHT-Algorithmus erlaubt auf diese Weise die Kodierung der Reihenfolge der Koeffizienten, ohne dass Koordinaten explizit übermittelt werden müssen.

3.4.2 Volumendaten

Erstmalig wurde die Anwendung von Wavelets auf Volumendatensätze von Muraki beschrieben [MURAKI 1992, MURAKI 1993]. Hierbei steht jedoch noch nicht der Aspekt der Kompression im Vordergrund, sondern eine Untersuchung der Möglichkeiten, die eine Multi-Skalen Darstellung für Volumendaten bietet. In [IHM und PARK 1998, IHM und PARK 1999, KIM und SHIN 1999] werden unterschiedliche Datenstrukturen beschrieben, welche den wahlfreien Zugriff auf einzelne Werte eines Wavelet-komprimierten Volumendatensatzes erlauben. Die Zeiten, die zur Darstellung einer Ansicht benötigt werden, sind allerdings bei sämtlichen Arbeiten nicht interaktiv. Rodler [RODLER 1999] schließlich faßt einen Volumendatensatz als Folge von Videobildern auf, die mit Techniken aus dem Bereich der Videodatenkompression bearbeitet werden. Die nach der Kompression verbleibenden Daten werden abschließend mit der Hilfe von Wavelets kodiert.

In [GUTHE und STRASSER 2001] wird ein Kompressionsverfahren für animierte Volumendatensätze beschrieben. Als einzige der in diesem Abschnitt beschriebenen Arbeiten erreicht Guthe et al. interaktive Darstellungszeiten von bis zu 12 Bildern pro Sekunde bei gleichzeitiger Dekompression der Daten. Durch die Kombination unterschiedlicher Kompressionsverfahren werden zudem sehr hohe Kompressionsraten ($1 : 200$) bei sehr geringen Fehler (PSNR $34,3$) erreicht.

3.4.3 Lichtfelder

In [IHM et al. 1997, PETER und STRASSER 1999, LALONDE und FOURNIER 1999, MAGNOR et al. 2000, PETER und STRASSER 2001b] wird die Benutzung von Wavelets zur Kompression von Lichtfeldern vorgeschlagen. Ihm et al. [IHM et al. 1997] benutzt allerdings nur eine zweidimensionale Wavelet-Transformation und verschenkt damit zwei Dimensionen im Lichtfeld, deren Kohärenz zur Kompression genutzt werden könnte. Im Unterschied dazu werden in [PETER und STRASSER 1999, LALONDE und FOURNIER 1999, MAGNOR et al. 2000, PETER und STRASSER 2001b] Kompressionsverfahren für Lichtfelder vorgestellt, die auf vierdimensionaler Wavelet-Kompression basieren. In [MAGNOR et al. 2000] wird der oben beschriebene SPIHT-Algorithmus für die Kompression von Lichtfeldern auf vier Dimensionen erweitert. Die guten erreichten Kompressionsraten erlauben allerdings keine interaktive Dekompression und Darstellung von Lichtfeldansichten [MAGNOR 2001]. Das Verfahren von Lalonde und Fournier [LALONDE und FOURNIER 1999] ermöglicht die interaktive Dekompression und Darstellung von Lichtfeldern mit geringer Auflösung von bis zu 32^4 Werten.

Erst die im nächsten Kapitel beschriebene *Waveletstream*-Datenstruktur [PETER und STRASSER 1999, PETER und STRASSER 2001b] erlaubt die progressive Übertragung, Speicherung, Dekompression und Darstellung von Multi-Skalen-Lichtfeldern in voller Auflösung. Weiterhin ermöglicht der Waveletstream die Speicherung zusätzlicher Information, wie beispielsweise Tiefenwerte, und implementiert eine spezielle Kodierung der Objektsilhouette zur Integration von Lichtfeldern in virtuelle Umgebungen.

Kapitel 4

Kompression von Lichtfeldern

Mit dem *Light Field* [LEVOY und HANRAHAN 1996] bzw. dem zeitgleich veröffentlichten *Lumigraph* [GORTLER et al. 1996] werden Datenstrukturen beschrieben, die eine vierdimensionale Abtastung der plenoptischen Funktion enthalten. Die meisten bildbasierten Datenstrukturen setzen voraus, dass das gespeicherte Objekt ideal diffus reflektierende Oberflächen besitzt. Im Gegensatz dazu besitzen Light Field und Lumigraph keinerlei Einschränkungen in Bezug auf Geometrie, Material oder optische Eigenschaften der gespeicherten Objekte. Beide Datenstrukturen erlauben die Rekonstruktion neuer Ansichten für fast beliebige Betrachterpositionen. Dabei ist der Aufwand für die Berechnung eines Bildes völlig unabhängig von der Geometrie und damit von der Komplexität der dargestellten Szene.

Allerdings sind die benötigten Datenmengen, die bei Abtastung der plenoptischen Funktion anfallen und abgespeichert werden müssen, sehr groß. Trotz des hohen Speicherplatzbedarfs hat die Darstellung von Szenen mittels Light Field bzw. Lumigraph gerade für Objekte mit komplizierter Geometrie und komplexem Reflexionsverhalten große Vorteile. Aus diesem Grund soll in diesem Kapitel ein Wavelet-basiertes Kompressionsverfahren vorgestellt werden, das durch eine speichereffiziente Repräsentation der plenoptischen Funktion die interaktive Darstellung hoch komplexer Szenen ermöglicht.

Im folgenden Abschnitt wird zunächst auf die bildbasierte Darstellung von Objekten mit Lichtfeldern eingegangen. Nach der Darstellung der notwendigen Grundlagen werden die Anforderungen an ein Kompressionsverfahren für Lichtfelder formuliert und ein entsprechender Ansatz gewählt. In Abschnitt 4.3 wird die entwickelte *Waveletstream*-Datenstruktur und das zugehörige Darstellungsverfahren im Detail beschrieben. Anschließend wird eine objektorientierte Implementierung der benötigten Algorithmen und Datenstrukturen vorgestellt. Das Kapitel schließt mit einer Darstellung der erzielten Resultate.

4.1 Lichtfelder

Obgleich sich Light Field und Lumigraph in Details unterscheiden, wird im Folgenden der Begriff „Lichtfeld“ synonym für „Light Field“ und „Lumigraph“ benutzt.

4.1.1 Aufbau und Darstellung

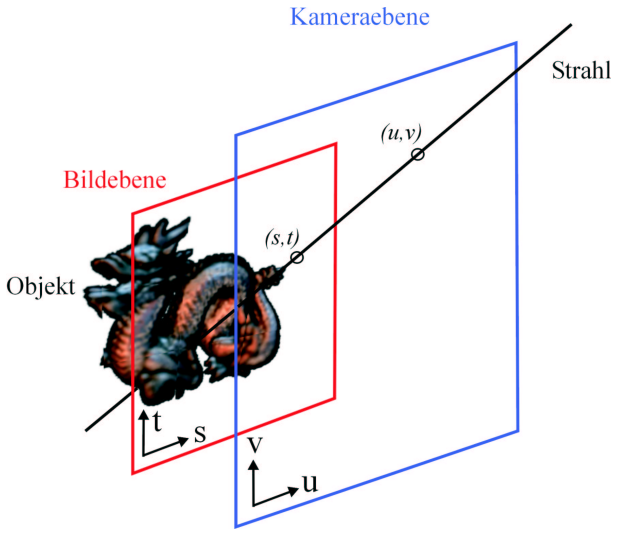


Abbildung 4.1: Zwei-Ebenen (2P) Parametrisierung eines Lichtfeld-Datensatzes. Jeder im Lichtfeld gespeicherte Strahl wird durch die Koordinaten, an denen er die *Kamera-* und die *Bildebene* (*uv-* und *st*-Ebene) schneidet, beschrieben.

Unter der Annahme, dass sich ein statisches Objekt im leeren Raum befindet, kann die ursprünglich siebendimensionale plenoptische Funktion (Abschnitt 2.2) eines Objekts in einem vierdimensionalen Datensatz gespeichert werden. In diesem wird die konstante Leuchtdichte aller Strahlen, die das Objekt schneiden, beispielsweise als RGB- oder YUV-Tupel gespeichert. Die Strahlen können, wie in [LEVOY und HANRAHAN 1996, GORTLER et al. 1996], mittels zwei parallel zueinander angeordneten Ebenen parametrisiert werden (*2P-Parametrisierung*, Abb. 4.1). Die 2P-Parametrisierung läßt sich dabei anschaulich so erklären, dass das Lichtfeld aus einer Menge Bilder besteht, deren Projektionszentren auf der *Kameraebene* liegen und deren *Bildebenen* mit der des Lichtfeldes identisch sind.

Der zu zwei bestimmten Ebenen gehörende Datensatz wird als *Lightslab* bezeichnet. Die Leuchtdichte einer Szene kann vollständig in einem Lichtfeld beschrieben werden, indem die Szene oder umgekehrt der Betrachter der Szene von einem Quader, der aus sechs Lightslabs besteht, umschlossen wird.

Zur Darstellung des im Lichtfeld gespeicherten Objekts müssen alle Strahlen, die den Augpunkt $\mathbf{e}$ des Betrachters schneiden, aus dem Lichtfeld ausgelesen und zu einer Ansicht zusammengesetzt werden. Hierzu wird in [LEVOY und HANRAHAN 1996] ein einfacher Algorithmus beschrieben, der vollständig in Software abläuft und nur auf die wirklich benötigten Lichtfeldwerte zugreift. Im Gegensatz dazu nutzt das von Gortler et al. [GORTLER et al. 1996] vorgeschlagene Verfahren Grafikhardware zur Beschleunigung der Darstellung. Mit Hilfe eines groben Geometriemodells erlaubt eine Tiefenkorrektur die Verbesserung der Bildqualität.

4.1.2 Weitere Arbeiten

Lichtfelder haben seit ihrer Veröffentlichung im Jahr 1996 eine Vielzahl weiterführender Arbeiten angeregt. Im Folgenden soll ein Überblick über die zahlreichen Verbesserungen, Variationen und Anwendungen von Lichtfeldern gegeben werden.

Erzeugung von Lichtfeldern

Um ein Lichtfeld zu erzeugen, ist eine Anzahl kalibrierter Bildern oder eine Videosequenz des zu speichernden Objekts notwendig. In [LEVOY und HANRAHAN 1996] und [WOOD et al. 2000] wird zur Erfassung der Position der Kamera ein Roboterarm, in [SHUM und HE 1999] ein Drehgestell benutzt. Für das *Digital Michelangelo Project* der Stanford University [LEVOY et al. 2000] wurden spezielle Vorrichtungen und Verfahren zur Erfassung von Geometriemodellen und Lichtfeldern von großen Skulpturen und anderen Kunstwerken entwickelt. Auch hier wird ein Roboterarm mit einer Kamera zur Erfassung eines großen Lichtfeldes benutzt. Gortler et al. [GORTLER et al. 1996] dagegen platziert das Objekt vor einem speziell gemusterten Hintergrund, der die Registrierung einzelner Ansichten erlaubt. Während der Benutzer die Kamera freihändig führt, wird er von einem Programm so gelenkt, dass eine gleichmäßige Erfassung des Objekts gewährleistet ist.

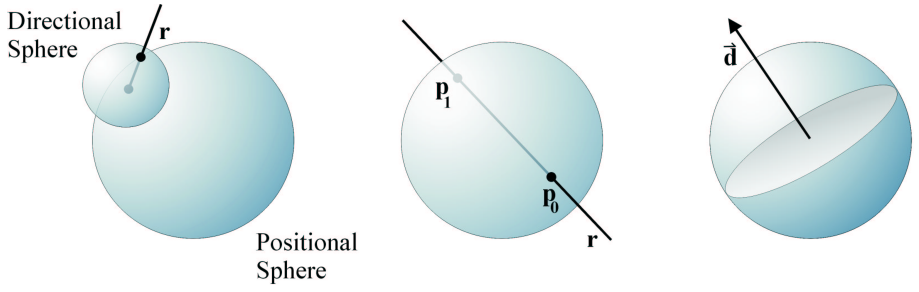
In verschiedenen Arbeiten werden Verfahren aus der Bildverarbeitung eingesetzt, um die Kameraparameter nur aus Merkmalen der Szene zu berechnen [KOCH et al. 1999b, KOCH et al. 1999a, HEIGL und NIEMANN 1999, VOGELSANG et al. 2000]. Hierdurch wird die Verwendung von speziellen Markierungen zur Bestimmung der Kameraposition wie in [GORTLER et al. 1996] überflüssig. Anstatt nur eine Kamera mit einem einzigen Objektiv zu verwenden, wird in [YANG et al. 2000] die preiswerte Konstruktion einer speziellen „Lichtfeld-Kamera“ beschrieben, die einen umgebauten Flachbrett-Scanner zur Erfassung von Lichtfeldern benutzt.

Bei der Erzeugung von Lichtfeldern aus Computer-generierten Bildern sind die Position und die Parameter der Kamera bekannt. Hier kann allerdings der Aufwand zur Berechnung der Vielzahl der benötigten Bilder ein Problem sein. Halle stellt mit *Multiple Viewpoint Rendering* [HALLE 1998] ein Verfahren vor, das durch Ausnutzung der Kohärenz zwischen benachbarten Ansichten die Berechnungen um mehrere Größenordungen beschleunigen kann.

Alternative Parametrisierungen

Die in [LEVOY und HANRAHAN 1996, GORTLER et al. 1996] vorgeschlagene 2P-Parametrisierung von Lichtfeldern ist in Bezug auf die Winkelauflösung nicht gleichmäßig. Deshalb sind in der Vergangenheit zahlreiche Vorschläge für alternative Lichtfeld-Parametrisierungen veröffentlicht worden.

In [IHM et al. 1997] wird das Lichtfeld-Objekt statt in einen Quader von Lightslabs in eine *Positional Sphere* eingeschlossen. Auf der Oberfläche der Positional Sphere werden danach *Directional Spheres* angeordnet, mit denen alle Strahlen $\mathbf{r}$, welche die Positional Sphere schneiden, parametrisiert werden (Abb. 4.2, links).



Ihm et al. 1997 | 2S-Parametrisierung | SP-Parametrisierung

Abbildung 4.2: Unterschiedliche Parametrisierungen für Lichtfelder: Ihm et al. [IHM et al. 1997] benutzt *Directional Spheres*, die mittels einer *Positional Sphere* parametrisiert werden, um die Richtungen der Strahlen zu speichern. Die *Two-Sphere Parameterization (2SP)* identifiziert jeden Lichtfeldwert an Hand der Schnittpunkte $\mathbf{p}_0$ und $\mathbf{p}_1$ des korrespondierenden Strahls $\mathbf{r}$ mit einer Kugel. Die *Sphere-Plane Parameterization (SPP)* speichert eine Anzahl von Parallelprojektionen des Objekts. Zur Parametrisierung der Richtungen $\vec{\mathbf{d}}$ der Projektionen wird eine Kugel benutzt.

Zur Speicherung der Lichtfeld-Daten werden Positional und Directional Spheres durch Dreiecksnetze approximiert. Beginnend mit einem Oktahedron werden dazu die Facetten mehrmals rekursiv in jeweils vier neue Facetten zerlegt. Camahort et al. stellt in [CAMAHORT et al. 1998] zwei weitere Parametrisierungen für Lichtfelder vor (Abb. 4.2, Mitte und rechts): Die *Two-Sphere Parameterization (2SP)* identifiziert jeden Lichtfeldwert anhand der Schnittpunkte $\mathbf{p}_0$ und $\mathbf{p}_1$ des korrespondierenden Strahls $\mathbf{r}$ mit einer Kugel. Bei der *Sphere-Plane Parameterization (SPP)* besteht das Lichtfeld aus einer Menge von Parallelprojektionen des Objekts. Die Ausrichtung $\vec{\mathbf{d}}$ jeder Projektion wird über eine Kugel parametrisiert. Zur Speicherung der Daten werden bei beiden Verfahren die verwendeten Kugeln, ähnlich wie in [IHM et al. 1997], durch Dreiecksnetze approximiert. Lalonde und Fournier [LALONDE und FOURNIER 1999] organisieren das Lichtfeld in einer Menge von Hemisphären über einer Ebene. Zur Parametrisierung der Hemisphären wird dabei das Nusselt-Analogon [SIEGEL und HOWELL 1981] benutzt. Schließlich wird in [TSANG et al. 1998] die Verwendung von Polarkoordinaten in einem Ansatz ähnlich zur SPP vorgeschlagen. Auf Grund der verwendeten trigonometrischen Funktionen bei der Koordinatenberechnung ist jedoch ein hoher Rechenaufwand bei der Abfrage von Lichtfeldwerten zu erwarten. Außerdem bleibt unklar, wie das Intervall der Polarkoordinaten gleichmäßig abgetastet werden soll.

Shum und He [SHUM und HE 1999] speichern nur eine dreidimensionale Repräsentation der plenoptischen Funktion. Zur Erzeugung von *Concentric Mosaics* wird eine Schlitzkamera zur Aufnahme von Panoramabildern nicht wie beim Lichtfeld auf einer Ebene, sondern lediglich entlang einer rotierenden Geraden geführt. Dies reduziert den Speicherplatzbedarf im Vergleich zu vierdimensionalen Lichtfeldern. Bewegt sich der Betrachter allerdings aus der Rotationsebene der Geraden heraus, muss die perspektivische Verzerrung der Panoramen, ähnlich wie beim Image Warping (Abschnitt 2.4.3), korrigiert werden. In [LEE und LEE 2000] wird eine veränderte Parametrisierung für

Concentric Mosaics vorgeschlagen, um deren Erzeugung zu erleichtern. Schließlich wurde von Shen et al. [SHEN et al. 2001] kürzlich eine Erweiterung dieser Technik zu *Spherical Mosaics* vorgestellt.

Die streng reguläre Anordnung von Kamerapositionen und Bildebenen der bislang beschriebenen Datenstrukturen wird in [BUEHLER et al. 2001] und [SCHIRMACHER et al. 2001] zu Gunsten eines flexibleren Konzepts aufgegeben. In beiden Arbeiten werden Verfahren beschrieben, welche die Rekonstruktion von Ansichten einer Szene aus einer Kollektion von Bildern erlauben. Hierbei können die Bilder mit beliebig positionierten Kameras aufgenommen werden. Es wird jeweils eine angenäherte Geometrie der Szene benutzt, um die Rekonstruktion zu verbessern. Leider steigt der Aufwand des Verfahrens von Buehler et al. [BUEHLER et al. 2001] linear mit der Anzahl der Eingabebilder. Da die Anzahl der Eingabebilder auch von den Eigenschaften der Szene abhängt, geht mit dieser Abhängigkeit auch einer der Hauptvorteile von IBR gegenüber herkömmlicher Computergrafik verloren. Die von Schirmacher et al. [SCHIRMACHER et al. 2001] vorgestellten *Generalized Lumigraphs* beschränken dagegen den Aufwand dadurch, dass für jede Betrachterposition die Quellbilder, die zur Darstellung benötigt werden, mit einer einfachen Heuristik bestimmt werden. Hierdurch wird deren Anzahl begrenzt und die Laufzeit des Verfahrens unabhängig von Eigenschaften der Szene.

In [MILLER et al. 1998] und [WOOD et al. 2000] wird schließlich die Idee der Unabhängigkeit des Lichtfeldes von der Geometrie des gespeicherten Objekts aufgegeben. In beiden Arbeiten wird ein (vereinfachtes) Modell des gespeicherten Objekts verwendet, um direkt auf dessen Oberfläche ein *Surface Light Field* zu parametrisieren. Allgemein ermöglicht die Benutzung geometrischer Information die Verringerung des Darstellungsfehlers [GORTLER et al. 1996, CHAI et al. 2000]. In diesem Fall wird sie zudem genutzt, um die Objektgeometrie interaktiv zu verändern [WOOD et al. 2000]. Die größten Nachteile der beschriebenen Verfahren sind, dass der Aufwand bei der Darstellung nicht mehr unabhängig vom dargestellten Objekt ist, eine Erfassung oder Modellierung der Objektgeometrie zwingend notwendig wird und, ähnlich wie bei Texturen und Bump Maps (Abschnitt 2.4.1), die Verwendung von Surface Light Fields an den Objektsilhouetten sichtbar werden kann.

Verbesserung der Darstellung

Das von Gortler beschriebene Verfahren zur Tiefenkorrektur (siehe Abschnitt 4.3.10) in Lumigraphen wird von Isaksen et al. in [ISAKSEN et al. 2000] verallgemeinert. *Dynamically Reparameterized Light Fields* erlauben die Darstellung von Lichtfeldern entsprechend beliebig platzierter Schärfeebenen.

In [SLOAN et al. 1997] werden Möglichkeiten zur Beschleunigung der Darstellung von Lichtfeldern vorgestellt. In [REGAN et al. 1999] wird eine Hardware beschrieben, welche die Darstellung von Lichtfeldern mit nur sehr geringer Latenzzeit erlaubt. Zur Verringerung der zu verarbeitenden Datenmenge werden nur dreidimensionale Lichtfelder verwendet, was die Bewegung des Betrachter auf einen Freiheitsgrad einschränkt. Schließlich werden in [SLOAN und HANSEN 1999] drei unterschiedliche Ansätze zur schnellen Lichtfeld-Darstellung durch Parallelisierung beschrieben.

Kompressionsverfahren

Die Datenmengen, die bei der Abtastung der plenoptischen Funktion im Lichtfeld gespeichert werden müssen, sind sehr groß. Ein Lichtfeld, das Echtfarben (24 Bit) benutzt und aus sechs einzelnen Lightslabs besteht, hat bei einer Auflösung von nur $256 \times 256 \times 32 \times 32$ Abtastpunkten pro Lightslab eine unkomprimierte Größe von $1,25$ GByte. Für die praktische Nutzung von Lichtfeldern sind leistungsfähige Kompressionsverfahren deshalb unverzichtbar.

Es wurden verschiedene Ansätze zur Kompression von Lichtfeldern veröffentlicht. In ihrer ursprünglichen Arbeit benutzen Levoy and Hanrahan [LEVOY und HANRAHAN 1996] *Vektor-Quantisierung (VQ)* mit einer anschließenden Lempel-Ziv Kodierung [ZIV und LEMPEL 1977, ZIV und LEMPEL 1978] der Daten. Der Nachteil dieses Ansatzes ist, dass das gesamte Lichtfeld vor der Nutzung dekomprimiert werden muss, da Lempel-Ziv Kodierung keinen wahlfreien Zugriff auf einzelne Werte erlaubt. Unterschiedliche Verfahren werden zur Kompression von Surface Light Fields angewandt. Wood et al. schlägt in [WOOD et al. 2000] eine Verallgemeinerung von Vektor-Quantisierung bzw. Principal Component Analysis vor. Im Gegensatz dazu unterteilt Miller et al. [MILLER et al. 1998] das Surface Light Field in Blöcke, die nach einer diskreten Cosinus-Transformation Huffman-kodiert werden. Ein Caching-Schema nutzt die Kohärenz im Zugriff auf die Daten und unterstützt die Wiederverwendung dekodierter Blöcke zur Erzeugung von Szenenansichten mit ähnlicher Betrachterposition.

Anstatt die Größe eines Lichtfeldes direkt durch Kompression zu verringern, beschreibt Schirmacher et al. in [HEIDRICH et al. 1999b, SCHIRMACHER 2000, SCHIRMACHER et al. 2000, SCHIRMACHER et al. 1999b] ein spezielles Verfahren, bei dem möglichst wenige Ansichten des Objekts im Lichtfeld gespeichert werden. Bei der Konstruktion des Lichtfeldes wird mit Hilfe eines speziellen Fehlermaßes entschieden, ob eine Lichtfeld-Ansicht für eine bestimmte Kameraposition aus den schon gespeicherten Bildern konstruiert werden kann oder die zusätzliche Ansicht im Lichtfeld gespeichert werden muss. Zur Erzeugung einer Lichtfeld-Ansicht aus schon gespeicherten Bildern wird ein Image Warping-Verfahren benutzt, für das zusätzliche Informationen über die Szenengeometrie benötigt werden. Hierzu kann eine approximierte Objektgeometrie, wie sie beispielsweise in einem Lumigraphen gespeichert ist [GORTLER et al. 1996], benutzt werden.

In [KIU et al. 1998, MAGNOR 2000] werden Kompressionsverfahren für Lichtfelder beschrieben, die auf Techniken aus dem Bereich der Video-Kompression beruhen. Das zu komprimierende Lichtfeld wird hierzu in Blöcke unterteilt, von denen eine Teilmenge ausgewählt wird. Aus diesen können mit Hilfe einer geringen Menge an zusätzlicher Information alle anderen Blöcke rekonstruiert werden [MAGNOR und GIROD 1999a]. Durch Geometrieinformation, die aus dem Lichtfeld gewonnener wird [EISERT et al. 1999], kann die Rekonstruktion und damit das Kompressionsverhältnis weiter verbessert werden [MAGNOR und GIROD 1999b, MAGNOR und GIROD 2000]. Leider lassen diese Ansätze im Unterschied zum in Abschnitt 4.3 beschriebenen Waveletstream-Kompressionsverfahren eine interaktive Dekompression und Darstellung von Lichtfeldern nicht zu [MAGNOR 2001]. Ein Verfahren, der dem Magnors ähnlich ist, wird von Xin et al. in [TONG und GRAY 1999,

TONG und GRAY 2000] beschrieben, erreicht aber nicht die Kompressionsraten, wie in den Arbeiten von Magnor et al.

Andere Anwendungen

Heidrich et al. [HEIDRICH et al. 1999a] speichert in einem Lichtfeld an Stelle von Farbwerten für jeden Sichtstrahl die zugehörige Reflexionsrichtung. Auf diese Weise kann die spekulare Reflexion komplexer Objekte interaktiv berechnet und dargestellt werden. Mit einem ähnlichen Ansatz simuliert Heidrich et al. [HEIDRICH et al. 1997] auch das Verhalten von optischen Linsen. In [HEIGL et al. 1998] wird die Anwendung von Lichtfeldern im Bereich der automatischen Objekterkennung beschrieben. Kawasaki et al. [KAWASAKI et al. 2001] benutzt Surface Light Fields zur realistischen Darstellung der Oberfläche animierter Objekte.

Zusammenfassend kann gesagt werden, dass trotz des großen Speicherplatzbedarfs die Darstellung von Szenen mittels Lichtfeldern anderen Szenenrepräsentationen überlegen ist. Denn ist die plenoptische Funktion hinreichend genau abgetastet, kann für fast beliebige Betrachterpositionen ein korrektes Bild der Szene mit einem Berechnungsaufwand erzeugt werden, der unabhängig vom Szeneninhalt ist. Der Aufwand ist unabhängig davon,

- wie die Lichtfeld-Daten gewonnen wurden, d.h. durch Rendering synthetischer Szenen, aus Fotografien oder Videos realer Szenen,
- ob komplizierte optische Phänomene, wie z.B. vielfache Reflexionen, Brechung, diffuse Spiegelung oder Transparenz, auftreten und
- wie komplex die Geometrie der betrachteten Szene ist.

4.2 Anforderungen an ein Kompressionsverfahren für Lichtfelder

Bei der Wahl eines geeigneten Kompressionsverfahrens für Lichtfelder müssen unterschiedliche Aspekte betrachtet werden. Folgende Anforderungen müssen an ein leistungsfähiges Kompressions- und zugehöriges Darstellungsverfahren gestellt werden.

- Obgleich bei Verwendung der 2P-Parametrisierung, wie in [LEVOY und HANRAHAN 1996, GORTLER et al. 1996], die zur Darstellung einer Ansicht benötigten Werte einen zweidimensionalen affinen Unterraum im Lichtfeld bilden [GU et al. 1997], sind die Zugriffsmuster während der Berechnung einer Lichtfeld-Ansicht weitgehend zufällig. Dies ist leicht dadurch zu erklären, dass bei Abfrage der zweidimensionalen Wertemenge nur zwei Freiheitsgrade im Lichtfeld-Datensatz $\mathcal{L}$ verbleiben, die Position des Betrachters aber drei Freiheitsgrade besitzt. Somit ist es schwierig, die Lichtfeld-Daten so zu strukturieren, dass Kohärenz zur Beschleunigung der Werteabfrage genutzt

werden kann. Insbesondere kann aus der Tatsache, dass ein bestimmter Wert $\mathcal{L}(s,t,u,v)$ abgefragt wird, nicht auf die Benutzung bestimmter benachbarter Werte geschlossen werden.

Ein Kompressionsschema muss **wahlfreien und gleich schnellen Zugriff** auf alle Werte des komprimierten Lichtfeldes erlauben.

- Eine Alternative zum wahlfreien Zugriff auf die komprimierten Lichtfeldwerte wäre die vollständige Dekompression des Lichtfeldes vor dessen Benutzung. Wie aber in Abschnitt 4.1.2 dargestellt, sind die Datenmengen, die selbst in einem gering aufgelösten Lichtfeld gespeichert werden müssen, sehr groß. Aus diesem Grund kommt eine vollständige Dekompression eines Lichtfeldes meist nicht in Frage. Somit muss ein Kompressionsschmema für Lichtfelder den Zugriff auf und die Rekonstruktion von beliebigen Einzelwerten im Lichtfeld in Zeiten ausreichend für **interaktive Anwendungen** erlauben.

Neben den oben stehenden Anforderungen, die unverzichtbar für ein praktisch anwendbares Verfahren sind, gibt es weitere wünschenswerte Eigenschaften.

- In Kapitel 2 wird der Zusammenhang zwischen traditioneller geometriebasierter Computergrafik und bildbasierten Darstellungsmethoden beschrieben. Um die Vorteile beider Welten nutzen zu können, müssen bildbasierte Objekte in geometrisch modellierte Szenenbeschreibungen integriert werden können. Damit dabei die gegenseitige Verdeckung der Szenenobjekte korrekt rekonstruiert werden kann, sollten Informationen zur **Unterscheidung von Objekt und Hintergrund** gespeichert werden. Weiterhin ist die Speicherung einer geringen Menge von **Geometrieinformation** sinnvoll, um die Tiefe eines bildbasierten Objekts (annähernd) bestimmen zu können.

- Zur Vermeidung von Alias-Effekten ist bei der Darstellung von Objekten auf dem Bildschirm häufig eine **Anpassung der Auflösung** erforderlich. Diese Fähigkeit wird in der Computergrafik oft benötigt und trägt erheblich zur Steigerung der Darstellungsqualität bei.

4.3 Die *Waveletstream*-Datenstruktur

Zur Verwirklichung der oben definierten Anforderungen wird in diesem Abschnitt ein auf Wavelet-Kompression basierendes Kompressionsschema für Lichtfelder vorgestellt. Die neue *Waveletstream*-Datenstruktur (kurz *Waveletstream*) speichert Wavelet-Koeffizienten und erlaubt den wahlfreien Zugriff auf diese. Das zugehörige Dekompressions- und Darstellungsverfahren ermöglicht die interaktive Darstellung von komprimierten Lichtfeldern in unterschiedlichen Auflösungsstufen. Zusätzliche Informationen unterstützen die Integration von Lichtfeldern in geometrisch modellierten Szenen.

4.3.1 Entscheidungsgründe

Folgende Gründe haben zur Auswahl dieses Ansatzes und zur Entwicklung des Waveletstream in der hier beschriebenen Form geführt:

1. Die Darstellung eines Signals in einer Wavelet-Basis ermöglicht dessen **(verlustbehaftete) Kompression** und gleichzeitig eine einfache Darstellung auf **unterschiedlichen Skalen**.

2. Die Berechnung einer Lichtfeld-Ansicht ist gleich bedeutend mit der Projektion einer zweidimensionalen Teilmenge des vierdimensionalen Lichtfeldes auf eine Ebene im vierdimensionalen Raum. Die Tatsache, dass für die Darstellung einer Lichtfeld-Ansicht nur eine zweidimensionale Teilmenge der Daten benötigt wird, macht eine Anpassung von Kompressionsverfahren für zweidimensionale Bild- oder dreidimensionale Volumendaten (Abschnitt 3.4) für die Kompression von Lichtfeldern schwierig. Diesen Verfahren liegt nämlich die durchaus vernünftige Annahme zu Grunde, dass bei der Dekompression der gesamte Datensatz von Interesse ist. Der gezielte Zugriff und die Dekompression einzelner Werte aus dem komprimierten Datensatz ist nicht vorgesehen. Die bevorzugte Behandlung ausgewählter Bildteile (*region of interest*) wird in der Regel statisch in den Datenstrom kodiert (z. B. JPEG 2000 [JPEG 2000]). Diese Vorgehensweise erlaubt zwar die Entwicklung effizienter Kompressionsverfahren, bei denen die Koeffizienten ausschließlich entsprechend ihrem Beitrag zum Gesamtergebnis behandelt werden (z. B. SPIHT, Abschnitt 3.4.1), für die Kompression von Lichtfeldern sind solche Ansätze jedoch nicht geeignet.

   Aus diesem Grund wurde die Entscheidung getroffen, mit dem Waveletstream eine komplett neue Datenstruktur zu entwickeln. Die Anordnung der Koeffizienten im Waveletstream ermöglicht einerseits deren **komprimierte Speicherung**, anderseits läßt spezielle Navigationsinformation den **wahlfreien Zugriff** auf einzelne Werte im Lichtfeld zu. Die Neuentwicklung der Waveletstream-Datenstruktur ermöglichte weiterhin die Beachtung von Aspekten, die für Lichtfelder spezifisch sind. So werden für eine einfache Integration von Lichtfeldern in andere Szenen Informationen über die **Objektsilhouette** gespeichert. Mit dieser können Lichtfeldwerte, die zum Objekt gehören, von denen, die den Hintergrund zeigen, unterschieden werden. Durch den flexiblen Entwurf des Waveletstream können weiterhin beliebige vektorielle Einträge im Lichtfeld gespeichert werden. So kann beispielsweise Tiefen- und damit **Geometrieinformation** im Lichtfeld abgelegt werden, welche die korrekte Berechnung der gegenseitigen Verdeckung von Szenenobjekten erlaubt.

3. Mit der Haar-MSA wurden Tree-Wavelets (Abschnitt 3.2.3) zur Dekomposition der Lichtfeld-Daten ausgewählt. Dies geschah weil, wie oben beschrieben, zur Berechnung einer Lichtfeld-Ansicht nur eine zweidimensionale Teilmenge aller Lichtfeldwerte benötigt wird. Tree-Wavelets zeichnen sich dadurch aus, dass sich die Träger aller Wavelets einer Skala nicht gegenseitig überlappen. Die Veränderungen, die sich durch Benutzung einer Wavelet-Basis mit überlappenden Trägern ergeben, lassen sich an einem Beispiel veranschaulichen, das auf zwei Dimensionen reduziert ist.

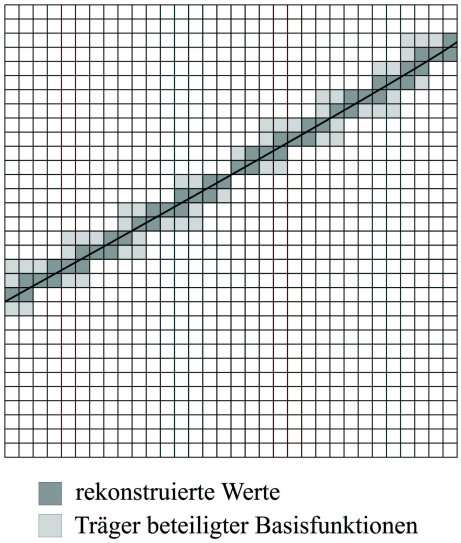


Abbildung 4.3: Zugriff auf alle Pixel auf einer beliebigen Geraden in einem zweidimensionales Bild. Die rekonstruierten Werte sind dunkel, die Träger der Tree-Wavelets, deren Koeffizienten ausgewertet werden müssen, sind in einem helleren Grauton markiert.

Im zweidimensionalen Fall entspricht die Berechnung einer Lichtfeld-Ansicht dem Zugriff auf alle Pixel eines Bildes entlang einer beliebigen Geraden (Abb. 4.3). Zur Rekonstruktion aller Werte auf der Geraden müssen alle Koeffizienten der Basisfunktionen, deren Träger die Gerade schneiden, ausgewertet werden. Bei Verwendung von Tree-Wavelets sind dies im zweidimensionalen Fall jeweils vier Koeffizienten. Bei Benutzung von Basisfunktionen mit sich gegenseitig überlappenden Trägern, erhöht sich die Anzahl der Basisfunktionen und damit der Koeffizienten, die Einfluss auf die Werte der Geraden haben.

Werden im zweidimensionalen Fall Wavelets mit einer Trägergröße von $n \times n$ Pixeln benutzt, erhöht sich die Anzahl der Koeffizienten, die zur Bestimmung einer Basisfunktion ausgewertet werden müssen, um den Faktor $\frac{n^2}{4}$ gegenüber Tree-Wavelets. Manche Koeffizienten können für mehr als eine Basisfunktion verwendet werden. Obgleich sich dadurch die Zeit für den Zugriff auf die benötigten Koeffizienten verringert, lässt sich der Gesamtaufwand der Rekonstruktion dadurch nicht senken. Allgemein gilt, dass die Anzahl der auszuwertenden Koeffizienten mit $O((\frac{n}{2})^d)$ steigt, wobei d die Dimensionalität des Datensatzes und n die Größe des Trägers in Pixeln ist.

Damit eine **interaktive Rekonstruktion** von Lichtfeld-Ansichten auf Standard-PC-Hardware möglich wird, muss somit ein Tree-Wavelet ausgewählt werden. Die ausgewählte **Haar-Waveletbasis** hat weiterhin den Vorteil, dass zur Berechnung der Wavelet-Koeffizienten nur einfachste Rechenoperationen auf ganzen Zahlen benötigt werden (Abschnitt 3.2.3). Für die Konstruktion der benötigten höherdimensionalen Basisfunktionen wurde der Non-Standard-Ansatz (Abschnitt 3.3.1) gewählt. Die Verwendung anderer Wavelet-Basen zur Lichtfeld-

Kompression wird in Abschnitt 4.3.13 diskutiert.

4.3.2 Überblick

Die neue Waveletstream-Datenstruktur wurde entsprechend der in Abschnitt 4.2 formulierten Anforderungen entwickelt. Die Konstruktion des Waveletstream und die Lichtfeld-Darstellung erfolgt in einzelnen Schritten, über die jetzt ein kurzer Überblick gegeben werden soll.

Nach der Wavelet-Dekomposition (Abschnitte 4.3.4 und 4.3.5) werden die berechneten Koeffizienten zur Konstruktion des Waveletstream in einen Bytestream geschrieben (Abschnitt 4.3.6). Hierbei werden die Wavelet-Koeffizienten für die niedrigen Skalen zuerst gespeichert (Abschnitt 4.3.3). Dieses Schema ermöglicht die Übertragung der Koeffizienten in der Reihenfolge ihrer Wichtigkeit, sowie eine progressive Verfeinerung der Daten. Informationen über die existierenden Koeffizienten (Abschnitt 4.3.7) und Navigationsinformationen (Abschnitt 4.3.8) vervollständigen den Waveletstream und ermöglichen die schnelle Rekonstruktion (Abschnitte 4.3.9) einzelner Lichtfeldwerte zur Berechnung von Lichtfeld-Ansichten (Abschnitt 4.3.10). Eine zusätzliche Kodierung der Objektsilhouette verbessert die Kompression und die Integration von Lichtfeldern in virtuelle Umgebungen (Abschnitt 4.3.12).

Alle im Folgenden beschriebenen Verfahren und Datenstrukturen arbeiten mit vierdimensionalen vektoriellen Werten. Zur Vereinfachung der Darstellung sind allerdings alle Abbildungen für den zweidimensionalen Fall ausgeführt.

4.3.3 Der Koeffizienten-Baum

In Abschnitt 4.3.1 wurde begründet, warum für die Kompression von Lichtfeldern Tree-Wavelets benutzt werden. Da sich bei Tree-Wavelets die Träger der Waveletfunktionen einer Skala nicht gegenseitig überschneiden, ist es nahe liegend, die Detailkoeffzienten, die während der Wavelet Dekomposition der Lichtfeld-Daten gewonnen werden, in einem *Koeffizientenbaum* zu organisieren (Abb. 4.4). Durch die Träger der Waveletfunktionen ist für jede Skala j eine Zerlegung des gesamten Lichtfeldes in disjunkte Teilbereiche vorgegeben. Im Koeffizientenbaum können dadurch die Koeffizienten so organisiert werden, dass jeder Knoten die Detailkoeffizienten für genau einen dieser Teilbereiche des Lichtfeldes speichert. Dabei enthalten die Nachfolger jedes Knotens die Koeffizienten, welche die Lichtfeldwerte des Vorgängers verfeinern. Eine Konsequenz dieser Anordnung in Verbindung mit der Benutzung von Tree-Wavelets ist, dass zur Rekonstruktion eines bestimmten Lichtfeldwertes der Koeffizientenbaum nur entlang eines einzelnen Pfades traversiert werden muss. Der Pfad ist dabei nur durch die Koordinaten des abgefragten Werts festgelegt.

Ein Lichtfeld in der 2P-Parametrisierung besitzt eine Auflösung von $res_s \times res_t \times res_u \times res_v$ Werten. Dabei ist häufig die Auflösung $res_u \times res_v$ der Kameraebene deutlich geringer als die Auflösung $res_s \times res_t$ der Bildebene. Dies bedeutet, dass der Rang der Knoten und die Anzahl der Koeffizienten, die in jedem Knoten gespeichert sind, zu den Blättern hin abnehmen kann. Denn jedes Niveau im Koeffizientenbaum repräsentiert das Lichtfeld auf einer bestimmten Skala. Dabei verdoppelt sich

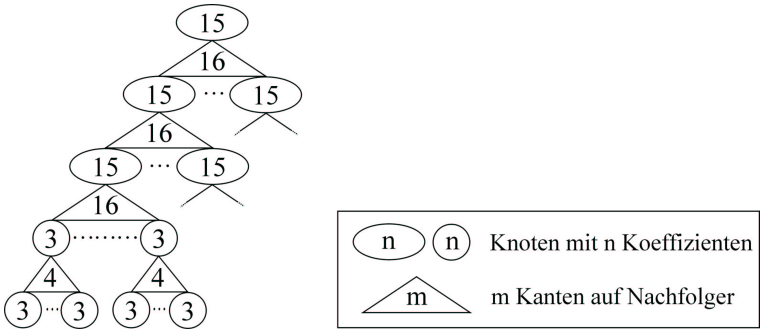


Abbildung 4.4: Schematische (Teil-)Darstellung eines Koeffizientenbaums für ein Lichtfeld der Größe $32 \times 32 \times 8 \times 8$. Der Rang der Knoten und die Anzahl der gespeicherten Koeffizienten pro Knoten nimmt zu den Blättern hin ab.

die Auflösung der vier Dimensionen s, t, u und v beim Abstieg im Baum auf jedem Niveau. Diese Verdopplung geschieht in jeder Dimension so lange, bis die Auflösung des Lichtfeld-Datensatzes erreicht ist. Auf den nachfolgenden Niveaus erhöht sich nur noch die Auflösung in Bezug auf die verbleibenden Dimensionen.

Wie oben beschrieben repräsentiert jeder Knoten im Koeffizientenbaum einen Teilbereich des Lichtfeldes. Als *Dimensionalität dim eines Knotens* soll ab jetzt die Anzahl der Dimensionen bezeichnet werden, in denen sich dieser Teilbereich unterteilen läßt. Ein Knoten, der beispielsweise einen Bereich von $64 \times 64 \times 8 \times 8$ Lichtfeldwerten beschreibt, hat somit eine Dimensionalität von 4, ein Knoten, zu dem ein Datenbereich von $16 \times 16 \times 1 \times 1$ Lichtfeldwerten gehört, eine Dimensionalität von 2. Da der Parameterbereich eines Lichtfeldes auf jedem Niveau des Koeffizientenbaums identisch unterteilt wird, ist die Dimensionalität aller Knoten eines Niveaus j gleich und kann in Abhängigkeit des Niveaus $dim(j)$ geschrieben werden.

Es ist leicht zu erkennen, dass ein Knoten mit der Dimensionalität $dim(j)$ eine Anzahl von $2^{dim(j)} - 1$ Detailkoeffizienten speichert und den Rang $2^{dim(j)}$ hat, sofern er ein innerer Knoten ist. Die maximale Tiefe d_{max} des Koeffizientenbaums berechnet sich mit

$$d_{max} = \lceil \max\{ld(res_s), ld(res_t), ld(res_u), ld(res_v)\} \rceil \tag{4.1}$$

4.3.4 Dekomposition der Daten

Bevor die Wavelet-Koeffizienten in den Koeffizientenbaum eingetragen werden können, muss zunächst die Wavelet-Transformation des Lichtfeldes berechnet werden. Zur Dekomposition der Lichtfeld-Daten wird ein Algorithmus ähnlich dem in Abschnitt 3.3.1 benutzt. Da die einzelnen Dimensionen eines Lichtfeldes unterschiedliche Auflösungen besitzen können, muss in der praktischen Ausführung vor allem die unterschiedliche Dimensionalität der Knoten auf den verschiedenen Niveaus im Koeffizientenbaum beachtet werden. Deshalb beruht der unten angegebenen Dekompositions-Algorithmus auf der Idee, den Dekompositionsschritt zunächst auf bestimmte Lichtfeld-Dimensionen zu beschränken, um dann nach und nach alle

weiteren hinzu zu nehmen.

Zu Beginn der Prozedur `decomposeLightField` werden alle Lichtfeld-Daten in ein vierdimensionales Feld $data$ geschrieben. Entsprechend dem gewählten Non-Standard-Ansatz wird auf den in $data$ gespeicherten Lichtfeld-Daten nacheinander und in Bezug auf unterschiedliche Dimensionen ein Schritt der Wavelet-Dekomposition ausgeführt (Schritt 5). Die Menge $\mathcal{D} \subset \{s,t,u,v\}$ enthält die Dimensionen, in Bezug auf welche das Lichtfeld gefiltert wird. $\mathcal{D}$ wird in Schritt 2 mit denjenigen Dimensionen des Lichtfeldes initialisiert, die die höchste Auflösung besitzen. Nach jedem Analyseschritt wird $\mathcal{D}$ gegebenenfalls erweitert (Schritt 6). Der eigentliche Analyseschritt wird in der Prozedur `decompositionStep` durchgeführt.

```
PROCEDURE decomposeLightField(L, data)
BEGIN
 1. Trage alle Werte aus L in das vierdimensionales Feld data ein.
 2. D := {dim ∈ {s,t,u,v} | ⌈ld(res_dim)⌉ = d_max}
 3. FOR j = d_max − 1 DOWNTO 0 DO
     4. FOR dim ∈ D DO
         5. decompositionStep(data, j, dim, D)
        OD
     6. D = D ∪ {dim ∈ {s,t,u,v} | ⌈ld(res_dim)⌉ = j}
    OD
END
```

Die Prozedur `decompositionStep` führt für den Bereich $[0,\ldots,2^{j+1}-1]^4$ von $data$ in Bezug auf die übergebene Dimension dim einen Schritt in der Wavelet-Dekomposition durch und speichert das Ergebnis wieder in $data$. In Abschnitt 3.1.3 wird hierzu erklärt, wie dabei eine gefilterte Repräsentation eines Signals mit halbierter Auflösung und eine Anzahl von Detailkoeffizienten berechnet wird. Auf die Angabe eines Pseudo-Codes wird verzichtet, da die Prozedur neben einfachen Berechnungen vor allem eine Umordnung der Daten vornimmt, welche in Abbildung 4.5 schematisch dargestellt ist. Auf dieser ist angedeutet, wie die berechneten Skalierungskoeffizienten im unteren, die Detailkoeffizienten im oberen Parameterbereich von $data$ abgelegt werden (Abb. 4.5 (a)). Wenn `decompositionStep` für alle Dimensionen in $\mathcal{D}$ durchgeführt wurde, befindet sich im Parameterbereich $[0,\ldots,2^j-1]^4$ von $data$ schließlich eine Darstellung des Lichtfeldes in der Auflösung 2^j (Abb. 4.5 (b)). Die übrigen bearbeiteten Einträge von $data$ speichern Detailkoeffizienten. Durch den abnehmenden Wert von j (Schritt 3) wird im nächsten Schritt der Wavelet-Dekomposition $(j-1)$ nur der Bereich von $data$ bearbeitet, welcher die Skalierungskoeffizienten enthält.

Nach Beendigung der Prozedur `decomposeLightField` enthält das vierdimensionale Feld $data$ Einträge, die entsprechend Abbildung 4.5 (c) strukturiert sind. Dabei speichert $data$`[0,0,0,0]` mit dem Skalierungskoeffizienten c_0^0 den Mittelwert aller Lichtfeldwerte.

Während der Wavelet-Transformation des Lichtfeldes können, auch bei Verwendung

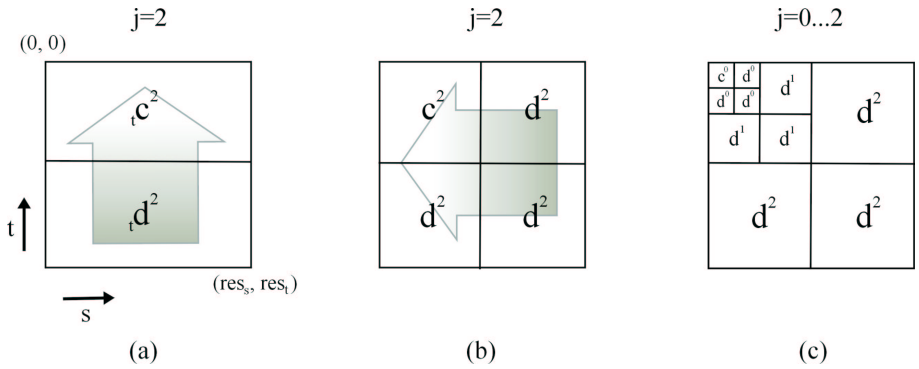


Abbildung 4.5: Schematische Darstellung der Wavelet-Dekomposition eines Lichtfeldes. Entsprechend der gewählten Non-Standard-Konstruktion wird die Wavelet-Dekomposition nacheinander in Bezug auf unterschiedliche Dimensionen des Lichtfeldes durchgeführt. Nach der Dekomposition in t-Richtung in Schritt (a) werden die Daten in s-Richtung transformiert (b). Mit dem Subskript t soll angedeutet werden, dass die Werte ${}_tc^j$ und ${}_td^j$ die Ergebnisse der Wavelet-Dekomposition nur für die t-Richtung sind und deshalb Zwischenergebnisse darstellen. Abbildung (c) zeigt das Endergebnis der Wavelet-Transformation.

der Haar-Wavelet-Basis, nicht ganzzahlige Werte auftreten. Aus diesem Grund enthält das Feld $data$ Einträge mit Fließkomma-Werten. Die Berechnungen während `DecompositionStep` werden ebenfalls mit Fließkomma-Arithmetik durchgeführt, um die Genauigkeit der Ergebnisse zu erhöhen. Die notwendige Quantisierung wird im folgenden Abschnitt beschrieben.

4.3.5 Quantisierung der Koeffizienten

Gewöhnlich speichert ein Lichtfeld Dreier-Tuppel, welche Farbwerte im RGB- oder YUV-Format enthalten. Doch das Konzept der Speicherung und Darstellung von Objekten mit Lichtfeldern läßt allgemeinere Datenformate zu. Die Daten, die in einem Lichtfeld gespeichert sind, können aus mehreren unabhängigen *Kanälen* bestehen. So kann ein Lichtfeld sowohl monochrome Daten, Farbwerte oder auch Zusatzinformationen speichern. Diese können beispielsweise aus einem α-Kanal, der Informationen zur Transparenz enthält, oder aus Tiefeninformation bestehen. Deshalb ist die Waveletstream-Datenstruktur allgemein in der Lage, Lichtfelder, die aus einer beliebigen Anzahl von Datenkanälen bestehen, zu speichern und darzustellen.

In den meisten praktischen Fällen bestehen die Einträge eines Lichtfeldes allerdings aus Farbwerten im YUV- oder RGB-Format mit einer Größe von 3×8 Bit. Aus diesem Grund werden die Wavelet-Koeffizienten im coefficient stream mit 8 Bit Genauigkeit quantisiert. Die hierdurch mögliche Ausrichtung der Daten an der Bytegrenze beschleunigt den Zugriff auf die einzelnen Werte. Zusätzliche Informationen, wie ein α-Kanal oder Tiefeninformation, die im Waveletstream gespeichert werden können, werden ebenfalls auf 8 Bit Genauigkeit oder ein Vielfaches davon quantisiert.

Ein einfacher und leistungsfähiger Ansatz zur Verringerung des bei der Quantisierung auftretenden Fehlers ist die Speicherung zusätzlicher Korrekturfaktoren für jedes Niveau im Koeffizientenbaum. Für alle Detailkoeffizienten eines Niveaus j werden deren absolutes Minimum min_j und Maximum max_j bestimmt. Alle Minima und Maxima werden in eine Tabelle geschrieben, die dem Waveletstream hinzugefügt wird. Danach werden alle Koeffizienten d_i^j so skaliert, dass sie den maximalen, mit 8 Bit darstellbaren Wertebereich $[-128, \ldots, +127]$ überdecken.

$$\bar{d_i^j} = \frac{255(d_i^j - min_j)}{max_j - min_j} \tag{4.2}$$

$\bar{d_i^j}$ ist der korrigierte Koeffizientenwert.

Mit den gespeicherten Minima und Maxima können während des Dekompressionsschritts die ursprünglichen Werte wiederhergestellt werden. Die in Abschnitt 4.5.5 enthaltenen Ergebnisse zeigen, wie mit diesem Verfahren der Kompressionsfehler verringert werden kann.

4.3.6 Der coefficient stream

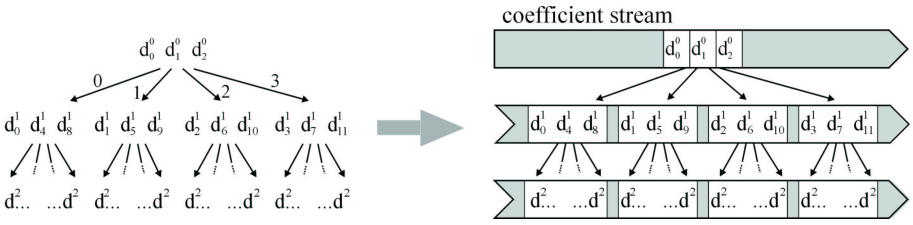


Abbildung 4.6: Anordnung der Koeffizienten im coefficient stream: Durch die Traversierung des Koeffizientenbaums von links nach rechts und von oben nach unten werden die Detailkoeffizienten in den coefficient stream geschrieben.

Zur Speicherung der Lichtfeld-Daten werden die Wavelet-Koeffizienten in den *coefficient stream* geschrieben. Hierzu müssen die Werte aus $data$ in der richtigen Reihenfolge ausgelesen und im Bytestream gespeichert werden. Der Skalierungskoeffizient c_0^0 wird als erster Wert in den coefficient stream eingefügt. Die Koordinaten der Einträge in $data$ werden so berechnet, dass ihre Reihenfolge der einer Breitensuche, also einem Durchlauf von rechts nach links und von oben nach unten, im Koeffizientenbaum, entspricht (Abb. 4.6).

Die durch die Träger der Waveletfunktionen auf jeder Skala induzierte Zerlegung des Lichtfeld-Datensatzes ist in Abbildung 4.7 angedeutet. Um die Koordinaten von Lichtfeldwerten zu berechnen, die bestimmten Werten im Koeffizientenbaum entsprechen, werden die Teilbereiche des Lichtfeldes und die Knoten im Koeffizientenbaum auf jeder Skala durchnummeriert. Der Zusammenhang zwischen den *Ordnungsnummern* im Baum und denen im Lichtfeld ist offensichtlich (Abb. 4.7). Zur Übertragung der Koeffizienten in den coefficient stream wird der Koeffizientenbaum „virtuell“ nur mit Hilfe der Nummerierung durchlaufen. Auf diese Weise werden die Koordinaten in der richtigen Reihenfolge berechnet.

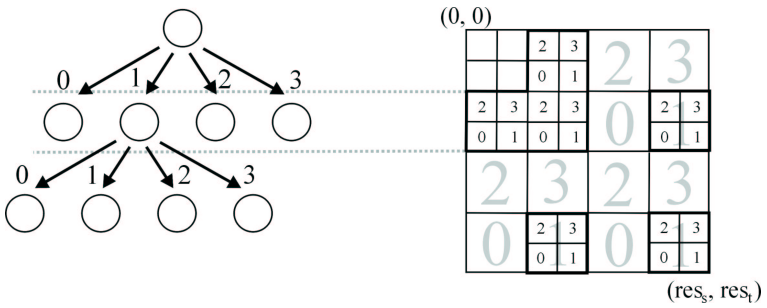


Abbildung 4.7: Zusammenhang zwischen Position der Koeffizienten im Koeffizientenbaum und im Feld, das die Wavelet-Dekomposition der Lichtfeldwerte speichert. Durch die Nummerierung der Knoten im Koeffizientenbaum können die Einträge im Feld und im Baum in Beziehung gesetzt werden.

4.3.7 Die node description-Datenstruktur

Wie in Abschnitt 3.2 erklärt, kann eine Wavelet-Transformation durch Weglassen von Koeffizienten mit (geringem oder) einem Wert gleich Null (verlustbehaftet) komprimiert werden. Bei Verwendung einer orthonormalen MSA, wie der Haar Wavelet-Basis, kann der auftretende Kompressionsfehler leicht mit (3.1) berechnet werden. Der Grenzwert, mit welchem diejenigen Detailkoeffizienten bestimmt werden, die während der Kompression gelöscht werden, kann vom Benutzer für jeden Kanal der Daten frei festgelegt werden. Hierdurch kann den unter Umständen unterschiedlichen Kompressionseigenschaften der einzelnen Kanäle Rechnung getragen werden. Die unterschiedliche Kompression der Kanäle kann dazu führen, dass deren Anzahl zu den Blättern des Koeffizientenbaums hin uneinheitlich abnimmt.

Ein Kanal C wird als *aktiv in einem Knoten* N des Koeffizientenbaums bezeichnet, wenn N oder einer seiner Nachfolger Informationen bezüglich C speichert. Während der Rekonstruktion von einzelnen Werten aus dem Waveletstream muss die Aktivierung bzw. Deaktivierung der Kanäle eines Lichtfeldes überwacht werden.

Durch die Löschung von Koeffizienten im Zuge der Wavelet-Kompression wird der Koeffizientenbaum unvollständig. Einzelne Koeffizienten oder auch ganze Teilbäume können nach der Wavelet-Kompression fehlen, falls alle Koeffizienten des Teilbaums entfernt wurden. Zur Kodierung der Position der verbleibenden Koeffizienten und der existierenden Nachfolger jedes Knotens muss zusätzliche Metainformation in den coefficient stream eingefügt werden.

Für jeden im coefficient stream gespeicherten Knoten N wird eine *node description*-Datenstruktur angelegt (Abb. 4.8). Eine node description besteht aus einer Menge von Bitfeldern (*significance maps* [SHAPIRO 1993]). Diese sind eindimensionale Felder mit binären Einträgen. Für jeden aktiven Kanal in N ist in der zugehörigen node description genau ein *coeff map*-Bitfeld enthalten. Das coeff map-Bitfeld speichert für jeden Koeffizienten im Knoten, ob dieser existiert und im coefficient stream gespeichert ist. Jede coeff map hat $2^{dim(j)} - 1$ Einträge.

Ein Knoten N kann Vorgänger von bis zu $2^{dim(j)}$ Nachfolgern im Koeffizientenbaum

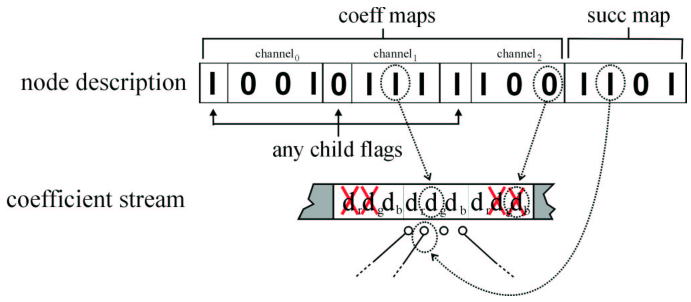


Abbildung 4.8: Node description-Datenstruktur mit dem zugehörigen Teil des coefficient stream. Für drei Einträge in der node description ist deren Bedeutung im coefficient stream markiert.

sein, der selber wiederum die Wurzeln von bis zu $2^{dim(j)}$ Teilbäumen sind. Falls N mindestens einen Nachfolger besitzt, wird die node description von N um ein *succ map*-Bitfeld erweitert, welches die Position der existierenden Nachfolger kodiert (Abb. 4.8). Um zu vermeiden, dass in einer node description ein succ map-Bitfeld gespeichert wird, obwohl der betreffende Knoten keine Nachfolger hat, wie dies bei Blättern im Koeffizientenbaum der Fall ist, wird das *any child flag* verwendet.

Für jeden aktiven Kanal C in einem Knoten N wird ein any child flag gespeichert. Dieses zeigt an, ob mindestens ein Nachfolger von N Koeffizienten des Kanals C speichert. Wenn mindestens eines der any child flags einer node description gesetzt ist, enthält die node description ein succ map-Bitfeld. Ist das any child flag eines Kanals C nicht gesetzt, so sind keinerlei Koeffizienten von C in einem der Nachfolger von N gespeichert. Der Kanal C wird dann als *deaktiviert* bezeichnet. Ist keines der any child flags einer node description gesetzt, enthält diese kein succ map-Bitfeld.

In Abbildung 4.8 ist eine node description und das Teilstück im coefficient stream, welches von der node description beschrieben wird, beispielhaft dargestellt. Da für zwei der drei Kanäle das any child flag gesetzt ist, enthält die node description ein succ map-Bitfeld, das die Position der Nachfolger des Knotens N beschreibt. Alle Nachfolger von N speichern Informationen zu den Kanälen 0 und 2, aber nicht zu Kanal 1.

Platzierung der node description-Datenstruktur im Waveletstream

Es wäre nahe liegend, die node description eines jeden Knotens N direkt vor den Koeffizienten, welche durch diese beschrieben werden, in den Waveletstream einzufügen. Bei einem solchen Vorgehen würde jedoch eine große Menge Speicherplatz verschwendet. Wie in Abschnitt 4.3.3 beschrieben, sinkt die Dimensionalität der Knoten im Koeffizientenbaum gewöhnlich mit steigendem Niveau. Parallel hierzu verringert sich auch die Größe der coeff map-Bitfelder, die in einer node description gespeichert sind, auf 3 Bits bzw. 1 Bit Länge für einen zwei- bzw. eindimensionalen Knoten. Falls in einer node description keines der any child flags gesetzt ist, beträgt deren Größe $2^{dim(j)} \times c$ Bits, wobei c die Anzahl der aktiven Kanäle im Knoten bezeichnet.

Um den Zugriff auf die Daten im Waveletstream zu erleichtern, sind alle Koeffizienten an der Byte-Grenze ausgerichtet. Somit muss auch für jede node description eine ganzzahlige Anzahl von Bytes zur Speicherung verwendet werden. Auf diese Weise würden aber bei zwei- bzw. eindimensionalen Knoten vier bzw. sechs Bits nicht genutzt ($c=1$ oder $c=3$). Im schlechtesten Fall blieben so bei einem Lichtfeld mit einer Auflösung von $256 \times 256 \times 8 \times 8$ Werten 512 kByte oder acht Prozent des Gesamtspeichers im Waveletstream ungenutzt.

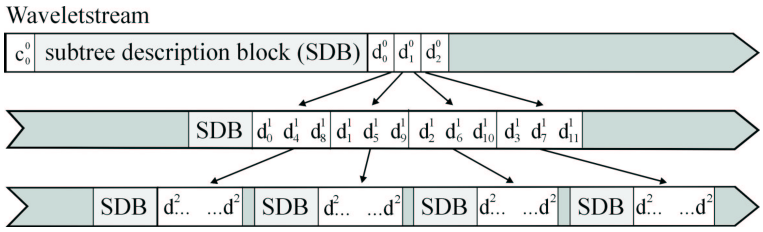


Abbildung 4.9: Position der subtree description blocks (SDB) innerhalb des Waveletstream: Vor jeder Gruppe von Knoten mit gemeinsamem direktem Vorgänger befindet sich ein SDB.

Dieses Problem kann zum Teil dadurch vermieden werden, dass jeweils zuerst alle node descriptions und danach alle Koeffizienten von Knoten, welche einen gemeinsamen direkten Vorgänger im Koeffizientenbaum besitzen, im Waveletstream gespeichert werden. Dazu werden alle node descriptions von Knoten mit einem gemeinsamen direkten Vorgänger in *subtree description blocks (SDB)* (Abb. 4.9) zusammengefasst.

Wie oben beschrieben, wird für jeden in einem Knoten N aktiven Kanal ein coeff map-Bitfeld mit einer Länge von $2^{dim(j)}-1$ Bits und das einzelne any child-Bit gespeichert (Abb. 4.8). Das succ map-Bitfeld hat ebenfalls eine Länge von $2^{dim(j)}$ Bits. Dies ist für eine kompakte Anordnung der Daten im Speicher günstig. Durch Bündelung aller Bitfelder im subtree description block bleibt höchstens am Ende des SDB eine Lücke von ungenutzten Bits im Waveletstream. Weiterhin kann durch dieses Verfahren die Anzahl der Bytes, die später für Navigationszwecke in den Waveletstream eingefügt werden müssen, spürbar reduziert werden (siehe Abschnitt 4.3.8).

Nachdem der Koeffizientenbaum durch Weglassen von Koeffizienten komprimiert und entsprechende subtree description blocks in den coefficient stream eingefügt wurden, enthält der so entstandene Waveletstream alle notwendigen Informationen zur Speicherung, Übermittlung und Rekonstruktion eines durch Kompression unvollständig gewordenen Koeffizientenbaums.

Um die im Waveletstream gespeicherten Lichtfeld-Daten zur Darstellung von Objekten nutzen zu können, muss auf einzelne Werte zugegriffen werden. Hierzu werden die im folgenden Abschnitt beschriebenen Navigationsinformationen in den Waveletstream eingefügt.

4.3.8 Navigation im Waveletstream

Wie in Abschnitt 4.3.3 beschrieben, ist eine Konsequenz der Benutzung der Haar-MSA zur Berechnung der Wavelet-Dekomposition, dass zur Rekonstruktion eines einzelnen Lichtfeldwertes der Koeffizientenbaum entlang nur eines bestimmten Pfades traversiert werden muss.

Somit ist die Rekonstruktion eines einzelnen Wertes eines Lichtfeldes aus dem Waveletstream ein rekursiver Prozess, der an der Wurzel des Koeffizientenbaums beginnt. An jedem besuchten Knoten werden die gespeicherten Koeffizienten ausgewertet und das Ergebnis weiter verfeinert. Der Pfad, auf welchem der im Waveletstream gespeicherte Baum traversiert wird, ist durch die Koordinate (s, t, u, v) des abgefragten Wertes bestimmt.

Pfadberechnung

Eine Folge von Knotennummern definiert den Pfad, entlang dem der Koeffizientenbaum traversiert wird. Dabei sind die Knotennummern jeweils die Ordnungsnummern des direkten Nachfolgerknotens (Abschnitt 4.3.6) im Pfad.

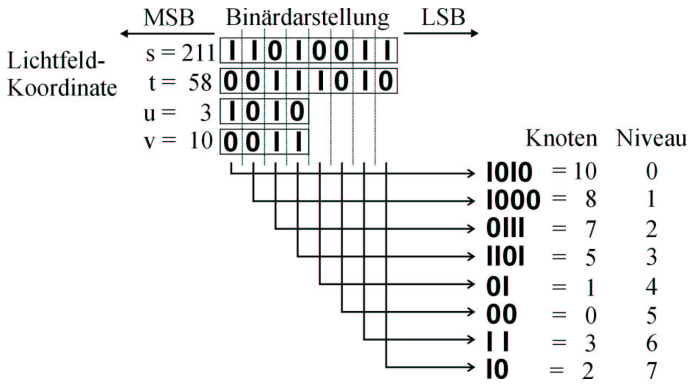


Abbildung 4.10: Berechnung der Knotennummern des zu traversierenden Pfades aus der abgefragten Lichtfeld-Koordinate: Beginnend beim höchstwertigsten Bit werden die Ordnungsnummern aus jeweils einem Bit der Binärdarstellung der Koordinatenwerte gebildet.

Die Ordnungsnummern der zu besuchenden Knoten werden aus der Binärrepräsentation der abgefragten Koordinate (s, t, u, v) gebildet. In Abbildung 4.10 sind hierzu beispielhaft die binären Darstellungen für Koordinatenwerte s, t, u und v angegeben. Beginnend beim höchstwertigsten Bit jeder Koordinate werden die Ordnungsnummern aus jeweils einem Bit von s, t, u und v zusammengesetzt. Die Tatsache, dass die Dimensionalität der Knoten im Koeffizientenbaum mit wachsender Tiefe abnehmen kann, spiegelt sich darin wieder, dass sich die Höhe der möglichen Ordnungsnummern automatisch verringert, sobald alle Stellen einer Koordinate „verbraucht“ sind.

Navigationsinformation

Auf Grund der Organisation der Daten im Waveletstream entspricht der gezielte Zugriff auf einen direkten Nachfolger im Koeffizientenbaum dem Vorrücken um eine Anzahl von Bytes im Waveletstream. Da während der Kompression der Lichtfeld-Daten einzelne Koeffizienten oder ganze Teilbäume aus dem Koeffizientenbaum entfernt wurden, ist es nicht möglich die Position eines Nachfolgers im Waveletstream mit einer einfachen Regel zu berechnen. Deshalb muss während des Einlesens in den Speicher zusätzliche Navigationsinformation in den Waveletstream eingefügt werden.

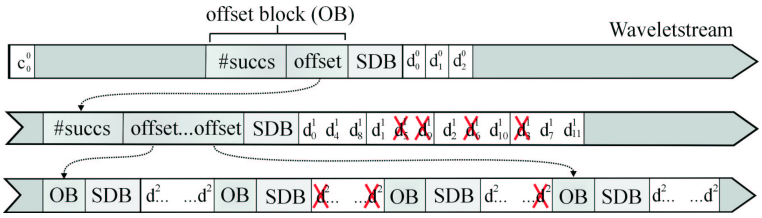


Abbildung 4.11: Waveletstream mit Offsets. Die Offsets werden vor den subtree description blocks (SDB) eingefügt. Der Offset eines Knotens zeigt auf das erste Kind eines Knotens. Während der Kompression der Daten gelöschte Koeffizienten sind mit einem Kreuz markiert.

Die benötigte Navigationsinformation besteht aus zwei Teilen. Im Folgenden soll der aktuelle Knoten im Pfad mit N, sein direkter Vorgänger mit N_{pre} und sein direkter Nachfolger mit N_{succ} bezeichnet werden.

1. Für jeden Knoten N, der mindestens einen Nachfolger hat, d.h. mindestens ein any child flag der zugehörigen node description ist gesetzt, wird ein Offset in den Waveletstream eingefügt. Dieser Offset bezeichnet die Position von N_{succ}. Die Offsets aller Knoten, die von einem subtree description block beschrieben werden, sind in einem *offset block (OB)* zusammengefasst. Der offset block wird direkt vor dem SDB in den Waveletstream eingefügt (Abb. 4.11).

   Um den Platzbedarf der Navigationsinformation zu reduzieren, wird die Größe der Offsets an den tatsächlichen Bedarf angepasst. Außerdem werden die Offsetwerte relativ zum Beginn des jeweiligen Niveaus im Koeffizientenbaum angegeben. Der Beginn jedes Niveaus wird hierzu mit seinem absoluten Wert in einer separaten Tabelle *levelStart* gespeichert. Die benötigte Anzahl Bytes für die Darstellung des größtmöglichen Offsetwertes auf jedem Niveau wird bestimmt und in einer weiteren Tabelle *offsetSize* abgelegt. In Abschnitt 4.5.5 wird der hierdurch erzielte Vorteil untersucht.

2. Um eine bestimmte node description innerhalb eines subtree description block und die Koeffizienten eines bestimmten Knotens dahinter zu finden, muss weitere Hilfsinformation in den Waveletstream eingefügt werden.

   Die Größe eines offset blocks kann nicht ohne das Wissen über die Anzahl der Knoten mit mindestens einem Nachfolger im zugehörigen SDB bestimmt werden. Denn nur Knoten mit einem Nachfolger besitzen einen Offset. Ebenso

kann die Gesamtgröße des subtree description block nicht ohne diese Angabe bestimmt werden. Aus diesem Grund wird die Anzahl *#succs* der Knoten mit mindestens einen Nachfolger in ein Byte geschrieben, welches direkt vor dem offset block eingefügt wird (Abb. 4.11).

Damit sämtliche einen Knoten betreffenden Informationen gelesen werden können, bezeichnet der Offset auf einen Knoten N genau die Position im Waveletstream, an der mit dem Eintrag von *#succs* die Informationen zu N im Waveletstream beginnen.

Auffinden eines Knotens

Das Auffinden der node description, des Offsets und der Koeffizienten eines Knotens N mit Ordnungsnummer n und dem Offset o auf Niveau j im Koeffizientenbaum kann entsprechend der Prozedur `findNode` erfolgen.

```
PROCEDURE findNode(n,o,j)
BEGIN
 1. pos = levelStart[j] + o
 2. Lese #succs.
 3. Rücke pos um #succs× offsetSize[j] Bytes vor.
 4. #coeffs = 0
 5. #succs_pre = 0
 6. FOR i = 1 TO n − 1 DO
     7. IF Knoten mit Nummer i existiert THEN
          8. Lese node description.
          9. Zähle die Koeffizienten in #coeffs
         10. Falls Knoten Nr. i mindestens einen Nachfolger hat, inkrementiere
             #succs_pre.
         FI
    OD
11. Lese node description von N.
12. IF N hat mindestens einen Nachfolger THEN
    13. Lese o_new an Position levelStart[j] + o + 1 +
        #succs_pre× offsetSize[j].
    FI
14. pos = levelStart[j] + o + 1 + #succs× offsetSize[j]
15. Rücke pos um ⌈(#channels × #nodes + #succs)2^dim(j)/8⌉ Bytes vor.
16. Rücke um #coeffs Bytes vor.
17. Lese Koeffizienten von N.
END
```

Nachdem der Zeiger *pos* gesetzt wurde (Schritt 1), kann die Gesamtzahl aller Knoten im SDB mit mindestens einem Nachfolger $\#succs$ gelesen werden. Der offset block wird zunächst übersprungen (Schritt 3). Die im SDB gespeicherten node descriptions werden gelesen und die Anzahl der Koeffizienten $\#coeffs$ in den Knoten vor N bestimmt (Schritt 6 bis 10). Dabei kann die Existenz von Knoten vor N mit Hilfe der Einträge des succ map-Bitfeld des Vorgängerknotens N_{pre} im Pfad überprüft werden (Schritt 7). Zur Bestimmung der Zahl der Koeffizienten, die den Koeffizienten von N im Waveletstream vorangehen, werden die gesetzten Bits in allen ausgewerteten coeff map-Bitfeldern gezählt. Die Anzahl der Knoten, die mindestens einen Nachfolger besitzen, werden in $\#succs_{pre}$ gespeichert.

Nach Ende der Schleife steht der Zeiger *pos* auf der richtigen Position, um die node description von N lesen zu können (Schritt 11). Falls N einen Nachfolger besitzt, wird der Offset auf den ersten Sohn in o_{new} gespeichert. Abschließend können mit der gewonnenen Information der Rest des SDB (Schritt 15) und die Koeffizienten der Knoten vor N (Schritt 16) übersprungen, sowie die Koeffizienten von N gelesen werden (Schritt 17). Hierbei kann die Gesamtzahl der Knoten $\#nodes$ aus der Anzahl der gesetzten Bits im succ map-Bitfeld des Vorgängerknotens berechnet werden. Die Anzahl der aktiven Kanäle $\#channels$ ergibt sich aus dem Gesamtzustand der Berechnungen.

Mit den Einträgen im succ map-Bitfeld von N kann festgestellt werden, ob der nächste zu traversierende Knoten im Pfad existiert. Falls ja, liegen mit o_{neu} und dem succ map-Bitfeld von N die notwendigen Informationen vor, um den Nachfolger zu finden und dessen Koeffizienten auszuwerten.

Einfügen der Navigationsinformation

Die oben beschriebenen Navigationsinformationen werden während des Einlesens des Waveletstream in den Speicher eingefügt. Die unten angegebene Prozedur `readWaveletstream` implementiert die notwendigen Operationen.

Die Grundidee des Verfahrens besteht darin, während des Einlesens der Daten vor jedem subtree description block (SDB) im Speicher eine Anzahl von Bytes frei zu lassen, in welche die Anzahl der Knoten mit Nachfolger und deren Offsets geschrieben werden, sobald diese feststehen. Der Waveletstream wird hierzu Niveau für Niveau bearbeitet (Schritt 3). Die beiden Listen $list_{curr}$ und $list_{next}$ speichern für jeden SDB die Einfügeposition für die Offsets, die Anzahl der Knoten im SDB und den SDB selbst (Schritt 15).

Für jeden Knoten N_{pre} mit Nachfolger auf Niveau $j-1$ existiert ein SDB auf Niveau j (Schritt 5). Nachdem die aktuelle Position in pos_{succs} gespeichert wurde (Schritt 6), wird der Offset auf den aktuellen SDB berechnet und in die Leerstelle auf Niveau $j-1$ eingetragen (Schritt 8). Danach werden der aktuelle SDB und die zugehörigen Koeffizienten gelesen. Dies ist möglich, weil im aktuellen Listeneintrag *info* das succ map Bitfeld von N_{pre} abgelegt ist. Für jeden Knoten mit Nachfolger im aktuellen SDB wird ein Eintrag in $list_{next}$ angelegt (Schritt 13). Nachdem vor dem aktuellen SDB eine passende Lücke für die Offsets auf die Nachfolger gelassen wurde (Schritt 14), werden der SDB und die zugehörigen Koeffizienten geschrieben (Schritt 15).

Ist ein Niveau vollständig abgearbeitet, werden die Einträge von $list_{next}$ auf $list_{curr}$ übertragen (Schritt `16` und `17`), und das nächste Niveau wird bearbeitet.

Da auf Niveau 0 keine Vorgänger existieren, muss $list_{curr}$ geeignet initialisiert (Schritt `2`) und die Einträge auf Niveau 0 gesondert behandelt werden (Schritt `7`). In der Darstellung des Algorithmus wurde auf eine explizite Behandlung der Veränderung der Anzahl der aktiven Kanäle $\#channels$ zu Gunsten der Übersichtlichkeit verzichtet. Die korrekte Behandlung dieser Fälle ist aber gewährleistet. d_{max} ist der mit (4.1) bestimmte Wert.

```
PROCEDURE readWaveletstream
BEGIN
 1. Lese Skalierungskoeffizient c_0^0 und Hilfsdaten, wie offsetSize[].
 2. list_curr.push(1, aktuelle Position)
 3. FOR j = 0 TO d_max − 1 DO
    4. WHILE list_curr ≠ ∅ DO
       5. info = list_curr.pop()
       6. pos_succs = aktuelle Position
       7. IF j > 0 THEN
          8. Schreibe Offset auf aktuelle Position bei info.pos.
          FI
       9. Lese SDB und Koeffizienten.
      10. #succs = Anzahl der Knoten mit mindestens einem Nachfolger im
          SDB.
      11. Schreibe #succs an Position pos_succs.
      12. FOR i = 1 TO #succs DO
          13. list_next.attach(Anzahl der Nachfolger von Knoten i,
              aktuelle Position, SDB)
          14. Gehe offsetSize[j] Bytes vor.
          OD
      15. Schreibe SDB und Koeffizienten des aktuellen Knotens.
       OD
   16. list_curr = stack_next
   17. list_next = ∅
    OD
END
```

4.3.9 Rekonstruktion eines Lichtfeldwertes

Zur Berechnung eines Lichtfeldwertes wird der im Waveletstream gespeicherte Koeffizientenbaum entlang eines Pfades durchwandert. An jedem besuchten Knoten werden dabei die gespeicherten Koeffizienten ausgewertet.

In jedem Knoten N des Koeffizientenbaums sind bis zu $2^{dim(j)} - 1$ Koeffizienten gespeichert. Mit Hilfe eines Skalierungskoeffizienten können aus diesen $2^{dim(j)}$ Werte

rekonstruiert werden, welche das Lichtfeld in dem von N beschriebenen Bereich darstellen (Abschnitt 4.3.3). Mit Hilfe einer Nummerierung kann der zu rekonstruierende Wert eindeutig bezeichnet werden. Die Ordnungsnummer des zu rekonstruierenden Wertes ist hierbei immer mit der Ordnungsnummer des Nachfolgerknotens N_{succ} im Pfad identisch. Die Nummerierung der Knoten im Koeffizientenbaum ist in Abbildung 4.10 dargestellt. Der rekonstruierte Lichtfeldwert wird in Knoten N_{succ} wiederum als Skalierungskoeffizient verwendet, um das Ergebnis zu verfeinern.

Seien

$$\mathbf{d}_N := [d_1, \ldots, d_{2^{dim(j)}-1}]^T$$

die in N gespeicherten Detailkoeffizienten und $c_{N_{pre}}$ der Skalierungskoeffizient, der aus den Koeffizienten des Vorgängerknotens N_{pre} rekonstruiert wurde. Dann können die Lichtfeldwerte

$$\mathbf{c}_N := [c_0, \ldots, c_{2^{dim(j)}-1}]^T$$

mit Hilfe eines Synthesefilters $\mathbf{M}$ aus $\mathbf{d}_N$ und $c_{N_{pre}}$ mit

$$\mathbf{c}_N = \mathbf{M} \left[\frac{c_{N_{pre}}}{\mathbf{d}_N} \right] \tag{4.3}$$

rekonstruiert werden (Abschnitt 3.1.4).

Da tatsächlich nur einer der Werte in $\mathbf{c}_N$ benötigt wird, muss lediglich eine Zeile von $\mathbf{M}$ ausgewertet werden. Weiterhin müssen die Berechnungen nur mit den existierenden $c_i \in \mathbf{c}_N$ ausgeführt werden. Die coeff map-Bitfelder in der node description von N geben an, welche Koeffizienten existieren. Eine weitere Möglichkeit zur Beschleunigung der Berechnungen besteht bei Verwendung der Haar-MSA darin, dass es ausreichend ist, die c_i zu addieren und subtrahieren. Da die Ausgangsdaten ganzzahlige Werte sind, genügen hierfür Operationen auf ganzen Zahlen. Abschließend werden zur Verbesserung des Ergebnisses die in Abschnitt 4.3.5 beschriebenen Korrekturfaktoren entsprechend (4.2) angewandt.

4.3.10 Berechnung einer Lichtfeld-Ansicht

Eine *Lichtfeld-Ansicht* ist die Projektion einer zweidimensionalen Teilmenge aller Lichtfeldwerte auf den Bildschirm. Die Projektion wird dabei von der Position des Betrachters $\mathbf{e}$ und der Position der Bildebene bestimmt. Beim hier vorgestellten Verfahren zur Berechnung einer Lichtfeld-Ansicht wird die Projektion in zwei Schritten durchgeführt. Diese Zweiteilung vereinfacht die Berechnung.

Zunächst wird die Lichtfeld-Ansicht in Bezug auf die aktuelle Betrachterposition und die **Bildebene des Lichtfeldes** berechnet. Diese Lichtfeld-Ansicht wird dann auf die eigentliche Bildebene, den Bildschirm, abgebildet. Letzterer Schritt kann mit Hardwareunterstützung durchgeführt werden. Hierzu werden die rekonstruierten Werte aus dem Lichtfeld ähnlich wie in [SLOAN und HANSEN 1999] in eine Textur eingetragen, welche auf ein Rechteck aufgebracht wird. Das texturierte Rechteck befindet sich an der Position der Bildebene des Lichtfeldes. Die Darstellung erfolgt mit der OpenGL-Grafikbibliothek [KEMPF et al. 1997].

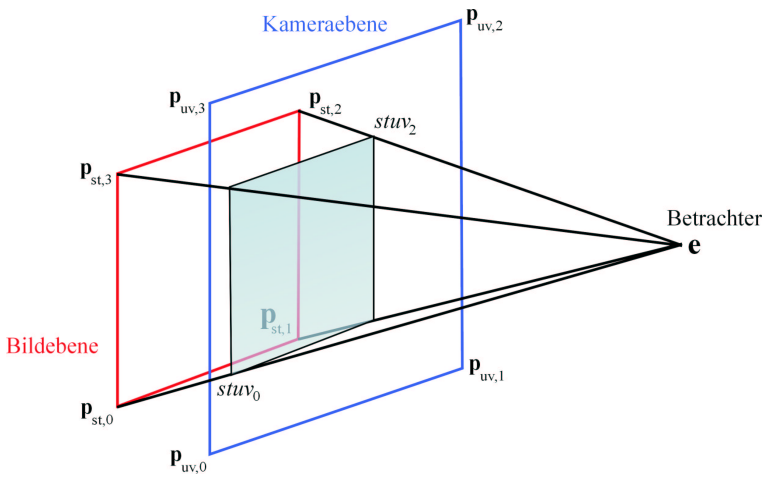


Abbildung 4.12: Abzufragendes Intervall von Lichtfeldwerten zur Berechnung einer Lichtfeld-Ansicht für Betrachterposition $\mathbf{e}$. Die Eckpunkte der Bildebene werden zur Bestimmung der Intervallgrenzen auf die Kameraebene projiziert und umgekehrt.

Wie in Abschnitt 4.2 erklärt, bilden die Koordinaten einer Lichtfeld-Ansicht einen zweidimensionalen affinen Unterraum im Parameterbereich des Lichtfeldes [GU et al. 1997]. Dieser Unterraum wird durch Intervallgrenzen $stuv_0, stuv_1, stuv_2$ und $stuv_3$ definiert. Da bei der 2P-Parametrisierung Bild- und Kameraebene des Lichtfeldes parallel sind, ist es zur Berechnung der Koordinaten ausreichend, den durch die Punkte $stuv_0, \ldots, stuv_3$ definierten Unterraum mit konstanten Inkrementen in zwei Dimensionen zu durchlaufen und die benötigten Werte abzufragen. Speziell bei der Darstellung komprimierter Lichtfelder ist dies von Vorteil, da nur auf die wirklich benötigten Werte zugegriffen wird. Im Gegensatz hierzu benötigt der von Gortler et al. [GORTLER et al. 1996] benutzte Algorithmus eine unkomprimierte Darstellung des gesamten Lichtfeldes. Allerdings kann bei diesem Verfahren Grafikhardware zur Beschleunigung eines großen Anteils des Darstellungs-Algorithmus eingesetzt werden. Das hier benutzte Verfahren läuft dagegen zum größten Teil in Software ab.

Zur Berechnung der Intervallgrenzen $stuv_0, stuv_1, stuv_2, stuv_3$, welche den abzufragenden Parameterbereich des Lichtfeldes beschreiben, werden die Eckpunkte der Bild- auf die Kameraebene projiziert und umgekehrt (Abb. 4.12). Durch Beschränkung der Ergebnisse auf den gültigen Definitionsbereich des Lichtfeldes $[0; res_s - 1] \times [0; res_t - 1] \times [0; res_u - 1] \times [0; res_v - 1]$ werden die Intervallgrenzen $uvst_0, \ldots, uvst_3$ bestimmt. Wenn die Kamera- und die Bildebene des Lichtfeldes nicht gegeneinander verdreht sind, genügt es, die schräg gegenüberliegenden Eckpunkte des Intervalls $uvst_0$ und $uvst_2$ bzw. $uvst_1$ und $uvst_3$ zu berechnen, weil die Intervallgrenzen parallel zu den Kanten der Bild- bzw. Kameraebene verlaufen.

Die zur Darstellung der berechneten Lichtfeld-Ansicht verwendete Textur hat die gleiche Auflösung $res_s \times res_t$ wie die Bildebene des Lichtfeldes. Somit betragen die Inkremente, mit denen das Lichtfeld durchlaufen wird, für die *s*- und *t*-Koordinate jeweils 1. Aus dem zweiten Strahlensatz folgt, dass die Größe der Inkremente in *u*- und *v*-Richtung ebenfalls konstant ist (Abb. 4.13). Hierdurch kann die Berechnung der

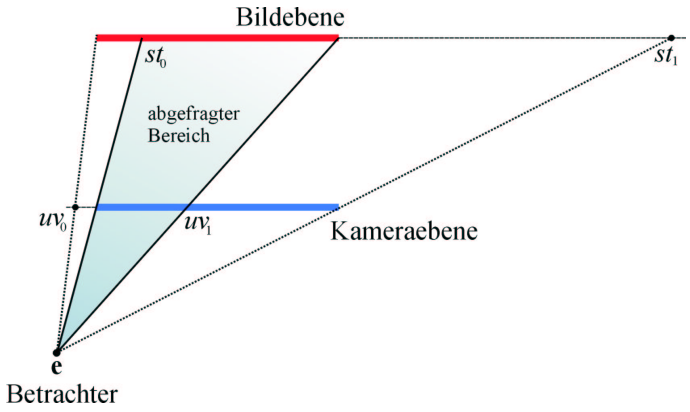


Abbildung 4.13: Zweidimensionale Darstellung der beiden Parameterebenen eines Lichtfeldes. Die Eckpunkte jeder Ebene werden auf die jeweils andere projiziert und so die Grenzwerte für das abzufragende Intervall bestimmt.

Koordinaten der abgefragten Lichtfeldwerte äußerst effizient durchgeführt werden.

Die im folgenden angegebene Prozedur `calcView` berechnet die Ansicht eines Lichtfeldes $\mathcal{L}$ für die Betrachterposition $\mathbf{e}$ und schreibt die berechneten Bildpunkte in ein zweidimensionales Feld p. Der Aufruf $\mathcal{L}(s,t,u,v)$ liefert den Lichtfeldwert an Position (u,v,s,t) zurück.

```
PROCEDURE calcView(e,L,p)
BEGIN
 1. Projiziere die Eckpunkte p_st,0, ..., p_st,3 der Bildebene auf die Kameraebene.
 2. Projiziere die Eckpunkte p_uv,0, ..., p_uv,3 der Kameraebene auf die Bildebene.
 3. Bestimme die Intervallgrenzen stuv_0, ..., stuv_3 aus den projizierten Eckpunkten
    und dem Definitionsbereich von L.
 4. incr_u = (stuv_2,u − stuv_0,u)/(stuv_2,s − stuv_0,s)
 5. incr_v = (stuv_2,v − stuv_0,v)/(stuv_2,t − stuv_0,t)
 6. u = stuv_0,u
 7. FOR s = stuv_0,s TO stuv_2,s − 1 DO
      8. v = stuv_0,v
      9. FOR t = stuv_0,t TO stuv_2,t − 1 DO
           10. p[s,t] = L(s,t,u,v)
           11. v += incr_v
         OD
     12. u += incr_u
    OD
END
```

Interpolation und Tiefenkorrektur

Ein Lichtfeld speichert die reguläre Abtastung eines Teilbereichs der plenoptischen Funktion. Auf Grund der Annahme, dass das im Lichtfeld gespeicherte Objekt von leerem Raum umgeben ist, kann ein Lichtfeld auch als eine Menge von Strahlen, die das Objekt schneiden, aufgefasst werden (Abschnitt 4.1.1).

Ein Betrachter kann während der Betrachtung eines Lichtfeldes jede beliebige Position im Raum einnehmen. Aus diesem Grund ist es wenig wahrscheinlich, dass alle Strahlen, die zur Berechnung der jeweiligen Lichtfeld-Ansicht benötigt werden, genau die Position $\mathbf{e}$ des Betrachter schneiden. Deshalb müssen die benötigten Werte durch Interpolation aus Strahlen gewonnen werden, die in der Nähe von $\mathbf{e}$ verlaufen.

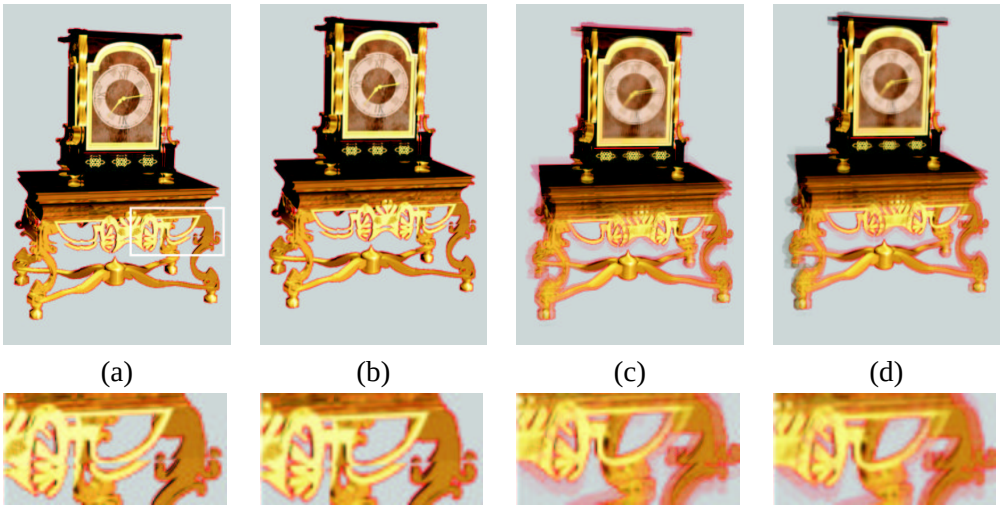


Abbildung 4.14: Vergleich unterschiedlicher Interpolationsmethoden zur Auswertung eines Lichtfeldes. (a) Nearest neighbour Interpolation. In der Vergrößerung des Details (untere Reihe) sind deutliche Artefakte zu erkennen. (b)-(d) Interpolation in *st*-, *uv*- bzw. alle Koordinatenrichtungen. An Stelle der Artefakte treten Unschärfen in verschiedenen Bereichen des Bildes auf.

In Abbildung 4.14 sind die Ergebnisse unterschiedlicher Interpolationsverfahren zusammengestellt. Am einfachsten ist die Rekonstruktion der Lichtfeldwerte mit dem Verfahren der nächsten Nachbarschaft (*nearest neighbour interpolation*). Allerdings können bei einer zu groben Abtastung der Kameraebene Artefakte auftreten (Abb. 4.14 (a)). Diese lassen sich durch eine Interpolation der Koordinaten in *st*-, *uv*- oder alle Richtungen gleichzeitig (Abb. 4.14 (b)-(d)) vermeiden. Allerdings werden die Lichtfeld-Ansichten dann in bestimmten Bereichen unscharf.

Die erkennbaren Artefakte und die Unschärfen in einigen Bildbereichen sind darauf zurückzuführen, dass im Lichtfeld keinerlei Information über die Geometrie des gespeicherten Objekts enthalten ist. Hierdurch kommt es bei einer zu groben Abtastung der Kameraebene zu den beobachteten Effekten. Zur Verdeutlichung wurde in Abbildung 4.14 ein Lichtfeld verwendet, dessen Kameraebene eine Auflösung von nur 8×8 Punkten hat. In Abbildung 4.15 ist der Zusammenhang zwischen der Interpola-

tion von Lichtfeldwerten und der Objektgeometrie am Beispiel der nearest neighbour-Interpolation schematisch dargestellt.

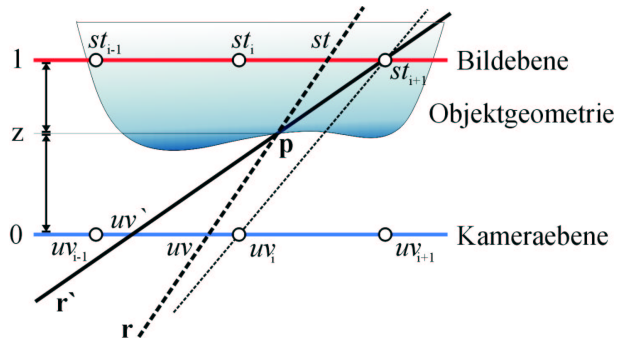
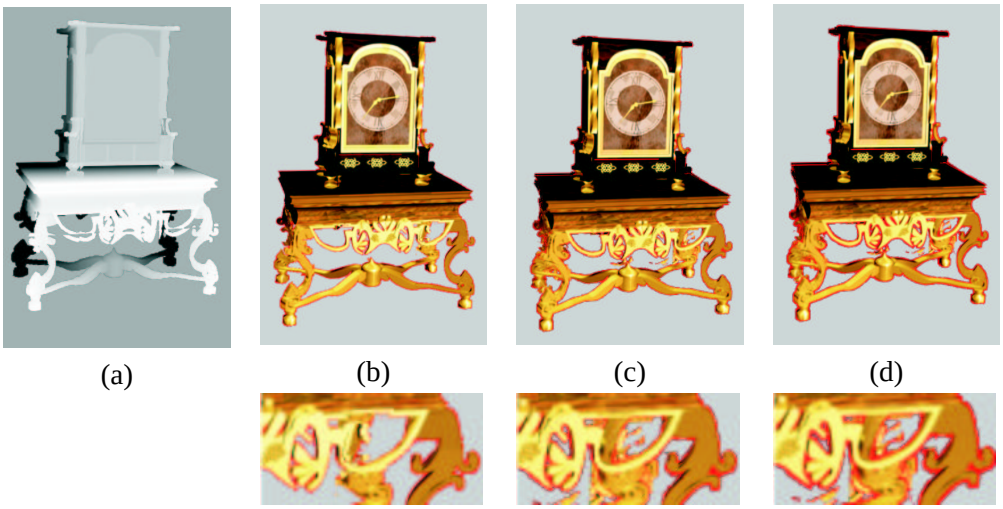


Abbildung 4.15: Schematische Darstellung der Tiefenkorrektur eines Strahls $\mathbf{r}$. Mit dem Tiefenwert z kann der Punkt $\mathbf{p}$, an dem $\mathbf{r}$ erstmals die Objektoberfläche schneidet, bestimmt werden. Der Strahl $\mathbf{r}'$ wird so berechnet, dass er den Abtastpunkt st_{i+1} und $\mathbf{p}$ schneidet.

Im Beispiel auf Abbildung 4.15 schneidet ein Strahl $\mathbf{r}$ Kamera- und Bildebene des Lichtfeldes in den Punkten uv und st. Bei der nearest neighbour interpolation wird der Wert $\mathcal{L}(st_{i+1}, uv_i)$ aus dem Lichtfeld ausgelesen, da die Punkte st_{i+1} und uv_i am nächsten zu st und uv liegen. Ist die Objektgeometrie bekannt, kann mit Hilfe des Tiefenwerts z eine bessere Interpolation berechnet werden. Denn wenn die Position $\mathbf{p}$, an welcher der Strahl $\mathbf{r}$ erstmals die Objektoberfläche schneidet, feststeht, kann für jeden Abtastpunkt auf der Bild- oder Kameraebene ein Strahl berechnet werden, der sowohl den Abtastpunkt als auch $\mathbf{p}$ schneidet [GORTLER et al. 1996]. In Abbildung 4.15 ist der Strahl $\mathbf{r}'$ eingezeichnet, der den Abtastpunkt st_{i+1} und $\mathbf{p}$ enthält. Es ist zu erkennen, dass eine Abfrage von $\mathcal{L}(st_{i+1}, uv_{i-1})$ den gesuchten Wert von $\mathbf{r}$ wesentlich besser approximiert, als der durch nearest neighbour interpolation berechnete Lichtfeldwert.
Zur Verbesserung der Rekonstruktion der Lichtfeldwerte wurde das oben beschriebene Verfahren zur Tiefenkorrektur implementiert. Die Berechnungen können leicht mit Hilfe des 2. Strahlensatzes durchgeführt werden und sind im folgenden Pseudocode zusammengefasst. u_n und v_n sind hierbei die nächsten ganzzahligen Werte zu u und v.

```
PROCEDURE depthCorrect(s,t,u,v,z,L)
BEGIN
 1. Berechne den Tiefenwert z für die Koordinaten (s,t,u,v).
 2. IF z ≠ 0 THEN
     3. s' = s + (1−z)/z (u − u_n)
     4. t' = t + (1−z)/z (v − v_n)
    FI
 5. Bestimme L(s',t',u_n,v_n)
END
```

Da in der Prozedur `calcView` alle abgefragten *s*- und *t*-Koordinaten ganzzahlig sind, schneidet jeder Abfragestrahl $\mathbf{r}$ die Bildebene des Lichtfeldes genau in einem der Abtastpunkte. Zur Verbesserung des Ergebnisses werden die *s*- und *t*-Koordinate mit Hilfe von Tiefeninformationen neu berechnet. Der korrigierte Strahl soll zusätzlich zum Oberflächenpunkt, der durch z bestimmt ist, die Kameraebene bei (u_n, v_n) schneiden. Hierdurch wird eine neue *st*-Koordinate festgelegt. Die Korrektur der *st*-Koordinate ist günstiger als eine Veränderung der *uv*-Koordinate. Da die korrigierte Koordinate (s', t') fast immer nicht-ganzzahlig ist, muss der Wert von $\mathcal{L}(s', t', u_n, v_n)$ bei der nächsten ganzzahligen Koordinate bestimmt werden. Auf Grund der im Allgemeinen höheren Abtastdichte der Bildebene ist diese Näherung genauer als bei der nicht so dicht abgetasteten Kameraebene.



(a) (b) (c) (d)

Abbildung 4.16: Vergleich unterschiedlicher Interpolationsmethoden in Verbindung mit einer Tiefenkorrektur. (a) Verwendetes Tiefenbild. (b)-(d) Interpolation in *st*-, *uv*- bzw. alle Koordinatenrichtungen. Im Unterschied zu Abbildung 4.14 sind auf den Ausschnittsvergrößerungen in der unteren Reihe deutlich weniger Artefakte zu erkennen.

Das Ausmaß der Verbesserung, die sich durch die beschriebene Tiefenkorrektur erzielen läßt, hängt stark von der Qualität der verfügbaren Geometrieinformation ab. Während Gortler et al. [GORTLER et al. 1996] nur ein Dreiecksnetz speichert, kann im Waveletstream zu jedem Pixel ein einzelner Tiefenwert abgelegt werden. Auch ohne das Vorliegen von Informationen über die Objektgeometrie kann eine Tiefenkorrektur durchgeführt werden. In [ISAKSEN et al. 2000] werden eine oder mehrere virtuelle Schärfeebenen im Lichtfeld platziert. Für jedes Pixel der Lichtfeld-Ansicht werden mit Hilfe der Tiefenkorrektur mehrere Strahlen berechnet, die sich in den Schärfeebenen schneiden. Mit diesem Verfahren kann beispielsweise ein Benutzer die in unterschiedlichen Tiefen im Lichtfeld platzierten Objekte genau betrachten.

Abbildung 4.16 zeigt Lichtfeld-Ansichten, welche mit unterschiedlichen Interpolationsverfahren und einer Tiefenkorrektur berechnet wurden. Mit Hilfe der Tiefenwerte aus Abbildung 4.16 (a) wurde eine Interpolation der Koordinaten in *st*-, *uv*- oder beide Richtungen gleichzeitig (Abb. 4.16 (b)-(d)) vorgenommen. Es zeigt sich, dass Artefak-

te trotz der Tiefenkorrektur nicht vollständig zu vermeiden sind. Für die erkennbaren Fehler ist allerdings auch die komplizierte Geometrie der Tischbeine verantwortlich. Ein ideales Verfahren würde deshalb die Objektgeometrie aus den einzelnen Tiefenwerten möglichst genau rekonstruieren. Auf diese Weise könnten Fehlinterpretationen der Daten vermieden werden. Am besten ist die Rekonstruktion auf Abbildung 4.16 (b) gelungen, wo nach der Tiefenkorrektur eine Interpolation in *st*-Richtung durchgeführt wurde.

4.3.11 Beschleunigungsdatenstrukturen

Die Darstellung eines Lichtfeldes am Bildschirm wird durch drei unterschiedliche Caching-Techniken beschleunigt, welche in drei unterschiedlichen Phasen des Darstellungsprozesses eingreifen.

Der Textur-Cache

Der *Textur-Cache* benutzt den eventuell auf der Grafikhardware vorhandenen Texturspeicher, um bereits berechnete Lichtfeld-Ansichten zwischenzuspeichern. Der Texturspeicher ermöglicht eine stark beschleunigte Darstellung texturierter Szenenobjekte, da der Speicher direkt auf der Grafikkarte untergebracht und die Zugriffszeit auf die Texturdaten dadurch extrem gering ist.

Mit einer last recently used (LRU) Strategie werden eine feste Anzahl von Lichtfeld-Ansichten, die im Texturspeicher abgelegt werden, und die zugehörigen Betrachterpositionen verwaltet. Vor Erzeugung einer neuen Lichtfeld-Ansicht wird im Textur-Cache nach der Ansicht gesucht, deren Betrachterposition den geringsten Abstand zu aktuellen Position des Betrachters hat. Wenn der Abstand zwischen der Betrachterposition im Cache und der aktuellen Position des Betrachters einen benutzerdefinierten Grenzwert unterschreitet, wird die im Textur-Cache gespeicherte Lichtfeld-Ansicht angezeigt. In einem Hintergrundprozess wird die korrekte Lichtfeld-Ansicht berechnet und über die Ansicht aus dem Textur-Cache geblendet.

Der Vorteil dieser Technik ist, dass der Zugriff auf den Texturspeicher technisch bedingt sehr schnell erfolgen kann. Leider können die gecachten Lichtfeld-Ansichten nur komplett wieder verwendet werden. Der Unterschied zwischen der Zeit für den Zugriff auf eine Lichtfeld-Ansicht aus dem Textur-Cache und der Zeit für die Erzeugung einer neuen Ansicht ist sehr groß. Dies kann zu einer stark schwankenden Bilderzeugungsrate führen, durch die der Benutzer irritiert werden könnte.

Der bildbasierte Cache

Im Unterschied zum Textur-Cache führt der Einsatz des *bildbasierten Cache* zu gleichmäßigeren Bilderzeugungsraten. Der bildbasierte Cache beruht auf der einfachen Idee, alle Pixel der zuletzt erzeugten Lichtfeld-Ansicht zusammen mit den zugehörigen Lichtfeldkoordinaten (u, v, s, t) zu speichern. Für den bildbasierten Cache wird nur ein zweidimensionales Feld mit $res_s \times res_t$ Einträgen benötigt, weil, wie oben beschrieben, die Auflösung einer Lichtfeld-Ansicht immer mit der Auflösung

des Lichtfeldes in *s*- und *t*-Richtung übereinstimmt. Da die *s*- und *t*-Koordinaten eines Eintrags implizit sind, speichert jeder Cacheeintrag lediglich einen Farbwert und die *u*- und *v*-Koordinate des zugehörigen Lichtfeldwertes.

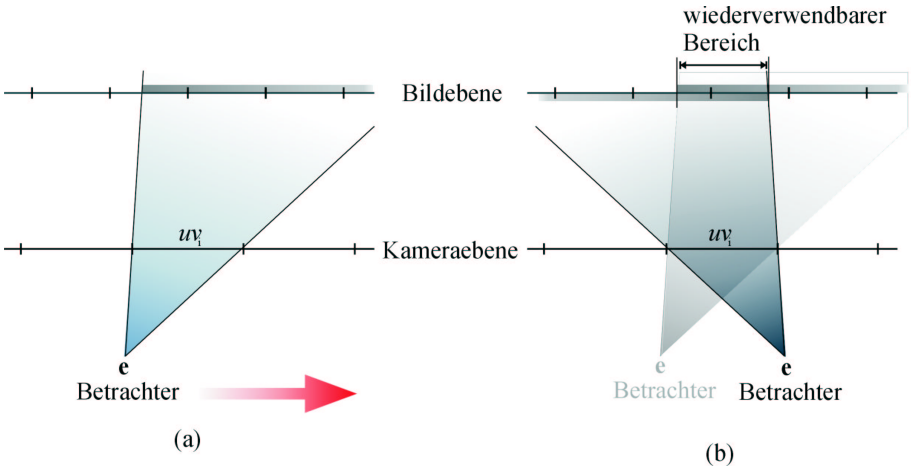


Abbildung 4.17: Schematische Darstellung der Funktionsweise des bildbasierten Cache. Nachdem sich der Betrachter von seiner Position in (a) nach rechts bewegt hat, kann der markierte Bereich auf der Bildebene zur Berechnung der Lichtfeld-Ansicht in (b) wiederverwendet werden, da sich die *uv*-Koordinate im markierten Bereich nicht geändert hat.

In Abbildung 4.17 wird gezeigt, wie der bildbasierte Cache die zeitliche Kohärenz ausnutzt, um die Darstellung von Lichtfeldern zu beschleunigen. Nachdem sich der Betrachter im Beispiel nach rechts bewegt hat, besitzt der markierte Bereich auf der Bildebene die gleichen *uv*-Koordinaten wie vor der Bewegung des Betrachters und kann wiederverwendet werden (Abb. 4.17 (b)). Anschaulich gesehen ist der markierte Bereich der, in dem sich die Projektion des Pixels uv_i für beide Betrachterpositionen überschneidet.

Ein Vorteil dieses Verfahrens ist, dass während der Berechnung einer Lichtfeld-Ansicht ein Vergleich der aktuellen mit den gespeicherten *uv*-Koordinaten ausreichend ist. Stimmen beide überein, kann der Wert aus dem Cache für den aktuellen Pixel benutzt werden. Falls nicht, muss der Lichtfeldwert bestimmt und die neuen *uv*-Koordinaten müssen in den Cache eingetragen werden.

Der bildbasierte Cache speichert nur einen Farbwert pro Pixel im Cache. Wird aber eine der in Abschnitt 4.3.10 beschriebenen Interpolationsmethoden verwendet, dann ist in der Regel mehr als ein Lichtfeldwert an der Berechnung des Farbwertes eines Pixels beteiligt. Um den bildbasierten Cache auch hier einsetzen zu können, muss er geeignet erweitert werden. Jeder Eintrag im Cache speichert dann eine Liste mit Einträgen aus Farbwert und *uv*-Koordinate. Die Liste muss während der Berechnungen jeweils durchsucht und gegebenenfalls aktualisiert werden.

Bildbasierter und Textur-Cache arbeiten unabhängig von der konkreten Repräsentation und Kompression des verwendeten Lichtfeldes und können somit auch zur beschleunigten Darstellung von Lichtfeldern in anderen Datenformaten als dem Waveletstream

genutzt werden. Im Unterschied hierzu ist der im nächsten Abschnitt beschriebene Baum-Cache speziell an den Waveletstream angepasst.

Der Baum-Cache

Der *Baum-Cache* beschleunigt die Rekonstruktion einzelner Lichtfeldwerte aus dem Waveletstream. Dabei wird ausgenutzt, dass sich die abgefragten Koordinaten von unmittelbar hintereinander erzeugten Lichtfeld-Ansichten nur wenig unterscheiden. In Abschnitt 4.3.8 wird beschrieben, wie die Rekonstruktion eines Lichtfeldwertes aus dem Waveletstream der Traversierung eines Pfades im Koeffizientenbaum entspricht. Im Unterschied zum bildbasierten Cache, der ebenfalls die zeitliche Kohärenz bei der Erzeugung von Lichtfeld-Ansichten ausnutzt, wird beim Baum-Cache zusätzlich die Ähnlichkeit der Pfade, entlang denen der Koeffizientenbaum im Waveletstream durchlaufen wird, zur Beschleunigung der Berechnungen ausgenutzt.

Alle Pfade, die zur Berechnung einer Lichtfeld-Ansicht durchlaufen werden müssen, beginnen an der Wurzel des Koeffizientenbaums und besitzen unterschiedlich lange, gemeinsame Teilstücke. Gemeinsam bilden sie einen *Unterbaum* innerhalb des Koeffizientenbaums. Der Baum-Cache speichert diesen Unterbaum.

Bei der Rekonstruktion eines Wertes aus dem Waveletstream wird das längste Teilstück des aktuell zu traversierenden Pfades, das im Baum-Cache gespeichert ist, gesucht. Die Berechnung des aktuell abgefragten Wertes beginnt erst an dem Knoten, an dem der im Baum-Cache gespeicherte Teilpfad endet. Während der Traversierung des Restpfades werden die besuchten Knoten in den Baum-Cache eingetragen.

Der Rang eines Knotens im Unterbaum, den der Baum-Cache speichert, kann höchstens 4 sein. Aus diesem Grund speichert der Baum-Cache an jedem seiner Knoten höchstens 4 Einträge. Diese erlauben den Vergleich von Pfaden, enthalten Informationen zum Nachfolger und ermöglichen den Start der Berechnung am jeweiligen Knoten. Alle Einträge werden nach dem last recently used-Prinzip verwaltet. Mit dieser Strategie kann die Kohärenz zwischen aufeinander folgenden Lichtfeld-Ansichten auf einfache Weise ausgenutzt werden.

In Abschnitt 4.5.3 wird die Wirksamkeit der beschriebenen Caching-Methoden untersucht und werden die einzelnen Verfahren verglichen.

4.3.12 Benutzung von Silhouetten-Information

Häufig werden Lichtfelder verwendet, um einzelne Objekte zu speichern und darzustellen. Dabei ist es im Zusammenhang mit einer Wavelet-Kompression problematisch, dass der Rand eines Objekts, wo die Lichtfeldwerte des Objekts und des Hintergrundes direkt benachbart sind, gewöhnlich eine hohe Ortsfrequenz darstellt. Diese führt jedoch zu einer großen Anzahl von Detailkoeffizienten. Andererseits ist es leicht einzusehen, dass zur Speicherung eines Objekts eine Zahl von Wavelet-Koeffizienten ausreichend sein muss, welche nicht größer als die Zahl der Lichtfeldwerte ist, die zum Objekt gehören.

Um diese Beschränkung in der Anzahl der Wavelet-Koeffizienten zu verwirklichen, muss für jeden Lichtfeldwert dessen Zugehörigkeit zum Objekt im Waveletstream gespeichert werden.

Der Silhouetten-Baum

Der *Silhouetten-Baum* ist eine speichereffiziente Datenstruktur, welche die Zugehörigkeit aller Werte eines Lichtfeldes zum Objekt beschreibt.

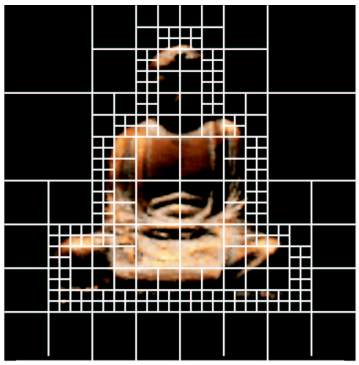


Abbildung 4.18: Unterteilungsschema des Silhouetten-Baums. Nur wenn der durch den jeweiligen Knoten repräsentierte Wertebereich teilweise zum Objekt und teilweise nicht dazu gehört, wird der Knoten weiter unterteilt.

Ähnlich wie ein *Quadtree* im Zweidimensionalen oder ein *Octree* im Dreidimensionalen wird der Silhouetten-Baum (Abb. 4.18) immer dann unterteilt, wenn der durch den jeweiligen Knoten repräsentierte Wertebereich nicht homogen ist. Ein Knoten im Silhouetten-Baum kann drei mögliche Werte haben:

- *Objekt*: Alle durch den Knoten repräsentierten Lichtfeldwerte gehören zum Objekt. Dieser Knoten hat keinen Nachfolger.
- *Nicht-Objekt*: Alle durch den Knoten repräsentierten Lichtfeldwerte gehören nicht zum Objekt. Dieser Knoten hat keinen Nachfolger.
- *Gemischt*: Die durch diesen Knoten repräsentierten Lichtfeldwerte gehören teilweise zum Objekt und teilweise nicht. Die genaue Zugehörigkeit aller Pixel wird erst durch die Nachfolger dieses Knoten beschrieben.

Der Silhouetten-Baum läßt sich gut in den Waveletstream integrieren, da Silhouetten- und Koeffizienten-Baum das Lichtfeld auf dieselbe Art unterteilen. Deshalb kann der eigentlich dreiwertige Silhouetten-Baum durch Ausnutzung von Kohärenz innerhalb des Waveletstream binär kodiert werden. In Abbildung 4.19 ist die für die Speicherung der Silhouetten-Information durchgeführte Erweiterung der node description-Datenstruktur (Abschnitt 4.3.7) dargestellt. Die Silhouetten-Information des im Lichtfeld gespeicherten Objekts ist auf diese Weise vollständig in den Waveletstream eingebettet.

Rekonstruktion mit Silhouetten-Information

Abbildung 4.19: Erweiterung des Waveletstream zur Speicherung der Silhouetten-Information. Die Einträge des succ map und des silhouette map Bitfeldes ermöglichen die Bestimmung der Zugehörigkeit der Lichtfeldwerte zum Objekt.

Um während der Rekonstruktion eines Wertes aus dem Waveletstream dessen Zugehörigkeit zum Objekt zu bestimmen, werden die Einträge aus den succ map- und silhouette map-Bitfeldern verwendet. Wie oben beschrieben, wird der Silhouettenbaum nur so lange unterteilt, bis alle durch einen Knoten beschriebenen Lichtfeldwerte entweder vollständig zum Objekt oder zum Hintergrund gehören. Deshalb muss die Tiefe des Silhouettenbaums nicht an allen Stellen mit der des Koeffizientenbaums übereinstimmen. Steht während der Traversierung eines Pfades die Zugehörigkeit zum Objekt fest, kann die Auswertung der Silhouetten-Information beendet und im Weiteren, wie in Abschnitt 4.3.8 beschrieben, vorgegangen werden.

succ map	silhouette map	Interpretation
1	1	gemischt
1	0	Objekt
0	0	Nicht-Objekt
0	1	Objekt

Tabelle 4.1: Interpretation der Einträge der silhouette map- und succ map-Bitfelder bei der Bestimmung der Zugehörigkeit der Lichtfeldwerte zum Objekt.

So wie das succ map-Bitfeld in der node description eines Knotens N die Existenz von Nachfolgern von N beschreibt, so beschreibt das silhouette map-Bitfeld die Zugehörigkeit der Nachfolgerknoten $N_{succ,i}$ zum Objekt. Durch die Interpretation beider Bitfelder können die Einträge des Silhouettenbaums rekonstruiert werden. Die gemeinsame Bedeutung der Einträge ist in Tabelle 4.1 zusammengefasst.

Falls der Eintrag i in beiden Bitfeldern gesetzt ist, gehört nur ein Teil der durch den Nachfolgerknoten $N_{succ,i}$ repräsentierten Lichtfeldwerte zum Objekt (Knotenwert *gemischt*). Die node description von $N_{succ,i}$ speichert dann wiederum ein silhouette map-Bitfeld. Wenn $N_{succ,i}$ existiert, der Eintrag i des silhouette map-Bitfeld aber nicht gesetzt ist, gehören alle durch $N_{succ,i}$ repräsentierten Werte zum Objekt (Knotenwert *Objekt*). Ist der Eintrag i des succ map-Bitfeldes '0' oder existiert keine succ map, weil keines des any child flags gesetzt ist, entscheidet der Eintrag des silhouette map-Bitfeldes direkt über die Zugehörigkeit zum Objekt.

Berechnung der Koeffizienten

Wenn innerhalb des Waveletstream Silhouetten-Information gespeichert ist, genügt zur Beschreibung eines Lichtfeldes eine Anzahl von Wavelet-Koeffizienten, die kleiner oder gleich der Zahl der Lichtfeldwerte ist, die zum Objekt gehören. Um dies zu verwirklichen, müssen die Wavelet-Koeffizienten speziell berechnet werden. Hierbei wird ausgenutzt, dass die Lichtfeldwerte, die nicht zum Objekt gehören, bei der Rekonstruktion beliebige Werte annehmen können, da sie mit Hilfe der Silhouetten-Information erkannt, jedoch nicht dargestellt werden.

In (4.3) wird beschrieben, wie aus einem Skalierungskoeffizienten $c_{N_{pre}}$, Wavelet-Koeffizienten $d_i \in \mathbf{d}$ und einer Matrix $\mathbf{M}$ Lichtfeldwerte $c_i \in \mathbf{c}$ rekonstruiert werden. Von den $2^{dim(j)}$ durch einen Knoten N repräsentierten Lichtfeldwerten sei n die Anzahl der Werte, die zum Objekt gehören, und $m := 2^{dim(j)} - n$ die Anzahl der Werte, die den Hintergrund darstellen. Ohne Beschränkung der Allgemeinheit und zur Vereinfachung der Notation sind die Werte in $\mathbf{c}$ entsprechend ihrer Zugehörigkeit zum Objekt sortiert. Es gilt entsprechend (4.3):

$$\begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \\ x_0 \\ \vdots \\ x_{m-1} \end{bmatrix} = \mathbf{M} \begin{bmatrix} c_{N_{pre}} \\ d_1 \\ \vdots \\ d_{2^{dim(j)}-1} \end{bmatrix}$$

Dabei sind die x_i die Lichtfeldwerte, die nicht zum Objekt gehören und denen bei der Rekonstruktion beliebige Werte zugewiesen werden können. Um die Lichtfeldwerte $c_i, i = 0, \ldots, n–1$ zu beschreiben, reichen die Skalierungsfunktion ϕ und $n–1$ Wavelets $\psi_i \in \mathbf{\Psi}$ aus. Die ψ_i müssen hierzu so gewählt werden, dass sie zusammen mit ϕ einen Funktionsraum aufspannen, in dem die n Lichtfeldwerte, die zum Objekt gehören, dargestellt werden können.

Zur Bestimmung der neuen Wavelet-Koeffizienten genügt es somit, folgendes Gleichungssystem zu lösen:

$$\begin{bmatrix} c_0 \\ \vdots \\ c_{n-1} \end{bmatrix} = \mathbf{M}' \begin{bmatrix} c_{N_{pre}} \\ d'_1 \\ \vdots \\ d'_{2^{dim(j)}-1} \end{bmatrix} \tag{4.4}$$

wobei $\mathbf{M}'$ eine $m \times m$-Matrix ist, die durch Streichung von m Zeilen und Spalten aus $\mathbf{M}$ entsteht. Die zu entfernenden Zeilen sind dabei durch die $\psi_i \in \mathbf{\Psi}$ und die zu entfernenden Zeilen durch die Position der x_i bestimmt. Die Koeffizienten $c_{N_{pre}}$ und $d'_i, i = 0, \ldots, n-1$ können leicht mit Hilfe der Inversen $\mathbf{M'}^{-1}$ ausgerechnet werden. Die restlichen Wavelet-Koeffizienten haben den Wert Null.

4.3.13 Verwendung anderer Waveletbasen

Die zur Kompression von Lichtfeldern eingesetzte Haar-Waveletbasis besitzt Eigenschaften, die sehr günstig für die interaktive Dekompression und Darstellung von Lichtfeldern sind. Die einfache Struktur der Haar-Basis ermöglicht eine sehr schnelle Ausführung der Dekompression, da nur elementare arithmetische Operationen benötigt werden. Außerdem verringert sich der Aufwand zum Auffinden und zur Rekonstruktion von Lichtfeldwerten (Abschnitt 4.3.9), weil Haar-Wavelets Tree-Wavelets sind. Andererseits sind Haar-Wavelets stückweise konstant und besitzen nur ein verschwindendes Moment. Hierdurch können bei zu starker Kompression blockförmige Artefakte sichtbar werden. Es ist somit nicht möglich, mit der Haar-MSA Funktionen glatt zu approximieren.

Wavelets mit einer höheren Anzahl von verschwindenden Momenten, wie die in Abschnitt 3.2.3 beschriebenen Daubechies-Wavelets, ermöglichen die glatte Approximation beliebiger Funktionen. Obwohl bei Messung des Approximationsfehlers in der zweidimensionalen Bildkompression meist eine Verringerung des Fehlers um nur etwa $10 - 20$ Prozent im Vergleich zur Haar-MSA festgestellt wird [PALL 1993], ist die subjektiv wahrgenommene Verbesserung der Bildqualität deutlich höher. Eine vergleichbare Qualitätssteigerung ist somit auch bei der Kompression von Lichtfeldern zu erwarten. Allerdings sind glatte Wavelets, wie die Daubechies-Wavelets, keine Tree-Wavelets. Die Träger benachbarter Waveletfunktionen einer Skala überlappen sich gegenseitig. Die Benutzung von Nicht-Tree-Wavelets erfordert daher eine Anpassung des im Abschnitt 4.3.9 beschriebenen Verfahrens zur Rekonstruktion eines Lichtfeldwertes aus dem Waveletstream.

Zur Rekonstruktion eines Lichtfeldwertes muss der Koeffizientenbaum entlang eines Pfades traversiert werden, der aus der Koordinate (u, v, s, t) des abgefragten Wertes berechnet wird (Abschnitt 4.3.8). Allerdings sind bei Benutzung von Nicht-Tree-Wavelets nicht alleine die Koeffizienten ausreichend, die in den Knoten N des verfolgten Pfades gespeichert sind, um einen Wert vollständig zu rekonstruieren. Da sich die Träger der Wavelets überlappen, müssen auch Koeffizienten zur Rekonstruktion benutzt werden, die in zu N benachbarten Knoten N_l und N_r auf dem jeweils selben Niveau im Koeffizientenbaum gespeichert sind (Abb. 4.20). Hierdurch erhöht sich der Aufwand bei der Rekonstruktion. Zum einen ist ein Zugriff auf die jeweils benachbarten Knoten erforderlich, zum anderen muss eine größere Anzahl von Werten verarbeitet werden. In Abschnitt 4.3.1 wird hierzu gezeigt, dass die Anzahl der auszuwertenden Koeffizienten mit $O((\frac{n}{2})^4)$ steigt, wenn n die Breite des Trägers ist.

Jedem Knoten wird eine Koordinate zugewiesen, um die zu einem Knoten N benachbarten Knoten im Koeffizientenbaum finden zu können (Abb. 4.20). Die Koordinaten von benachbarten Knoten können leicht aus der Koordinate von N berechnet werden. Um im Beispiel die Knoten N_l und N_r aufzufinden, werden mit Hilfe des auf Abbildung 4.10 erklärten Verfahrens Pfade zu N_l und N_r berechnet und die Knoten von der Wurzel des Koeffizientenbaums her aufgesucht. Der größte Nachteil dieser Vorgehensweise ist, dass zur Rekonstruktion eines einzelnen Lichtfeldwertes der Koeffizientenbaum entlang nicht nur eines, sondern einer Vielzahl von Pfaden durchlaufen werden muss. Die Anzahl der Pfade wächst dabei linear mit der Größe des Trägers der benutzten Wavelets. Durch die Wiederverwendung bereits einmal verfolgter Pfade

mit Hilfe einer angepassten Version des Baum-Cache (Abschnitt 4.3.11) kann dieser Nachteil allerdings etwas gemindert werden.

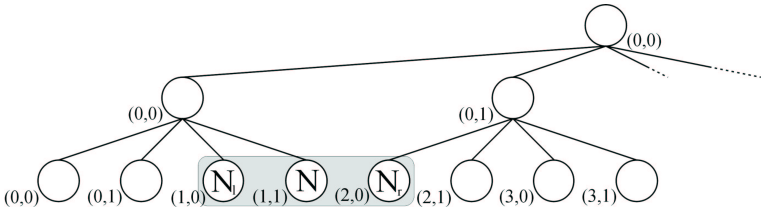


Abbildung 4.20: Beispiel eines Koeffizientenbaums. Die Knoten N_l und N_r, die zur Rekonstruktion von Werten in N besucht werden müssen, sind markiert. Die Koordinaten jedes Knotens erlauben das Auffinden der auf dem gleichen Niveau zu N benachbarten Knoten.

Um die zu einem Knoten N auf demselben Niveau benachbarten Knoten zu finden, wäre es auch möglich, den Waveletstream um zusätzliche Navigationsinformation zu erweitern. Diese würde eine vertikale Bewegung innerhalb des Koeffizientenbaums erlauben. Allerdings verschlechtert sich hierdurch das Kompressionsverhältnis, da bei diesem Ansatz Rechenzeit gegen Speicherplatz eingetauscht wird.

Eine weitere Möglichkeit zur Verringerung des Approximationsfehlers ist die Benutzung von Multiwavelets, die in Abschnitt 3.2.4 vorgestellt wurden. Diese vereinen die Vorteile von Tree-Wavelets mit dem glatten Approximationsverhalten herkömmlicher Wavelets höherer Ordnung. Durch die Benutzung eines Vektors von Wavelets an Stelle einzelner Funktionen erhöht sich allerdings auch die Anzahl der Koeffizienten, die in jedem Knoten gespeichert werden müssen. Selbst wenn eine große Anzahl dieser Koeffizienten während der Kompression wegfällt, müssen für jeden Koeffizient Informationen über dessen Existenz gespeichert werden. Hierzu werden im Waveletstream significance maps benutzt (Abschnitt 4.3.7). Allerdings muss schon bei der zweidimensionalen Bildkompression ein großer Anteil des gesamten Datensatzes für die Speicherung von significance maps aufgewendet werden [SHAPIRO 1993]. Durch die Benutzung vierdimensionaler Multiwavelets vergrößert sich dieses Problem noch mehr. Aus diesem Grund müsste beim Einsatz von Multiwavelets eine andere Methode zur Speicherung der Position der vorhandenen Koeffizienten gefunden werden.

4.4 Implementierung

Das beschriebene Kompressionsverfahren für Lichtfelder und die Waveletstream-Datenstruktur wurden mit der *inReal*-Klassenbibliothek implementiert. Diese bietet eine flexible Plattform zur Implementierung von sowohl bildbasierten als auch klassischen Konzepten in der Computergrafik.

Im Folgenden sollen die Anforderungen und die Kernkonzepte der entwickelten objektorientierten Bibliothek vorgestellt werden. Wo möglich, werden dabei standardisierte UML-Diagramme verwendet [BOOCH et al. 1999, RUMBAUGH et al. 1999].

Der Schwerpunkt der Beschreibung liegt auf der Implementierung der verschiedenen Lichtfeld-Datenstrukturen und deren Darstellung.

4.4.1 Anforderungen

Bei Entwurf und Implementierung der inReal-Klassenbibliothek (kurz *inReal*) mussten die folgenden, teilweise gegensätzlichen Ziele in einem einzigen System verwirklicht werden:

- **Flexibilität.** Da während der Entwicklung des Waveletstream verschiedene Lösungsalternativen getestet und verglichen werden mussten, war ein Systementwurf, der einen Austausch aller wesentlichen Komponenten erlaubt, besonders wichtig. Hierzu mussten leistungsfähige und zugleich allgemeine Schnittstellen entworfen werden.
- **Performance.** Obgleich es sich bei inReal um ein System handelt, das in erster Linie der prototypischen Implementierung neuer bildbasierter Konzepte dient, sollten ausgewählte Teile des Programms besonders schnell ausführbar sein. Dies betraf insbesondere die Dekompression und Darstellung von Lichtfeldern aus dem Waveletstream in Echtzeit.
- **Einfache und erweiterbare Benutzerschnittstelle.** Auf Grund der zahlreichen Optionen bei der Konstruktion des Waveletstream und der Darstellung von Lichtfeldern, musste die Benutzerschnittstelle einen einfachen und intuitiven Zugang zum System gewährleisten. Weiterhin sollte die Benutzerschnittstelle mit geringem Aufwand um neue Funktionen erweiterbar sein.

Um diese Anforderungen zu erfüllen, kam ein flexibler objektorientierter Entwurf zum Einsatz. Dieser erlaubt die Kapselung unterschiedlicher Repräsentationen für Lichtfelder, den Wechsel der Parametrisierung und die Kombination unterschiedlicher Darstellungs- und Cachingmethoden. Auf diese Weise war der Vergleich verschiedener Ansätze auf einfache Weise möglich.

4.4.2 Ein flexibler objektorientierter Systementwurf für die Darstellung von Lichtfeldern

In Abbildung 4.21 sind die wesentlichen Komponenten, in die das inReal-Programmdesign zerfällt, und der Datenfluss zwischen diesen dargestellt. Alle vom *Parser* gelesenen Daten werden in einer zentralen *Datenbank* abgelegt und stehen dort allen anderen Programmteilen zur Verfügung. Die Komponente *Neukodierung* ermöglicht danach die Veränderung schon existierender Datensätze. Um den großen Speicherplatzbedarf bei der Erzeugung eines Waveletstream aus einem Lichtfeld-Datensatz zu reduzieren (*Konvertierung*), kann dieser in jeder Koordinatenrichtung mehrmals unterteilt werden. Die so entstandenen Teil-Lichtfelder werden jeweils einzeln in einen Waveletstream umgerechnet. Eine spezielle Klasse wählt während der Dekompression, entsprechend dem abgefragten Wert, aus der Menge von Waveletstreams jeweils den richtigen Datensatz zur Dekompression aus.

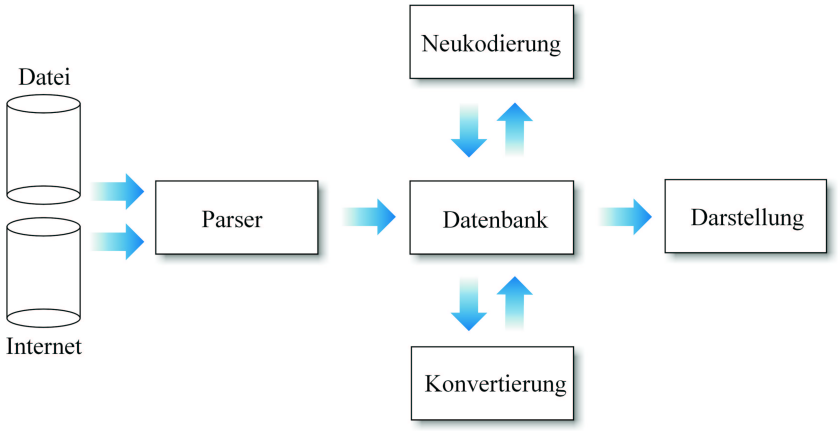


Abbildung 4.21: Elementare Komponenten der inReal-Klassenbibliothek und der Datenfluss zwischen diesen.

Da die Repräsentation und Darstellung der Szene am Bildschirm in diesem Zusammenhang am wichtigsten ist, wird die Komponente *Darstellung* im Folgenden genauer beschrieben.

Szenenrepräsentation und -darstellung

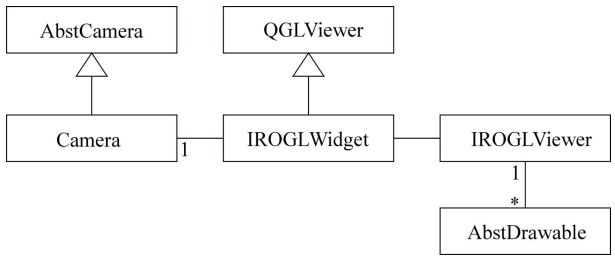


Abbildung 4.22: Die Klassen der inReal-Ausgabeschnittstelle. Jedem Bildschirmfenster ist eine Kamera und eine Szenenrepräsentation zugeordnet. Benutzereingaben werden von der Klasse `IROGLWidget` empfangen und weitergeleitet. Die Szenenelemente werden von der Klasse `IROGLViewer` verwaltet.

Mit inReal können eine beliebige Anzahl grafischer Ausgabeschnittstellen verwaltet werden. Jeder Ausgabeschnittstelle ist eine Kamera und eine Szenenrepräsentation zugeordnet. Das in Abbildung 4.22 dargestellte Diagramm zeigt die Klassen `IROGLViewer` und `IROGLWidget`, die gemeinsam eine Ausgabeschnittstelle in Form eines Bildschirmfensters realisieren. Zur Darstellung wird die von `IROGLViewer` verwaltete Szenenrepräsentation systematisch durchlaufen und alle (aktiven) Elemente dargestellt. Benutzereingaben zur Veränderung der Betrachterposition oder eine Änderung des Darstellungsmodus werden von `IROGLWidget` an die Kameraklasse und `IROGLViewer` weitergeleitet.

Das zur Szenenrepräsentation und -darstellung implementierte Konzept ist die Variation eines schon häufig erfolgreich eingesetzten Designs (z. B. [WERNECKE 1994, PG MOCART 1996]). Die enge Verwandschaft von inReal zu existierenden Systemen ermöglicht die einfache Übertragung und Integration von Komponenten und Konzepten aus inReal in andere Visualisierungssysteme (siehe Kapitel 5).

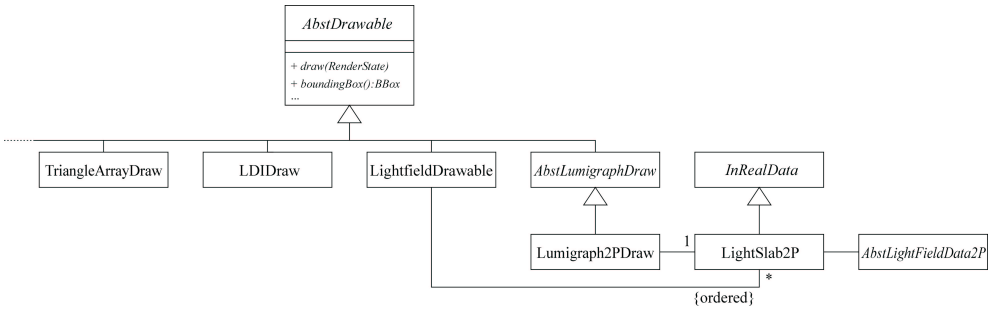


Abbildung 4.23: Alle Szenenelemente werden von einer gemeinsamen abstrakten Oberklasse abgeleitet und implementieren auf diese Weise die gleiche Schnittstelle. Manche Klassen sind mit zusätzlichen Daten assoziiert, die in der zentralen Datenbank abgelegt werden.

inReal implementiert eine Reihe der in der klassischen Computergrafik üblichen elementaren Körper und Oberflächen, sowie eine Anzahl von bildbasierten Darstellungen wie Lichtfelder und LDIs (Abb. 4.23). Alle Szenenelemente sind von der gemeinsamen abstrakten Oberklasse `AbstDrawable` (Abb. 4.23) abgeleitet. Die hierdurch vorgegebene Schnittstelle, die von allen konkreten Klassen implementiert werden muss, ermöglicht die einheitliche Behandlung aller Szenenelemente. Neben der `draw`-Methode zur Darstellung am Bildschirm müssen alle Szenenobjekte beispielsweise Auskunft über ihre Bounding-Box geben. Mit dieser kann die Position der Kamera so bestimmt werden, dass die gesamte Szene sichtbar ist. Zur Übermittlung des aktuell gültigen Zustandes während der Ausgabe, wie beispielsweise die Kameraposition, wird die Klasse `RenderState` verwendet, welche bei jedem Aufruf der `draw`-Methode übergeben wird.

Manche der von `AbstDrawable` abgeleiteten Klassen sind mit solchen assoziiert, welche die darzustellenden Daten speichern. So halten beispielsweise `Lumigraph-2PDraw` und `LightFieldDrawable` Referenzen auf Instanzen von `LightS-lab2P`, welche die eigentlichen Lichtfeld-Daten enthalten. Hierdurch ist die Datenorganisation des dargestellten Lichtfeldes transparent und kann problemlos ausgetauscht werden. Außerdem können alle von `inRealData` abgeleiteten Klassen in der zentralen Datenbank abgelegt und von anderen Programmteilen bearbeitet werden.

Darstellung von Lichtfeldern

Die eigentlichen Lichtfeld-Daten werden in verschiedenen, von `AbstLightField-Data` abgeleiteten Klassen gespeichert. Diese implementieren Lichtfelder in unterschiedlichen Darstellungen und Parametrisierungen (Abb. 4.24). Die abstrakte Ober-

klasse `AbstLightFieldData2P` ermöglicht eine spätere Erweiterung des Systems um Lichtfelder mit anderen als der 2P-Parametrisierung (Abschnitt 4.1.2). Während `LightFieldDataBox` einen einfachen Testdatensatz, `LightFieldData` unkomprimierte und `LightFieldDataVQ` mit Vektor-Quantisierung komprimierte Lichtfelder implementiert, werden in der Klasse `LightFieldWavelet` Lichtfeld-Daten im Waveletstream-Format gespeichert.

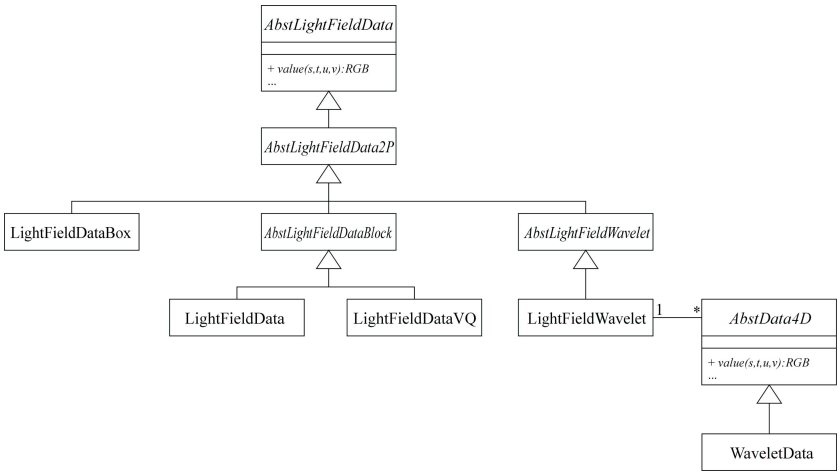


Abbildung 4.24: Alle Klassen, die Lichtfelder speichern, sind von einer gemeinsamen Oberklasse `AbstLightFieldData` abgeleitet. `LightFieldWavelet` speichert die eigentlichen Daten in `WaveletData`, damit unterschiedliche Strategien zur Aufteilung der Daten getestet werden können.

Die nochmalige Aufteilung der Wavelet-komprimierten Lichtfelder in die Klassen `LightFieldWavelet` und `WaveletData` bzw. `AbstData4D` war notwendig, um unterschiedliche Strategien bei der Aufteilung der zu komprimierenden Daten testen zu können. Mit Hilfe dieses Designs konnte überprüft werden, ob die getrennte oder die gemeinsame Kompression von Datenkanälen effizienter ist. `WaveletData` verwaltet nur eine Menge von vierdimensionalen Datensätzen, die jeweils eine beliebige Anzahl von Datenkanälen speichern. Die abstrakte Oberklasse `AbstData4D` wurde eingefügt, da `WaveletData` eine parametrisierte Klasse (Abb. 4.25) ist. Mit diesem Design konnte die Implementierung in C++ stark vereinfacht werden.

Bei der interaktiven Dekompression und Darstellung von Lichtfeld-Ansichten mit dem Waveletstream wirken die in Abbildung 4.25 dargestellten Klassen zusammen.

`Lumigraph2PDraw` implementiert das in Abschnitt 4.3.10 beschriebene Verfahren zur Berechnung einer Lichtfeld-Ansicht. Nach Aufruf der `draw`-Methode wird zunächst versucht, möglichst große Teile der benötigten Lichtfeld-Ansicht aus dem Textur-Cache (`TextureCache`) und dem bildbasierten Cache (`ImageBasedCache`) zu rekonstruieren. Die nicht in den Caches gefundenen Lichtfeldwerte werden von `LightSlab2P` abgefragt. Diese Klasse kapselt die aktuelle Einstellung zum Interpolationsverfahren (Abschnitt 4.3.10) und rekonstruiert die Lichtfeldwerte entsprechend. Die benötigten Lichtfeldwerte werden von `LightFieldWavelet` geliefert. In `WaveletData` wird hierzu der Waveletstream durchlaufen, der gesuchte Wert re-

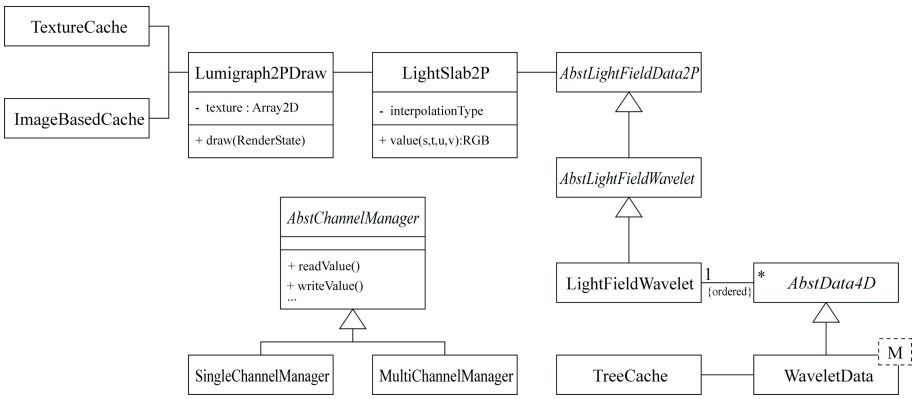


Abbildung 4.25: Die an der Berechnung einer Lichtfeld-Ansicht mit dem Waveletstream beteiligten Klassen.

konstruiert und zurückgegeben. Dabei wird der Baum-Cache (`TreeCache`) benutzt, um möglichst viele Ergebnisse aus vorangegangenen Berechnungen wieder zu verwenden.

Wie in Abschnitt 4.5.3 gezeigt, werden etwa 80 Prozent der Rechenzeit bei der Rekonstruktion einer Lichtfeld-Ansicht aus dem Waveletstream zum Lesen und zur Interpretation des Bytestreams verwendet. Deshalb musste gerade beim Entwurf der `WaveletData`-Klasse und der zugehörigen Hilfsklassen darauf geachtet werden, dass einerseits eine besonders schnelle Implementierung der entsprechenden Funktionalität, andererseits eine möglichst große Flexibilität gewährleistet ist. Die leichte Änderbarkeit in diesem Bereich ist wichtig, da gerade hier zahlreiche Alternativen der Datenstruktur zu testen und bewerten waren.

Um beide Ziele zu verwirklichen, wird `WaveletData` mit einer Klasse `M` parametrisiert. Die beiden als Parameter verwendeten Klassen `SingleChannelManager` und `MultiChannelManager` kapseln die Organisation der Daten und erlauben so die einheitliche Behandlung von Waveletstreams mit einer beliebigen Kombination von Datenkanälen. Da die Klassen als C++-Template-Argumente übergeben werden, sind zur Laufzeit keine Fallunterscheidungen nötig. Alle Entscheidungen über aufzurufende Methoden werden bei diesem Design schon zur Übersetzungszeit getroffen. Weitere Hilfsklassen wie `Flag` und `StreamIterator` abstrahieren vom Bytestream, in dem der Waveletstream gespeichert ist. `Flag` ermöglicht beispielsweise die Abfrage und das Setzen einzelner Werte in Bitfeldern, ohne dass zur Ausführungszeit auf Geschwindigkeit verzichtet werden muss. Die Klasse `StreamIterator` unterstützt die Navigation und das Lesen von Werten aus dem Waveletstream.

Systemumgebung

Die inReal-Klassenbibliothek wurde in C++ implementiert. Als Entwicklungsplattform wurde ein PC mit Windows 2000 und dem Microsoft Visual Studio verwendet.

Zur Implementierung der grafischen Benutzeroberfläche (*Graphical User Interface (GUI)*) wurde die *Qt-Klassenbibliothek* von Trolltech [Trolltech] verwendet (Abb.

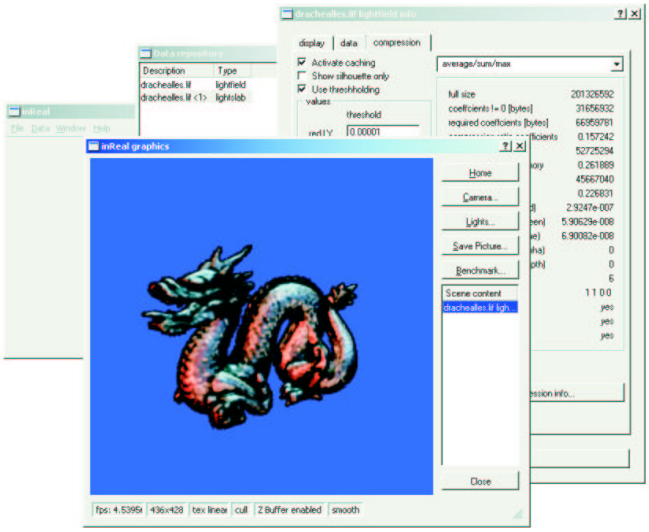


Abbildung 4.26: Einige Elemente der inReal-Benutzeroberfläche. Auf Grund seiner einfachen objektorientierten Struktur und um die Portierung von inReal auf andere Betriebssystem-Plattformen zu erleichtern, wurde die Qt-Bibliothek zur Programmierung der grafischen Benutzeroberfläche (GUI) benutzt.

4.26). Obgleich das Aussehen der GUI-Elemente in Qt nicht immer genau mit denen des original Windows-GUI übereinstimmt und nicht jedes Element der Windows-GUI eine Entsprechung in Qt besitzt, überwiegen die Vorteile beim Einsatz der Qt-Klassenbibliothek. Die einfache und logische Struktur von Qt ermöglicht die schnelle Implementierung auch großer und komplexer Benutzerschnittstellen. Die properitäre Spracherweiterung der C++-Klassendefinition um *signals* und *slots* vereinfacht die Programmierung von Kommunikation zwischen verschiedenen GUI-Elementen erheblich. Weiterhin ist Qt für verschiedene Betriebssysteme, wie Windows, Linux, verschiedene UNIX-Derivate und MacOS, verfügbar. Dies erleichtert eine spätere Portierung von mit Qt geschriebenen Programmen auf andere Betriebssystemplattformen.

4.5 Ergebnisse

Zum Abschluß dieses Kapitels werden die Ergebnisse unterschiedlicher Tests und Vergleiche, mit denen die Eigenschaften des Waveletstreams bewertet wurden, dargestellt. Neben elementaren Messwerten, wie dem Kompressionsfehler und der Darstellungsgeschwindigkeit, werden auch die Auswirkungen der verschiedenen Caching-Verfahren und Optimierungen des Waveletstreams untersucht.

4.5.1 Verwendete Testdaten und Fehlermaße

Zum Test des Waveletstreams wurden die frei verfügbaren Lichtfelder *Drache* und *Buddha* vom *Stanford Light Fields Archive* [SLFA] verwendet (Abb. 4.27). Hierdurch können die Ergebnisse mit anderen Arbeiten auf dem Gebiet verglichen werden.

Drache

Buddha

Abbildung 4.27: Die beiden Lichtfelder, die zur Berechnung der Testreihen verwendet wurden. Beide sind vom *Stanford Light Fields Archive* [SLFA] frei erhältlich.

In Tabelle 4.2 sind die wichtigsten Kenndaten der Lichtfelder Drache und Buddha angegeben.

Lichtfeld	Drache	Buddha
Auflösung	$256 \times 256 \times 32 \times 32$	$256 \times 256 \times 32 \times 32$
Datenformat	24 Bit RGB	24 Bit RGB
unkomprimierte Größe	201.236.592 Bytes	201.236.592 Bytes

Tabelle 4.2: Die wichtigsten Daten der beiden Lichtfelder, die für die Tests verwendet wurden.

Da die unkomprimierten Ausgangsdaten vollständig zur Verfügung standen, konnte der Kompressionsfehler korrekt gemessen werden. Hierzu wurde der auf der L^2-Norm basierende ϵ_{rms}-Fehler und die im Bereich der Signalverarbeitung verbreitete *Peak Signal to Noise Ratio (PSNR)* verwendet. Die Fehlermaße werden für Lichtfelder $\mathcal{L}$ und $\bar{\mathcal{L}}$ wie folgt definiert:

$$\epsilon_{rms} := \left[\frac{1}{res_s res_t res_u res_v} \sum_s \sum_t \sum_u \sum_v \left[\mathcal{L}(s,t,u,v) - \bar{\mathcal{L}}(s,t,u,v) \right]^2 \right]^{\frac{1}{2}}$$

und

$$PSNR := 20\, log_{10}(\frac{1}{\epsilon_{rms}})$$

Lichtfelder enthalten im Allgemeinen RGB- oder YUV-Werte, die mit 3×8 Bit Genauigkeit gespeichert werden. Zur Berechnung des Fehlers wurden die auf den Wert 1 normierten Kanäle getrennt behandelt. Der Gesamtfehler ergibt sich aus dem arithmetischen Mittel der Fehler der einzelnen Kanäle. Auf allen folgenden Diagrammen wird der Kompressionsfehler als Funktion der Datenrate angegeben. Diese bezieht sich jeweils auf die Dateigröße des Waveletstreams. Durch Einfügen der Navigationsinformation während des Einlesens, wie in Abschnitt 4.3.8 beschrieben, wird der Waveletstream im Durchschnitt um etwa 30 Prozent vergrößert.

4.5.2 Kompressionsfehler

Zur visuellen Bewertung des Kompressionsfehlers werden in Abbildung 4.29 Ansichten von unterschiedlich stark komprimierten Drachen- bzw. Buddha-Lichtfeldern gegenüber gestellt. Mit wachsender Kompression nimmt die Bildqualität geringfügig ab, gleichwohl sind nur bei sehr hoher Kompression Artefakte zu erkennen (Abb. 4.29, unterste Reihe).

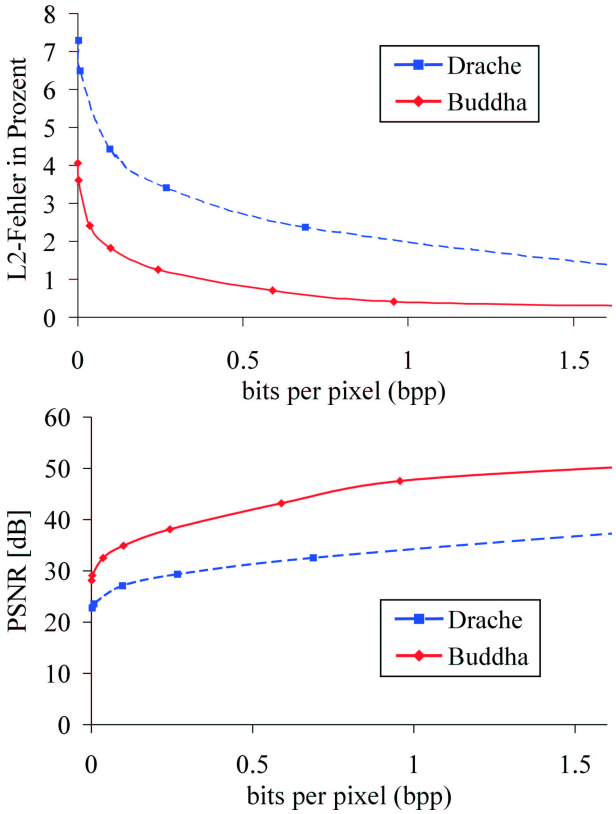


Abbildung 4.28: ϵ_{rms} und PSNR für das Drachen- und Buddha-Lichtfeld in Abhängigkeit von der Datenrate. Die Kurven für ϵ_{rms} schneiden die X-Achse nicht, da der Waveletstream durch die Löschung von Koeffizienten mit Wert Null eine initiale Kompression besitzt. Der geringe auftretende Fehler ist auf die Quantisierung der Koeffizienten zurückzuführen.

Die gemessenen Fehler (Abb. 4.28) bestätigen den guten Eindruck der visuellen Bewertung. Es ist zu erkennen, dass das Buddha-Lichtfeld besser komprimiert wird, da dessen Oberfläche glatter ist und eine nur geringe Zahl von Details aufweist. Keine der Messkurven für ϵ_{rms} schneidet die X-Achse. Dies ist damit zu erklären, dass durch die Entfernung von Wavelet-Koeffizienten mit dem Wert Null der Waveletstream initial komprimiert wird. Der geringe gemessene Kompressionsfehler kommt durch die Quantisierung der Koeffizienten zu Stande.

$2,94$ bpp, $\epsilon_{rms} = 0,0081$

$0,95$ bpp, $\epsilon_{rms} = 0,0042$

$1,78$ bpp, $\epsilon_{rms} = 0,012$

0.24 bpp, $\epsilon_{rms} = 0,0126$

$0,68$ bpp, $\epsilon_{rms} = 0,023$

$0,099$ bpp, $\epsilon_{rms} = 0,0183$

Abbildung 4.29: Das Drachen- und das Buddha-Lichtfeld, unterschiedlich stark komprimiert. Die Bildqualität nimmt mit wachsender Kompression des Waveletstreams ab. Allerdings sind nur bei sehr hoher Kompression Artefakte zu erkennen (unterste Reihe).

Im Diagramm auf Abbildung 4.30 werden nochmals der L^2-Fehler für den Waveletstream und die in [LEVOY und HANRAHAN 1996] benutzte Vektor-Quantisierung verglichen. Für diese Messungen wurde abweichend ein etwas geringer aufgelöstes Drachen-Lichtfeld mit $256\times 256\times 8\times 8$ Werten benutzt. Leider war kein Vergleich zwischen Waveletstream und Vektor-Quantisierung für hohe Kompressionsraten möglich, da mit der Software des Stanford Lightfields Archive keine hoch komprimierten Lichtfelder erzeugt werden können.

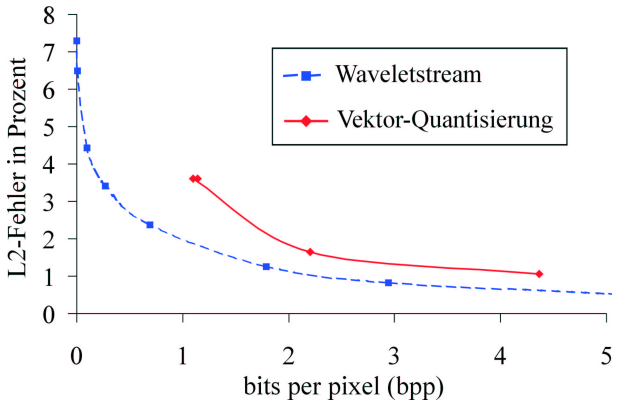


Abbildung 4.30: Vergleich von Waveletstream und Vektor-Quantisierung für das Drachen-Lichtfeld. Der Waveletstream ist in Bezug auf den Kompressionsfehler und die erreichbaren Kompressionsraten deutlich überlegen.

4.5.3 Darstellungsgeschwindigkeit

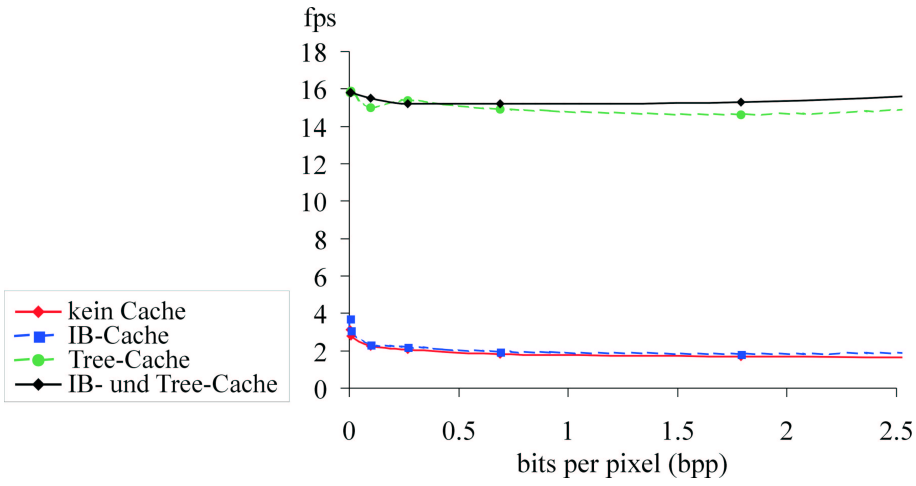


Abbildung 4.31: Bilderzeugungsraten für unterschiedlich stark komprimierte Waveletstreams und verschiedene Caching-Verfahren. Wie nicht anders zu erwarten, liefert eine Kombination aller Verfahren fast immer die besten Ergebnisse.

Die in Abschnitt 4.3.11 beschriebenen Beschleunigungsdatenstrukturen ermöglichen die Rekonstruktion und Darstellung von Lichtfeld-Ansichten aus dem Waveletstream in Zeiten, die für interaktive Anwendungen ausreichend sind. In Abbildung 4.31 wird die Wirkung der unterschiedlichen Caching-Verfahren für das Drachen-Lichtfeld verglichen. Erst mit Hilfe des Baum-Cache wird eine interaktive Bilderzeugungsrate *(frames per second (fps))* erreicht. Mit wachsender Kompression wird die Darstellung leicht beschleunigt, da sich die Anzahl der zu verarbeitenden Koeffizienten und die Länge der im Koeffizientenbaum zu traversierenden Pfade verringert. Auf die Darstellung der Messergebnisse für den Textur-Cache wurde verzichtet, da die Bilderzeugungsrate sehr stark schwankt, je nachdem ob eine passende Lichtfeld-Ansicht aus dem Cache wiederverwendet werden kann.

Die Messungen wurden auf einem PC mit einem 1,4 GHz AMD Athlon Prozessor, 512 MByte RAM und einer ELSA Gladiac 511 Grafikkarte mit GForce2 MX Chipsatz durchgeführt.

Auffällig an der Messung der Bilderzeugungsrate auf Abbildung 4.31 ist, dass die Verwendung des bildbasierten Cache keine Beschleunigung der Darstellung bewirkt. Dies liegt an der hohen Auflösung der Kameraebene des verwendeten Lichtfeldes. Hierdurch ändert sich die *u*- und die *v*-Koordinate jedes Pixels einer Lichtfeldansicht sehr schnell. Die zeitliche Kohärenz kann so nur selten ausgenutzt werden.

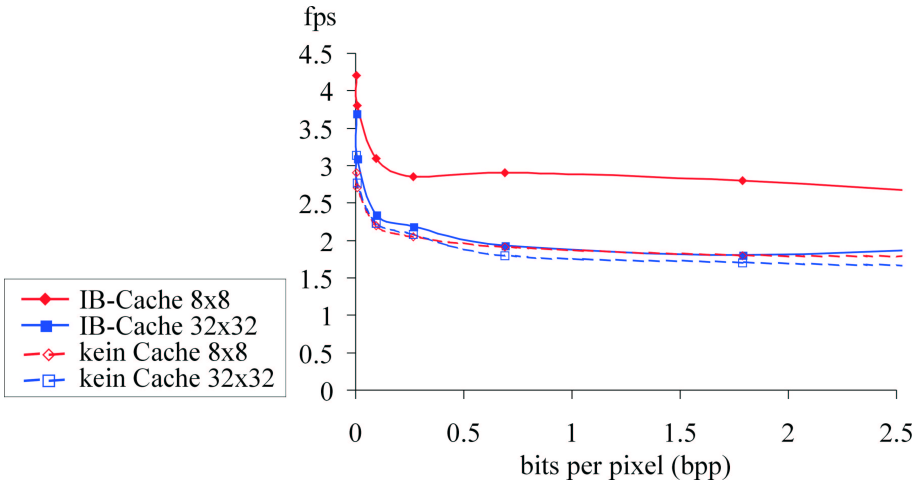


Abbildung 4.32: Vergleich der Bilderzeugungsraten für Drachen-Lichtfelder mit Auflösungen von $256 \times 256 \times 8 \times 8$ bzw. $256 \times 256 \times 32 \times 32$ Werten. Der bildbasierte Cache bewirkt beim geringer aufgelösten Lichtfeld eine deutliche Beschleunigung der Darstellung.

Wird die Auflösung der Kameraebene des Lichtfeldes verringert, werden die Vorteile des bildbasierten Cache deutlich. In Abbildung 4.32 werden hierzu die Bilderzeugungsraten für Drachen-Lichtfelder mit Auflösungen von $256 \times 256 \times 8 \times 8$ und $256 \times 256 \times 32 \times 32$ Werten verglichen. Der Einsatz des bildbasierten Cache zeigt nur beim geringer aufgelösten Lichtfeld deutliche Vorteile.

Im Zusammenhang mit den gemessenen Bilderzeugungsraten ist die Aufteilung der

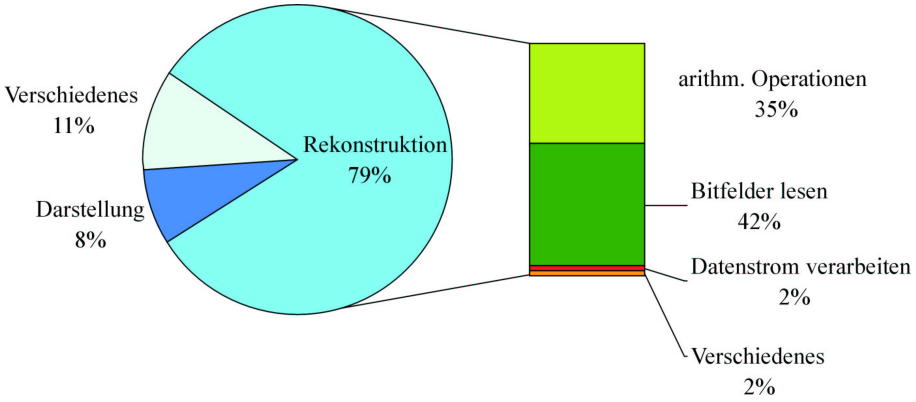


Abbildung 4.33: Aufteilung der Rechenzeit auf die verschiedenen Programmteile. Der größte Teil der Zeit wird für die Rekonstruktion der Lichtfeldwerte aus dem Waveletstream benötigt. Hierbei muss für die arithmetischen Operationen und das Lesen der Bitfelder jeweils fast die Hälfte der Zeit aufgewendet werden.

Rechenzeit auf die verschiedenen Programmteile von Interesse. Grafik 4.33 zeigt Werte, wie sie bei einer Darstellung des Drachen-Lichtfeldes ohne Benutzung von Caches ermittelt wurden. Annähernd 80 Prozent der Rechenzeit werden für die eigentliche Rekonstruktion der Lichtfeldwerte aufgewendet. Hierbei benötigen die arithmetischen Operationen auf den Wavelet-Koeffizienten und die Verarbeitung der Bitfelder jeweils fast die Hälfte der Zeit.

Diese Messwerte bestätigen die in Abschnitt 4.4.2 dokumentierten Designentscheidungen beim Entwurf der inReal-Klassenbibliothek. Es wird klar, dass der `Flag`-Klasse besondere Bedeutung für eine schnelle Ausführung der Operationen zukommt. Hierzu stellt diese einerseits eine einfach zu benutzende Schnittstelle zum Lesen und zur Manipulation von Bitfeldern zur Verfügung, anderseits werden die benötigten Operationen innerhalb der Klasse besonders performant implementiert. Außerdem wird deutlich, dass der Baum-Cache die größte Verbesserung von allen Beschleunigungsansätzen bewirkt, weil durch diesen die Häufigkeit der Ausführung der besonders rechenintensiven Anteile des Verfahrens verringert werden kann.

4.5.4 Multi-Skalen Darstellung

Die Vorteile der progressiven Übertragung der Lichtfeld-Daten im Waveletstream werden am folgenden Beispiel demonstriert. Während bei herkömmlicher Anordnung der Daten das Lichtfeld erst vollständig gelesen werden muss, ist schon mit etwa 12 Prozent der Waveletstream-Daten eine Darstellung des Lichtfeldes mit guter Qualität möglich (Abb. 4.34). Zum Vergleich wurden ein mit Vektor-Quantisierung komprimiertes Lichtfeld und ein Waveletstream mit dem ungefähr gleichen Kompressionsverhältnis von $1:24$ verwendet.

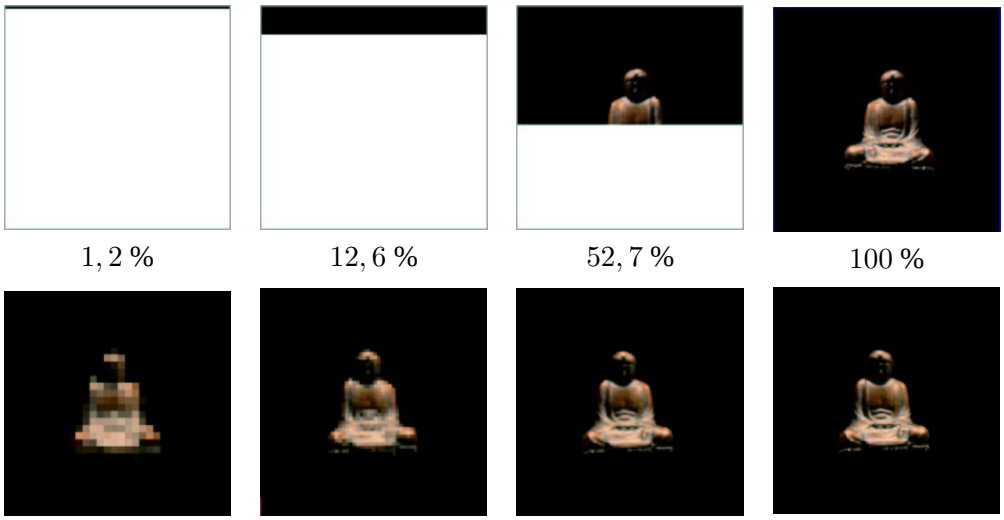


Abbildung 4.34: Vergleich der Darstellung eines Lichtfeldes mit wachsendem Anteil von geladenen Daten. Das in der oberen Reihe abgebildete Buddha-Lichtfeld ist mit Vektor-Quantisierung komprimiert und erst nach dem vollständigen Einlesen aller Daten sinnvoll nutzbar. Im Unterschied dazu kann das im Waveletstream gespeicherte Lichtfeld mit etwa 12 Prozent der Daten in guter Qualität dargestellt werden.

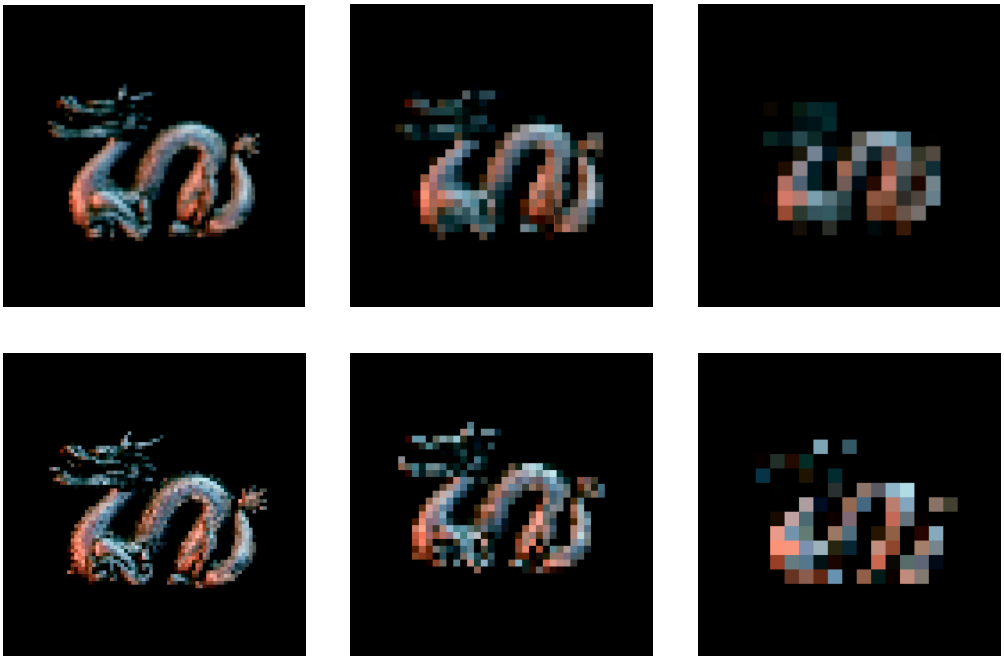


Abbildung 4.35: Vergleichende Darstellung eines Lichtfeldes in unterschiedlichem Abstand zum Betrachter (Wachsender Abstand von links nach rechts). Wird der Abstand zum Betrachter erhöht, sind bei der ungefilterten Darstellung in der unteren Reihe Artefakte zu erkennen. Die Auswahl der richtigen Skala aus dem Waveletstream in der oberen Reihe ermöglicht eine visuell ansprechende Lichtfeld-Rekonstruktion.

Ein weiterer Vorteil der Wavelet-Kompression ist die einfache Darstellung des Lichtfeldes auf unterschiedlichen Skalen. In Abbildung 4.35 ist das Drache-Lichtfeld in verschiedenen Abständen zum Betrachter dargestellt. Während bei nicht angepasster, weil ungefilterter, Darstellung Artefakte erkennbar sind (Abb. 4.35, untere Reihe), ermöglicht die Auswahl der richtigen Skala im Waveletstream (Abb. 4.35, obere Reihe) ein visuell ansprechendes Ergebnis. Zur Verdeutlichung wurden alle Bilder gegenüber der Darstellung am Bildschirm vergrößert.

4.5.5 Verbesserungen des Waveletstreams

In Abschnitt 4.3 sind viele kleine und große Verbesserungen des Waveletstreams beschrieben. In Abbildung 4.36 wird deshalb die Kompression für Waveletstreams mit und ohne Benutzung aller Optimierungen verglichen.

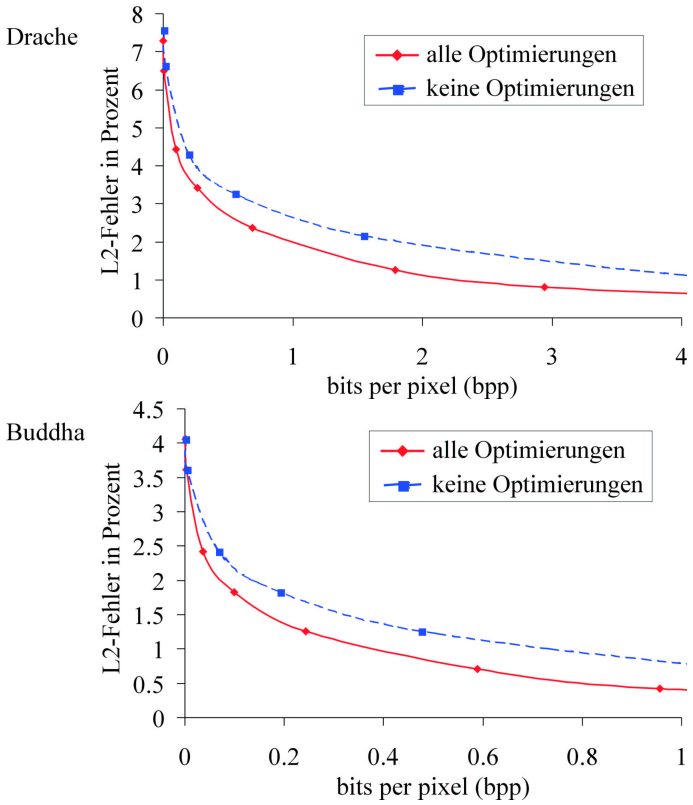


Abbildung 4.36: Vergleich des Kompressionsfehlers in Abhängigkeit von der Datenrate für einen Waveletstream mit und ohne Verbesserungen. Bei Einsatz aller Optimierungen gleichzeitig wird die Kompression bei gleichem Fehler mehr als verdoppelt.

Beim optimierten Waveletstream ist die Länge der Offsets an den wirklichen Bedarf angepasst. Außerdem speichert dieser neben Silhouetten-Information, Minima und Maxima zur Verbesserung der Quantisierung und Einträge im YUV-Format. Im Gegensatz dazu enthält der nicht optimierte Datensatz RGB-Werte. Eine Kombination aller Optimierungen bewirkt eine deutliche Steigerung der Kompressionsraten bzw.

Senkung des Kompressionsfehlers. Die Kompressionsrate wird bei gleichem Fehler teilweise um über 100 Prozent verbessert. Dabei wirken sich die verschiedenen Optimierungen sehr unterschiedlich aus.

Der Effekt, den die Anpassung der Länge der Offset-Information an den wirklichen Bedarf hat, muss als marginal bezeichnet werden. Durch diese Maßnahme konnte die Größe des Waveletstreams bei den durchgeführten Messungen um höchstens $2,5$ Prozent reduziert werden. Die Speicherung von Minima und Maxima zur Verringerung des Quantisierungsfehlers bewirkte insbesondere im Zusammenhang mit der Silhouetten-Kodierung eine deutliche Verbesserung der Ergebnisse, wohingegen beide Verfahren für sich betrachtet hinter den Erwartungen zurückblieben. Die Vorteile der Benutzung des YUV- an Stelle des RGB-Farbmodells sind aus dem Bereich der Bild- und Videokompression hinlänglich bekannt. Auch bei der Kompression von Lichtfeldern ermöglichte der Wechsel des Farbraums entsprechende Verbesserungen.

Kapitel 5

Benutzung von bildbasierten Szenenprimitiven in interaktiven Umgebungen

Bildbasierte Szenenobjekte können vielseitig und nützlich in interaktiven Umgebungen eingesetzt werden, um die Darstellung virtueller Welten zu verbessern. In diesem Kapitel soll an drei Beispielen gezeigt werden, wie bildbasierte Techniken zur Ergänzung und Verbesserung traditioneller Anwendungen der Computergrafik genutzt werden.

Im folgenden Abschnitt wird an Hand der Integration von Lichtfeldern in die *Inventor*-Klassenbibliothek demonstriert, wie durch bildbasierte Primitive die Ausdrucksfähigkeit klassisch modellierter Szenen vergrößert werden kann. In Abschnitt 5.2 werden bildbasierte Techniken zur Komplexitätsreduktion in einer VR-Anwendung genutzt. Durch den Einsatz einer auf geschichteten Tiefenbildern beruhenden Datenstruktur können einfache und komfortable Navigationswerkzeuge auch für hoch komplexe virtuelle Welten implementiert werden. Schließlich wird am Beispiel eines virtuellen Museums der Einsatz von Lichtfeldern zur Präsentation von Objekten im Internet vorgestellt.

5.1 Integration von Lichtfeldern in virtuelle Welten

Wie in Kapitel 2 deutlich wurde, haben bildbasierte und geometriebasierte Darstellungen jeweils spezifische Vor- und Nachteile. Dies und die Tatsache, dass eine große Zahl geometrischer Modelle bereits existiert und genutzt werden sollte, motiviert die Integration bildbasierter Primitive in traditionell geometriebasiert erstellte Szenen.

5.1.1 Open Inventor

Open Inventor [WERNECKE 1994] ist eine Bibliothek von Klassen und Methoden zur Ausgabe interaktiver dreidimensionaler Grafik. Open Inventor ist streng objektorientiert in der Programmiersprache C++ entwickelt worden, kann aber auch

in C-Programme eingebunden werden. Implementierungen von Open Inventor stehen für verschiedene Betriebssystemplattformen zur Verfügung [SGI Open Inventor, TGS Open Inventor, Apprentice Open Inventor Clone].

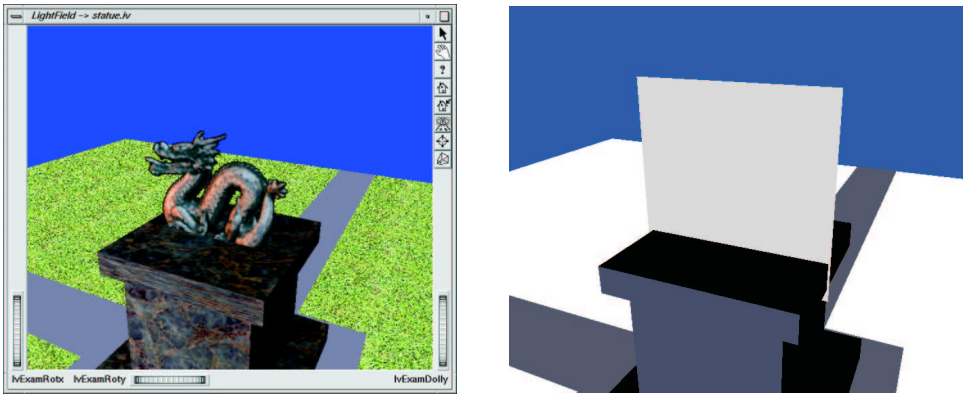
Open Inventor erlaubt die Entwicklung von Programmen mit mächtigen graphischen Fähigkeiten bei geringem Programmieraufwand. Basierend auf der OpenGL-Bibliothek [KEMPF et al. 1997] stellt Open Inventor ein Anzahl von Grafikprimitiven zur Modellierung, Manipulation und Darstellung von Szenen zur Verfügung. Jede Szene wird hierbei durch einen gerichteten azyklischen Graphen (DAG), dem *Inventor-Szenengraph*, repräsentiert.

Das objektorientierte Design von Open Inventor erlaubt die Veränderung und Erweiterung der Klassenbibliothek entsprechend den eigenen Bedürfnissen. Die Open Inventor-Bibliothek enthält neben reinen Grafikprimitiven

- Klassen zur interaktiven Manipulation von Szenenobjekten durch den Benutzer,
- Komponenten, welche die einfache Erstellung komplexer grafischer Benutzerschnittstellen erlauben und
- Funktionalität zum Lesen und Schreiben von Dateien im Open Inventor-Szenenbeschreibungs-Format.

Open Inventor ist unabhängig von der benutzten grafischen Benutzeroberfläche. Allerdings sind Klassen zur Einbindung von GUI-Komponenten im jeweiligen Betriebssystem nützlich.

5.1.2 Integration von Lichtfeldern



(a) (b)

Abbildung 5.1: (a) Einfache Testszene in einem Open Inventor-Viewer. Der Drache auf dem Podest ist ein Lichtfeld. Das Podest und dessen Umgebung ist herkömmlich mit texturierten Polygonen modelliert. (b) Die benutzte Szenengeometrie in einfacher Darstellung. Die Fläche, auf welche die Lichtfeldansicht als Textur abgebildet wird, ist gut zu erkennen.

Zur Integration von Lichtfeldern in eine Open Inventor-Anwendung wurden Teile der inReal-Klassenbibliothek (Abschnitt 4.4) verwendet. Abbildung 5.1 (a) zeigt eine einfache Viewer-Anwendung mit dem Drache-Lichtfeld auf einem Podest. In der dargestellten Testszene wurden das Podest und dessen Umgebung herkömmlich mit geometrischen Primitiven modelliert.

Abbildung 5.2: Buddha-Lichtfeld vor einer Rekonstruktion des Residenzschlosses Ludwigsburg.

Die interaktive Darstellung von Lichtfelder in Open Inventor erfolgt, ähnlich wie in inReal, durch die Berechnung von Texturen. Diese werden auf eine Fläche an der Position der Bildebene des Lichtfeldes aufgebracht (Abb. 5.1 (b)). Hierzu wurde eine Klasse `Lightfield` von `SoTexture2` abgeleitet. `SoTexture2` ist eine Klasse aus Open Inventor und repräsentiert eine zweidimensionale Textur. `Lightfield` überschreibt die `GLRender`-Methode von `SoTexture2` und implementiert so an Stelle einer konstanten Textur die Darstellung eines Lichtfeldes. Bei Aufruf der `GLRender`-Methode, einer Instanz von `Lightfield`, wird die Position des Betrachters bestimmt, eine entsprechende Lichtfeld-Ansicht berechnet und als Textur zurückgegeben. Die `Lightfield`-Klasse kann in jedem Inventor-Szenengraphen verwendet werden.

Zur Erstellung einer Open Inventor-Szene mit einem Lichtfeld wird die Geometrie der Szene zunächst herkömmlich modelliert. Da eine Szene im Open Inventor-Datenformat in einer einfachen Textdatei gespeichert wird, kann die Datei anschliessend leicht um die notwendigen Lichtfeld-Einträge erweitert werden. Hierzu wird für jedes Lichtfeld eine Fläche mit dem Lichtfeld als Textur definiert. Zur Erzeugung der benötigten Instanzen der `Lightfield`-Klasse im Szenengraphen während des Einlesens der Szene wird die entsprechende Komponente von Open Inventor angepasst. Auf diese Weise kann eine Inventor-Szenendatei mit Lichtfeldern problemlos eingelesen werden.

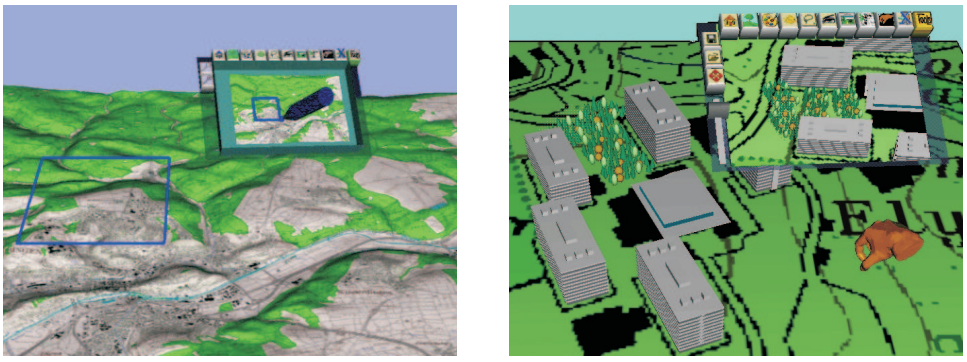
Das beschriebene Konzept ist für SGI IRIX implementiert. Die vorliegende Implementierung erlaubt die Darstellung und das Durchwandern der abgebildeten Szenen mit einer Bildwiederholrate von 6-10 Bildern pro Sekunde auf einer SGI Octane (2$\times$

MIPS R10000). Abbildung 5.2 zeigt ein weiteres Beispiel, das mit dem entwickelten Programm berechnet wurde.

5.2 Der *Multi-LDI*

Die Hauptziele auf dem Forschungsgebiet der virtuellen Realität (*Virtual Reality (VR)*) sind, einem Betrachter einen möglichst realistischen Eindruck einer künstlichen Welt zu vermitteln, ihn in diese zu integrieren und mit dieser interagieren zu lassen. Da sich der Betrachter, im Unterschied zum klassischen Anwendungsszenario, virtuell in eine Computer-generierte Szene hineinbegibt, müssen besondere Hilfsmittel gefunden werden, um dem Betrachter die Orientierung, Navigation und Bedienung der VR-Umgebung zu ermöglichen.

Zu diesen Hilfsmitteln gehören die aus herkömmlichen grafischen Benutzeroberflächen (GUI) bekannten Elemente wie Knöpfe, Schieber oder Pull-down-Menus, aber auch Techniken, die keine Entsprechung in GUIs oder der realen Welt besitzen. Ein Beispiel hierfür ist die *World in Miniature (WIM)* Technik [STOAKLEY et al. 1995], bei welcher die gesamte virtuelle Umgebung in verkleinerter Form auf eine dreidimensionale Karte abgebildet wird, die der Benutzer in der Hand hält (Abb. 5.3 (a)). Mit Hilfe der WIM kann der Benutzer seine Position bestimmen und verändern oder frei den Teil der Szene auswählen, der angezeigt werden soll.



(a) (b)

Abbildung 5.3: (a) Die World in Miniature (WIM) Technik. Eine verkleinerte Ansicht der gesamten virtuellen Umgebung wird auf einer dreidimensionalen Karte abgebildet, die der Benutzer in der Hand hält. (b) Das snapshot tool ermöglicht die gleichzeitige Betrachtung der Szene aus unterschiedlichen Perspektiven, die in einzelnen Fenstern dreidimensional abgebildet werden.

Ein anderes Verfahren zur Verbesserung der Orientierung in und der Manipulation von virtuellen Umgebungen ist das *snapshot tool* [SCHMALSTIEG et al. 1999], durch das eine oder mehrere zusätzliche Ansichten der Szene zur Verfügung gestellt werden. Frei im virtuellen Raum positionierbare Fenster zeigen die Szene aus unterschiedlichen, frei definierten Perspektiven. Dabei stellt jedes Fenster, im Gegensatz zu statischen Bildern, die Szene dreidimensional und perspektivisch korrekt dar (Abb. 5.3 (b)). Das

snapshot tool ist besonders nützlich, um Szenenbereiche, die für den Betrachter nicht einsehbar sind, darzustellen oder um Veränderungen der Szene über die Zeit zu visualisieren.

Zur Darstellung der künstlichen Welten in VR-Anwendungen werden, wie auf allen anderen Anwendungsgebieten der Computergrafik auch, immer größere und detailliertere Modelle verwendet. Dies und die Tatsache, dass gerade VR-Anwendungen ein hohes Maß an Interaktivität erfordern, motiviert den Einsatz bildbasierter Verfahren. Die beiden oben beschriebenen Navigationshilfen erfordern bei ihrer Darstellung jeweils die Abbildung der gesamten Szene. Beim WIM muss die gesamte Szene entsprechend verkleinert und beim snapshot tool aus einer speziellen Perspektive gezeigt werden. Bei naiver Implementierung stiege deswegen die Darstellungszeit einer VR-Anwendung linear mit der Zahl der existierenden Instanzen von WIM bzw. snapshot tool an. Dies würde die interaktive Darstellung komplexer Szenen unmöglich machen.

Der Einsatz einer bildbasierten Repräsentation, wie der *Multi-LDI-Datenstruktur* [STOEV et al. 2000, STOEV et al. 2001], erlaubt die simultane Verwendung zahlreicher Instanzen von VR-Navigationshilfen wie WIM oder snapshot tool. Hierzu kombiniert der Multi-LDI eine Anzahl von geschichteten Tiefenbildern (LDI). Deren Darstellungsaufwand hängt stark von der Auflösung und nur schwach von der Komplexität der repräsentierten Objekte ab. Weiterhin erlaubt der im folgenden Abschnitt beschriebene Algorithmus die schnelle Berechnung von Ansichten eines LDI. Diese Kombination von Eigenschaften ermöglicht die interaktive Darstellung auch sehr komplexer Szenen mit Hilfe von LDIs.

5.2.1 Geschichtete Tiefenbilder (LDIs)

Wie in Abschnitt 2.4.5 erklärt, kombinieren geschichtete Tiefenbilder (Layered Depth Images (LDI)) [SHADE et al. 1998] eine Anzahl von Tiefenbildern, die von unterschiedlichen Positionen aufgenommen wurden, in eine einzige Datenstruktur. Hierzu können LDIs für jeden Pixel eine beliebige Anzahl von Farbwerten mit zugehörigem Tiefenwert abspeichern. Alle Werte sind hierbei in ein einheitliches Bezugssystem eingeordnet, welches durch das Projektionszentrum des Tiefenbildes definiert ist. In einem LDI können auf diese Weise auch verdeckte Szenenteile gespeichert und so das Auftreten von „Löchern“ bei der Darstellung vermieden werden.

Zur Darstellung wird der LDI entlang der *epipolaren Geraden* in höchstens vier Teile zerlegt. Die epipolare Gerade verläuft durch das Projektionszentrum des LDI und die Betrachterposition $\mathbf{e}$ und schneidet die Bildebene des LDI im *epipolaren Punkt* (Abb. 5.4). Jeder der vier Teile des LDI kann nun einzeln dargestellt werden. Um die gegenseitige Verdeckung der Tiefenpixel korrekt zu rekonstruieren, ist es ausreichend, einen LDI in einer speziellen Reihenfolge zu durchlaufen. Auf diese Weise kann auf einen expliziten Tiefentest und damit auf die Berechnung der Abstände der Tiefenpixel vom jeweiligen Betrachter verzichtet werden [MCMILLAN 1995]. Da alle Pixel im LDI unabhängig von deren Verdeckung gezeichnet werden, ist es günstig, dass die Pixel entlang der Struktur des LDI ausgerichtet sind. Hierdurch kann die Position jedes projizierten Tiefenpixels auf der Bildebene des Zielbildes sehr schnell inkrementell berechnet werden. Die aufwendigste Operation bei der Darstellung eines LDI ist die

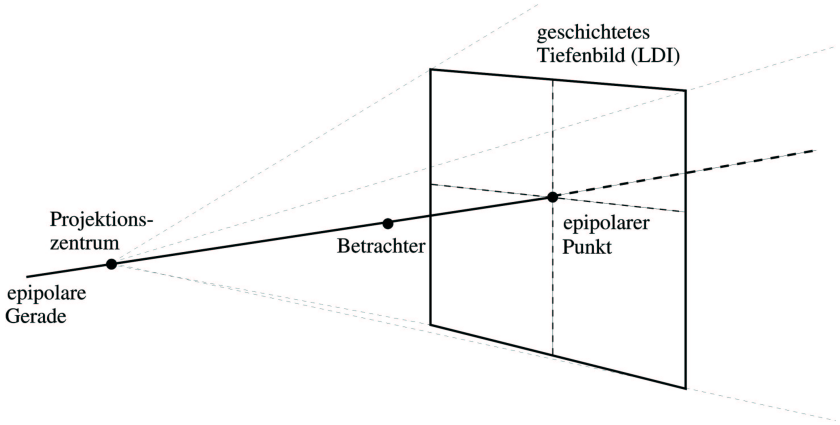


Abbildung 5.4: Aufteilung eines geschichteten Tiefenbildes (LDI) in vier Teile entlang der epipolaren Geraden. Jeder der vier Teile wird einzeln in einer speziellen Reihenfolge durchlaufen, um einen expliziten Tiefentest zu vermeiden.

Projektion der berechneten homogenen Koordinaten nach $\mathbb{R}^3$. Hierbei wird eine Division benötigt [GORTLER et al. 1997].

Weiterführende Arbeiten

Geschichtete Tiefenbilder können zur Komplexitätsreduktion bei großen geometrischen Modellen, beispielsweise Architekturmodellen, eingesetzt werden [POPESCU et al. 1998, RAFFERTY et al. 1998b]. Weiter entfernte Teile des Modells werden hierbei durch LDIs ersetzt. Allerdings ist der Speicherplatzbedarf dieser Technik bei großen und ausgedehnten Szenen, wie beispielsweise Industrieanlagen, sehr hoch, da unter Umständen eine große Anzahl von LDIs vorab erzeugt und gespeichert werden muss [ALIAGA und LASTRA 1999]. Zur Darstellung isolierter Objekte beschreibt Oliveira et al. in [OLIVEIRA und BISHOP 1999b] eine Datenstruktur, die eine Kombination von sechs LDIs enthält. Hierbei sind die LDIs in Form eines Würfels zueinander angeordnet. Je ein LDI speichert die Ansicht des Objekts aus Richtung der Normalen der jeweiligen Würfelseite.

5.2.2 Die *Multi-LDI*-Datenstruktur

Navigationshilfen für VR-Umgebungen erfordern die interaktive Darstellung virtueller Umgebungen aus unterschiedlichen Blickwinkeln. Durch die Benutzung einer bildbasierte Darstellung an Stelle von Geometrie können WIM und snapshot tool auch in komplexen Szenen verwendet werden. Die Auswahl des LDI als Basis für eine neue bildbasierte Datenstruktur bietet sich an, weil

- LDIs mit einem einfachen und effizienten Verfahren am Bildschirm dargestellt werden können (Abschnitt 5.2.1),

- sich der Abstand des Betrachters zu WIM oder snapshot tool während der gesamten Anwendung nur wenig ändert und

- eine dynamische Erzeugung von LDIs zur Laufzeit möglich ist.

Wie oben beschrieben, steigt der Aufwand bei der Darstellung eines LDI linear mit der Anzahl der gespeicherten Pixel, da alle Pixel unabhängig von ihrer Verdeckung immer abgebildet werden. Bei WIMs oder dem snapshot tool muss die korrekte Darstellung der Szene für einen großen Sichtwinkel gewährleistet sein. Da ein einzelner LDI, der für einen weiten Sichtwinkel „gültig“ ist, zwangsläufig einen großen Anteil von verdeckten Pixeln speichern muss, vereinigt der Multi-LDI eine Anzahl einzelner LDIs in einer gemeinsamen Datenstruktur. Hierzu wird die Hemisphäre über dem Fenster, durch welches die darzustellende Szene betrachtet wird, unterteilt (Abb. 5.5). Jedem Teilbereich der Hemisphäre wird ein LDI zugeordnet und zur Darstellung verwendet, wenn der Blickwinkel des Betrachters innerhalb des jeweiligen Hemisphärenabschnitts liegt. Der Sichtwinkel, für den jedes LDI ein korrektes Bild der Szene zeigen muss, ist im Vergleich zu [OLIVEIRA und BISHOP 1999b] klein. Dies hat den Vorteil, dass der Anteil der gezeichneten, jedoch verdeckten LDI-Pixel gering ist.

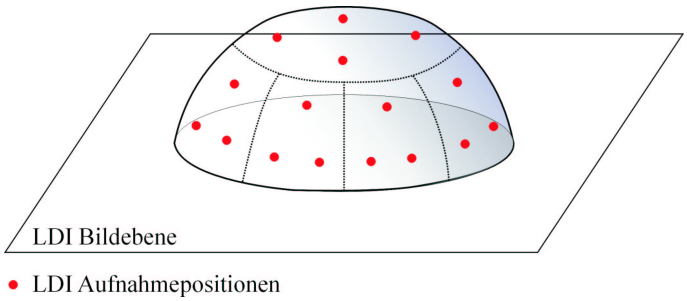


Abbildung 5.5: Unterteilung der Hemisphäre durch den Multi-LDI. Die markierten Positionen auf der Hemisphäre sind die Aufnahmepositionen für die Tiefenbilder, die zur Konstruktion des jeweiligen LDI benötigt werden.

Zur Konstruktion des Multi-LDI wird eine Anzahl von Tiefenbildern berechnet und deren Pixel in die jeweiligen LDIs eingetragen. In Abbildung 5.5 sind die Positionen, von denen die Tiefenbilder erzeugt werden, markiert. Die Aufnahmepositionen sind so gewählt, dass eine möglichst gleichmäßige Abdeckung aller Blickwinkel gewährleistet ist. Um Fehler in Bezug auf die Verdeckung in der Szene zu vermeiden, werden bei der Konstruktion jedes LDIs auch die Tiefenbilder, die im direkt benachbarten Teilbereich aufgenommen wurden, jeweils mit einbezogen.

5.2.3 Implementierung

Der WIM, das snapshot-tool sowie der Multi-LDI wurden auf Basis der Studierstube-Klassenbibliothek implementiert.

Die *Studierstube*-Klassenbibliothek

Die objektorientierte *Studierstube*-Klassenbibliothek [SCHMALSTIEG et al. 1996] basiert auf der in Abschnitt 5.1.1 beschriebenen Open Inventor-Klassenbibliothek und erweitert diese um Funktionalität, welche in dreidimensionalen immersiven VR-Umgebungen benötigt wird. Die Studierstube erlaubt die Erfassung und Verarbeitung von Signalen, die von Geräten zur Bestimmung der dreidimensionalen Position im Raum erzeugt werden. Weiterhin wird die Konstruktion von Benutzerschnittstellen für VR-Anwendungen mit den üblichen Elementen wie Knöpfen, Schiebern, Menus usw. unterstützt.

Integration in Open Inventor

Ähnlich wie bei der in Abschnitt 5.1 beschriebenen Arbeit wurden Multi-LDIs durch Ableitung einer geeigneten Klasse aus Open Inventor in eine VR-Anwendung integriert.

Die inReal-Klassenbibliothek (Abschnitt 4.4.2) stellt verschiedene Methoden zur Darstellung von LDIs zur Verfügung. Die Verfahren nutzen dabei teilweise spezielle Erweiterungen der OpenGL-Grafikbibliothek [KILGARD], um die Abbildung der LDI-Pixel mit Hilfe der Grafikhardware zu beschleunigen. Zur Darstellung des Multi-LDI am Bildschirm wurde ein spezieller *LDI splatter* entwickelt, der alle notwendigen Berechnungen nur in Software durchführt. Hierdurch werden alle Ressourcen des Rechners, auf dem die VR-Anwendung läuft, ausgenutzt. Denn während die virtuelle Welt mit Hilfe herkömmlicher Grafikhardware dargestellt wird, nutzt der LDI Splatter die Rechenleistung der CPU.

Zur Darstellung eines Multi-LDI wird eine Textur berechnet, welche in dem Fenster dargestellt wird, das dem WIM oder dem snapshot tool zugeordnet ist. Hierzu wird zunächst der Winkel bestimmt, aus welchem der Betrachter das Fenster sieht und der entsprechende LDI aus dem Multi-LDI zur Darstellung ausgewählt. Aus der aktuellen Betrachterposition und der Position des Fensters wird eine Matrix berechnet, mit welcher die LDI-Pixel auf die Textur projiziert werden. Dies geschieht vollständig in Software. Durch Ausnutzung der oben beschriebenen Reihenfolge bei der Projektion kann auf einen expliziten Tiefentest verzichtet werden. Anders als in [SHADE et al. 1998], erhöht eine individuelle und damit genaue Berechnung der Größe der projizierten Pixel die Bildqualität.

5.2.4 Dynamische Akquisition und Modifikation

Wie oben beschrieben, besteht ein Multi-LDI aus einer Anzahl einzelner LDIs. Zur dynamischen Konstruktion eines LDI wird eine Anzahl von Tiefenbildern der Szene berechnet. Durch Auslesen des z-Buffers der Grafikhardware wird der Abstand jedes Bildpixels zum Betrachter bestimmt. Die hieraus berechenbare Position im Raum wird danach genutzt, um die Pixel in die LDIs einzufügen. Pixel, die sich innerhalb einer ϵ-Umgebung an der gleichen Raumposition befinden, werden nur einmal gespeichert. Der Vorteil der Konstruktion von LDIs aus Tiefenbildern besteht darin, dass nur die

wirklich sichtbaren Szenenanteile gespeichert werden. Auf diese Weise wird die Anzahl der Pixel in jedem LDI und damit die Darstellungszeit reduziert. Verfahren, die beispielsweise ray-casting benutzen [PFISTER et al. 2000], um die LDI-Pixel zu bestimmen, speichern unter Umständen einen hohen Anteil an nicht benötigten Daten.

Die Berechnung der Tiefenbilder, die zur Konstruktion eines einzelnes Multi-LDI benötigt werden, kann ein zeitaufwendiger Vorgang sein. Aus diesem Grund wird bei der dynamischen Akquisition von Multi-LDIs zunächst nur der LDI berechnet, welcher der Blickrichtung des Betrachters entspricht. Weitere LDIs können danach in einem Hintergrundprozeß konstruiert werden.

Zur dynamischen Veränderung von Szenen im Multi-LDI kann klassische geometriebasierte Computergrafik mit dem Multi-LDI kombiniert werden. Für jeden LDI-Pixel wird ein Tiefenwert berechnet und in den z-Buffer der Grafikhardware eingetragen. Auf diese Weise ist es möglich, die Verdeckung korrekt zu rekonstruieren, wenn herkömmliche geometrische Objekte in den Multi-LDI integriert werden.

5.2.5 Ergebnisse

Die in den Abbildungen 5.6 und 5.7 dargestellten Ergebnisse zeigen die Vorteile des Einsatzes bildbasierter Verfahren bei der Darstellung komplexer virtueller Umgebungen. Alle Werte wurden mit einer prototypischen, nicht optimierten Implementierung des Multi-LDI gemessen.

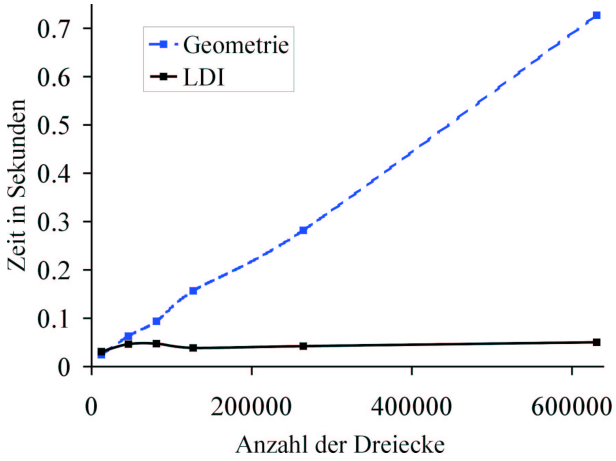


Abbildung 5.6: Darstellungszeit in Abhängigkeit von der Anzahl der Dreiecke. Die Rechenzeit steigt beim klassischen Verfahren linear an. Dagegen bleibt die zur Abbildung eines Multi-LDI benötigte Zeit annähernd konstant.

In Abbildung 5.6 wird die Darstellungszeit für die klassische und die bildbasierte Methode verglichen und ist in Abhängigkeit von der Anzahl der Dreiecke aufgetragen. Die Rechenzeit wächst bei klassischer Darstellung linear mit der Szenengröße an, wohingegen die Zeit, die für die Abbildung des Multi-LDI auf dem Bildschirm benötigt wird, fast konstant bleibt. Dies kann damit erklärt werden, dass die Menge der Pixel

im Multi-LDI nicht weiter ansteigt, wenn das gesamte Sichtfeld eines LDI mit Szenenelementen bedeckt ist. Durch die oben beschriebene Konstruktion des Multi-LDI aus Tiefenbildern ist die Größe der LDIs und damit der Aufwand zur Darstellung nach oben beschränkt.

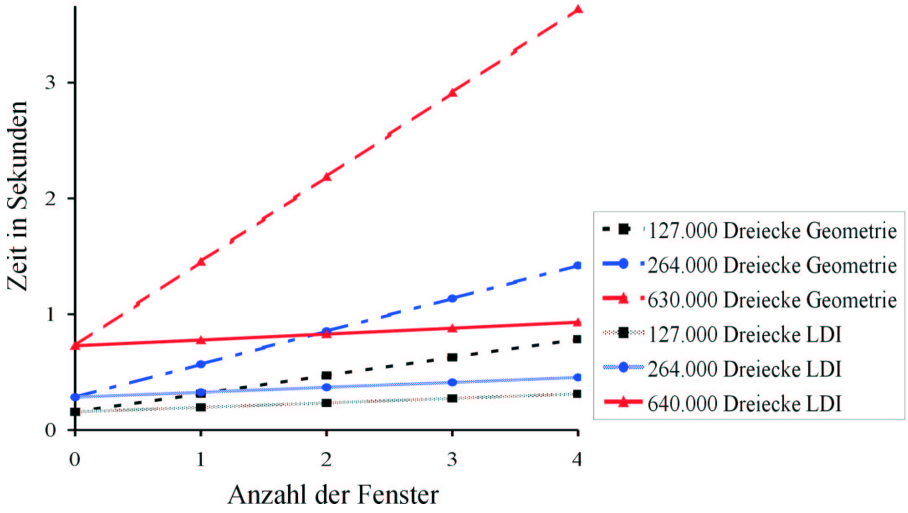


Abbildung 5.7: Gegenüberstellung der benötigten Rechenzeit für die bildbasierte, sowie die klassische Darstellung unterschiedlich großer Szenen. Die Steigerung der Rechenzeit bei bildbasierter Darstellung ist sehr gering, verglichen mit der klassischen Methode.

Wird die Anzahl der Instanzen von WIM oder snapshot tool erhöht, wächst die Zeit, die zu deren Darstellung benötigt wird. Allerdings schlägt sich die Benutzung des Multi-LDI in einer nur schwachen Steigerung der Rechenzeit nieder, wogegen der Aufwand zur Darstellung der Szene mit klassischen Mitteln signifikant ansteigt (Abb. 5.7). Zum Test wurden Szenen mit einer unterschiedlichen Zahl von Dreiecken benutzt. Am Deutlichsten werden die Vorteile des Multi-LDI bei einer Testszene mit etwa 630.000 Dreiecken.

5.3 Lichtfeld-Darstellung im Internet

Lichtfelder ermöglichen die Speicherung, Übermittlung und Darstellung von Objekten in sehr allgemeiner Form. Diese Unabhängigkeit der Beschreibung von konkreten Eigenschaften des Objekts ist für eine Übertragung und Präsentation im Internet günstig. Aus diesem Grund wird in diesem Abschnitt eine angepasste Implementierung eines *LightField-Viewers* [TERFLOTH 2001] zur Einbettung von Lichtfeldern in Webseiten vorgestellt.

Der LightField-Viewer ermöglicht den Empfang spezieller Dateien mit Lichtfeld-Daten und die interaktive Darstellung der Objekte mit geringem Rechenaufwand.

Mögliche Einsatzbereiche für Lichtfelder sind Produktpräsentationen im eCommerce, virtuelle Museen oder Bildungsangebote. Allgemein können alle Anwendungen von der Lichtfeld-Darstellung im Internet profitieren, welche dem Benutzer die interaktive Betrachtung beliebiger Objekte zu Studien- oder Informationszwecken ermöglichen wollen. Für Museen ist diese Form der Präsentation besonders attraktiv, da auf diese Weise die Exponate vom Benutzer wesentlich genauer betrachtet werden können, als dies sonst aus Sicherheitsgründen möglich wäre.

5.3.1 Implementierung

Java 3D

Der LightField-Viewer wurde in Java entwickelt. Zur Implementierung der grafischen Ausgabe wurde *Java3D (J3D)* [Java3D] benutzt. Als Bestandteil der *Java Media Family* erweitert Java3D die *Java 2 Platform* [Java 2 Platform] von Sun um eine Sammlung von Klassen und Methoden zur Repräsentation, Darstellung und Manipulation von dreidimensionalen virtuellen Welten und unterstützt so die Entwicklung komplexer Grafikanwendungen auf hohem Niveau. Die mit J3D entwickelten Applets können sowohl eigenständig, als auch in Webseiten integriert ausgeführt werden.

Java3D zeichnet sich durch folgende Eigenschaften aus:

- **Performance.** Durch eine geschichtete Architektur, welche auf einfachen Grafik-APIs wie OpenGl [KEMPF et al. 1997] oder DirectX [DirectX] aufsetzt, wird eine schnelle Ausführung der Programme gewährleistet.
- **Flexible Modellierung.** Java3D unterstützt die Modellierung von Szenen mit einer Vielzahl von Grundelementen, wie klassischen Grafikprimitiven, aber auch animierten Komponenten oder Soundobjekten.
- **Objektorientierung.** Durch Kapselung der Details der Grafikprogrammierung muss der Entwickler kein Spezialwissen über die beste Umsetzung seiner Ideen in Grafikhardware besitzen. Weiterhin kann J3D durch das hohe Abstraktionsniveau Optimierungen, wie eine Verbesserung der Szenenstruktur, vornehmen, ohne dass dies vom Entwickler bemerkt wird. Eine Erweiterung von J3D durch Ableitung neuer von existierenden Klassen ist jederzeit möglich.

Ähnlich wie Open-Inventor (Abschnitt 5.1.1) organisiert J3D die Szenenbeschreibung in einem Graphen. Eine Besonderheit von Java3D ist dabei die Verwendung von `Locale`-Objekten zur Modellierung von räumlich sehr ausgedehnte Szenen. Die direkt unter der Szenenwurzel platzierten `Locale`-Objekte definieren ein lokales Koordinatensystem, damit auch weit auseinander liegende Objekte mit der erforderlichen nummerischen Genauigkeit repräsentiert werden können. Instanzen der `ViewPlattform`-Klasse im Szenengraphen repräsentieren die Sicht eines oder mehrerer Betrachter auf die modellierte Szene. Dies erlaubt die Betrachtung und Manipulation der Szene durch mehrere Benutzer gleichzeitig. Hierzu kann die Szene auf einer Vielzahl unterschiedlicher Typen von Ausgabegeräten dargestellt werden.

Integration von Lichtfeldern

Das Zusammenwirken der zur Darstellung einer Lichtfeld-Ansicht im LightField-Viewer benutzten Klassen ist in Abbildung 5.8 schematisch dargestellt. Beim Aufbau des Szenengraphen wurde ein ähnlicher Ansatz wie bei der Integration von Lichtfeldern in Open-Inventor (Abschnitt 5.1.2) gewählt.

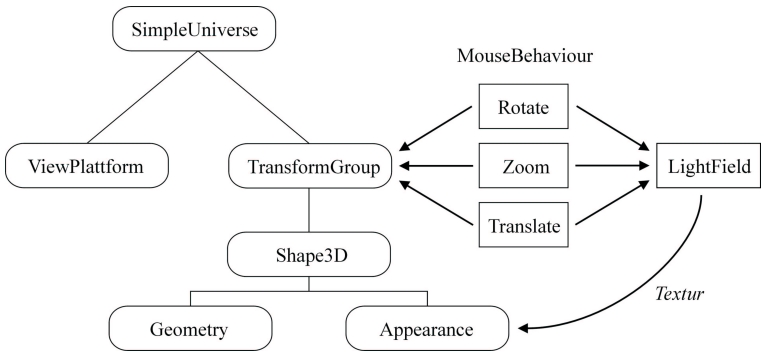


Abbildung 5.8: Zusammenwirken der verschiedenen Klassen bei der Darstellung vom Lichtfeldern im LightField-Viewer.

Der benutzte Szenengraph besitzt nur wenige Knoten, welche auf Abbildung 5.8 mit abgerundeten Ecken dargestellt sind. Zur Abbildung eines Lichtfeldes wird eine Textur auf eine Ebene (`Shape3D`) an der Position der Bildebene des Lichtfeldes aufgebracht. Die `LightField`-Klasse, welche die Lichtfeld-Daten speichert, berechnet die Textur entsprechend der relativen Position des Betrachters zur Bildebene. Ein Cache, ähnlich zu dem in Abschnitt 4.3.11 beschriebenen bildbasierten Cache, beschleunigt dabei die Berechnung, sodass die interaktive Betrachtung von Lichtfeldern möglich ist. Zur Veränderung der Position des Lichtfeldes werden die Benutzereingaben mit verschiedenen Instanzen von `MouseBehaviour`-Klassen überwacht und interpretiert. Bewegt der Nutzer die Maus, geben die `MouseBehaviour`-Instanzen entsprechende Signale an die `LightField`-Klasse und die `TransformGroup` weiter, welche die Position der Bildebene des Lichtfeldes kontrolliert. Die `LightField`-Klasse berechnet dann eine neue Textur, die an den `Appearance`-Knoten im Szenengraphen zur Darstellung übergeben wird.

Bei der Implementierung des LightField-Viewers konnten zahlreiche Konzepte aus der inReal-Klassenbibliothek (Abschnitt 4.4.2) benutzt werden. Da die Programmiersprachen Java und C++ eng miteinander verwandt sind, war die Portierung des entsprechenden Programmcodes nicht schwierig.

5.3.2 Resultate und Beispiele

Die vorliegende Implementierung erlaubt die interaktive Darstellung von Lichtfeldern, die mit Vektor-Quantisierung komprimiert wurden. Das entwickelte Applet kann sowohl als eigenständiges Programm, als auch in Webseiten eingebettet, ausgeführt werden. Leider ist es nicht möglich, Argumente an das Applet zu übergeben, wenn zur

Anzeige der entsprechenden Webseite der Microsoft Internet Explorer benutzt wird. Aus diesem Grund muss die Liste der darstellbaren Lichtfelder statisch kodiert werden. Diese Schwierigkeit und die sehr komplizierte Einbindung des Applets in die Webseite ist vielleicht ein Indiz für das bekanntermaßen aggressive Verhalten von Microsoft gegenüber Mitbewerbern.

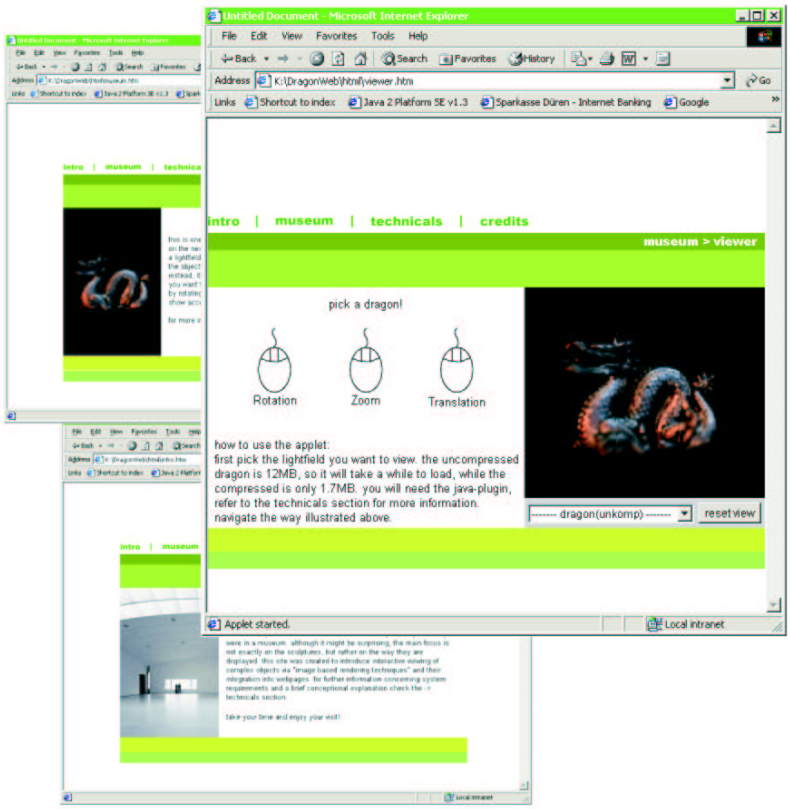


Abbildung 5.9: Einige der Seiten des virtuellen Museums. Im virtuellen Ausstellungsraum kann ein Lichtfeld interaktiv betrachtet werden.

Abbildung 5.9 zeigt einige der Webseiten eines virtuellen Museums, das als Beispiel für die Nutzung von Lichtfeldern im Internet erstellt wurde. Über eine Navigationsleiste im oberen Teil jeder Seite kann der Benutzer aus den unterschiedlichen Bereichen des Angebots wählen. Wenn der Benutzer den virtuellen Ausstellungsraum betritt, wird zunächst der LightField-Viewer geladen. Danach kann das darzustellende Lichtfeld aus einem Pull-Down Menu ausgewählt werden. Hilfsinformationen leiten den Benutzer bei der Bedienung des LightField-Viewers an.

Kapitel 6

Diskussion und Ausblick

Die bildbasierte Modellierung und Darstellung (IBR) von Szenen besitzt im Vergleich zur klassischen geometriebasierten Computergrafik zahlreiche Vorteile. In dieser Arbeit wurde deshalb die Kompression und Integration bildbasierter Primitive in interaktive Umgebungen beschrieben. Dabei behandelte der Hauptteil der Arbeit den Aufbau und die Funktionsweise eines Kompressionsverfahrens für Lichtfelder. Die erzielten Ergebnisse sollen jetzt kurz zusammengefasst und kritisch betrachtet werden. Abschließend gibt dieses letzte Kapitel einen Ausblick auf zukünftige Entwicklungen und interessante Forschungsthemen auf dem Gebiet IBR.

6.1 Zusammenfassung

Die in dieser Arbeit beschriebene Waveletstream-Datenstruktur ermöglicht die progressive Speicherung, Übertragung und Darstellung von komprimierten Lichtfeld-Datensätzen. Hierzu wird eine Wavelet-Dekomposition der Daten berechnet, mit welcher das Lichtfeld in unterschiedlichen Auflösungsstufen dargestellt werden kann. Die Wavelet-Koeffizienten werden in einem Datenstrom in einer Reihenfolge übertragen, welche die sukzessive Verfeinerung der Darstellung mit fortschreitendem Einlesen der Daten ermöglicht. Der Waveletstream erlaubt die Kompression von Lichtfeldern in einem Verhältnis von $1:100$. Nur $0,5$ Prozent der Koeffizienten sind ausreichend, um eine beliebige Lichtfeld-Ansicht mit guter Qualität zu rekonstruieren. Dabei wird die interaktive Dekompression und Darstellung der Lichtfelder durch den Einsatz spezieller Caching-Verfahren unterstützt. Zur Unterscheidung von Vorder- und Hintergrund wird die Objektsilhouette im Waveletstream gespeichert. Hierdurch wird nicht nur die Integration von bildbasierten Objekten in herkömmlich modellierte Szenen unterstützt, sondern auch die Kompression erhöht. Die Waveletstream-Datenstruktur erlaubt neben hohen Kompressionsraten als einzige der bislang veröffentlichten Ansätze die interaktive Dekompression und Darstellung von Lichtfeldern auf unterschiedlichen Skalen. Weiterhin zeichnet den Waveletstream die Möglichkeit aus, beliebige zusätzliche Informationen und die Objektsilhouette zu speichern.

Bildbasierte Datenstrukturen können in zahlreichen Anwendungen sinnvoll eingesetzt werden. In dieser Arbeit wurden als Beispiele die Übertragung und Darstellung von

Lichtfeldern im Internet, die Integration von Lichtfeldern in eine Programmierumgebung für klassische Computergrafik und die Nutzung einer bildbasierten Datenstruktur in virtueller Realität vorgestellt. Mit letzterer Anwendung wurde demonstriert, wie erst bildbasierte Techniken die Realisation eines leistungsfähigen Werkzeugs zur Navigation in künstlichen Welten ermöglichen.

6.2 Diskussion

In Abschnitt 4.2 wurden verschiedene Anforderungen an ein Kompressionsverfahren für Lichtfelder formuliert. An Hand dieser sollen die erreichten Ergebnisse kurz diskutiert werden.

- **Wahlfreier und gleich schneller Zugriff** auf alle Werte im komprimierten Lichtfeld. Die Wavelet-Koeffizienten aus dem Koeffizientenbaum werden Niveau für Niveau im Waveletstream gespeichert. Mit Hilfe zusätzlicher Navigationsinformation, die während des Ladens in den Waveletstream eingefügt wird, können beliebige Lichtfeldwerte im Datensatz gefunden und rekonstruiert werden. Der Aufwand für die Traversierung des Koeffizientenbaums entlang beliebiger Pfade und damit für die Rekonstruktion einzelner Lichtfeldwerte hängt nur von der Länge der Pfade ab.

- **Einsatz in interaktiven Anwendungen.** Verschiedene Caching-Verfahren beschleunigen die Dekompression von Lichtfeldwerten, sodass ein Einsatz des Waveletstreams in interaktiven Anwendungen problemlos erfolgen kann. Die in Kapitel 5 vorgestellten Anwendungsbeispiele zeigen, dass bildbasierte Datenstrukturen und insbesondere Lichtfelder erfolgreich in interaktive Grafikanwendungen integriert werden können und eine Bereicherung für diese darstellen.

- **Unterstützung der Integration in virtuelle Umgebungen.** Durch die Kodierung der Objektsilhouette im Waveletstream wird nicht nur die Kompression verbessert. Mit dieser Information kann auch zwischen Objekt und Hintergrund unterschieden werden. Hierdurch können Lichtfelder leicht mit klassischer Computergrafik kombiniert werden (Kapitel 5).

- **Multi-Skalen-Darstellung.** Mit den Koeffizienten einer Wavelet-Dekomposition können unterschiedlich aufgelöste Darstellungen einer Funktion auf einfache Weise berechnet werden. In Kapitel 4 wurde gezeigt, wie hierdurch die Bildqualität eines Lichtfeldes bei Betrachtung aus unterschiedlichen Abständen verbessert wird, da störende Artefakte durch die Verwendung einer entsprechend gefilterten Version des Lichtfeldes vermieden werden.

Die in dieser Arbeit vorgestellte Waveletstream-Datenstruktur wurde mit dem Ziel entworfen, Lichtfelder einerseits möglichst stark zu komprimieren, andererseits die Rekonstruktion beliebiger Lichtfeld-Ansichten aus den komprimierten Daten zu ermöglichen. Mit dem Waveletstream ist ein guter Kompromiss zwischen beiden Anforderungen gelungen. Die Kompressionsraten könnten durch den Einsatz von Wavelets höherer Ordnung verbessert werden. In Abschnitt 4.3.13 werden die hierzu notwendigen

Änderungen beschrieben. Diese haben allerdings einen erhöhten Rechenaufwand zur Folge, sodass nicht feststeht, ob die interaktive Dekompression und Darstellung eines auf diese Weise komprimierten Lichtfeldes möglich ist.

6.3 Ausblick

Obwohl schon zahlreiche grundlegende Arbeiten zur bildbasierten Modellierung und Darstellung erschienen sind, bleiben noch zahlreiche offene Fragen und neue Aufgabenstellungen.

Die Abbildung von Objekten mittels Lichtfeldern besitzt viele einzigartige Vorteile. Dennoch kann eine in Bezug auf Speicherplatzbedarf, Berechnungsaufwand und andere Kriterien ideale Szenenbeschreibung nur aus einer Kombination unterschiedlicher Techniken bestehen. Denn jedes Verfahren hat spezifische Vor- und Nachteile, die, geschickt ausgenutzt, eine kompakte und mit geringem Aufwand darstellbare Szenenrepräsentation ermöglichen. Leider sind bislang kaum Arbeiten erschienen, die Verfahren zur Gewinnung einer solchen Szenenrepräsentation beschreiben. Es müssen Kriterien oder Heuristiken gefunden werden, welche die Auswahl der richtigen Datenstrukturen in der jeweiligen Situation erlauben. Eine interessante Aufgabenstellung wäre auch der Entwurf von Datenstrukturen, die als gemeinsame Basis für unterschiedliche Objektrepräsentationen dienen und die (dynamische) Erzeugung von diesen ermöglichen. Der in Kapitel 2 vorgestellte randomisierte z-Buffer ist ein erster Schritt in diese Richtung.

Wie in Abschnitt 2.5 ausgeführt, ist die plenoptische Funktion ein Konzept, mit welchem die Beleuchtung einer Szene zu einem beliebigen Zeitpunkt vollständig beschrieben wird. Da die in der Computergrafik dargestellten virtuellen Welten, wie die wirkliche Welt, nicht statisch sondern dynamisch veränderbar sind, muss die plenoptische Funktion durch ein flexibleres Konzept ersetzt werden. Erste Ansätze wurden bereits veröffentlicht. Idealerweise stünde allerdings eine Datenstruktur zur Verfügung, die einerseits, wie ein Lichtfeld, allgemein und unabhängig von den Eigenschaften der Szene wäre, andererseits dynamische Veränderungen der Geometrie und der Beleuchtung einer Szene zuließe.

Klassische Bildverarbeitung und bildbasierte Darstellung müssen in Zukunft in neuen, sinnvollen Anwendungen zusammengeführt werden, die sich auch in der Praxis bewähren. Verfahren zur automatischen Akquisition, interaktiven Bearbeitung, Speicherung und Übertragung von beliebigen Objekten wären auch für kommerzielle Anwender in vielen Bereichen von großem Nutzen.

Literaturverzeichnis

[ADELSON und BERGEN 1991] ADELSON, E. H. und J. R. BERGEN (1991). *The Plenoptic Function and the Elements of Early Vision*. Computational Models of Visual Processing.

[ALIAGA und CARLBOM 2001] ALIAGA, DANIEL und I. CARLBOM (2001). *Plenoptic Stiching: A Scalable Method for Reconstructing 3D Interactive Walkthroughs*. In: FIUME, EUGENE, Hrsg.: *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 443–450. Addison Wesley Longman.

[ALIAGA und LASTRA 1999] ALIAGA, DANIEL G. und A. LASTRA (1999). *Automatic Image Placement to Provide a Guaranteed Frame Rate*. In: ROCKWOOD, ALYN, Hrsg.: *SIGGRAPH 99, Computer Graphics Proceedings*, Annual Conference Series, S. 307–316. Addison Wesley Longman.

[ALIAGA und LASTRA 1997] ALIAGA, DANIEL G. und A. A. LASTRA (1997). *Architectural Walkthroughs Using Portal Textures*. In: YAGEL, RONI und H. HAGEN, Hrsg.: *Proceedings Visualization '97*, S. 355–362.

[ALPERT et al. 1993] ALPERT, B., G. BEYLKIN, R. COIFMAN und V. ROKHLIN (1993). *Wavelet-like bases for the fast solution of second-kind integral equations*. SIAM Journal on Scientific Computing, 14(1):159–184.

[ALPERT 1993] ALPERT, BRADLEY K. (1993). *A Class of Bases in L^2 for the Sparse Representation of Integral Operators*. SIAM Journal on Mathematical Analysis, 24(1):246–262.

[Apprentice Open Inventor Clone] APPRENTICE OPEN INVENTOR CLONE. `http:www.mrpowers.com/Apprentice`.

[BARTZ et al. 1999] BARTZ, DIRK, M. MEISSNER und T. HÜTTNER (1999). *OpenGL-assisted occlusion culling for large polygonal models*. Computers and Graphics, 23(5):667–679.

[BLINN 1977] BLINN, J. F. (1977). *Models of light reflection for computer graphics*. Computer Graphics, 11(2):192–198.

[BLINN 1978] BLINN, J. F. (1978). *Simulation of wrinkled surfaces*. Computer Graphics (SIGGRAPH '78 Proceedings), 12(3):286–292.

[BLINN und NEWELL 1976] BLINN, JAMES F. und M. E. NEWELL (1976). *Texture and Reflection in Computer Generated Images*. Communications of the ACM, 19(10):542–547.

[BOIVIN und GAGALOWICZ 2001] BOIVIN, SAMUEL und A. GAGALOWICZ (2001). *Image-Based Rendering of Diffuse, Specular and Glossy Surfaces from a Single Image*. In: FIUME, EUGENE, Hrsg.: *SIGGRAPH 2001, Computer Graphics Proceedings*, S. 107–116. Addison Wesley Longman.

[BOOCH et al. 1999] BOOCH, GRADY, J. RUMBAUGH und I. JACOBSON (1999). *The Unified Modeling Language User Guide*. Addison Wesley Longman, Reading, Mass., 1. Aufl.

[BUEHLER et al. 2001] BUEHLER, CHRIS, M. BOSSE, L. MCMILLAN, S. GORTLER und M. COHEN (2001). *Unstructured Lumigraph Rendering*. In: FIUME, EUGENE, Hrsg.: *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 425–432. Addison Wesley Longman.

[CAMAHORT et al. 1998] CAMAHORT, EMILIO, A. LERIOS und D. FUSSELL (1998). *Uniformly Sampled Light Fields*. In: DRETTAKIS, GEORGE und N. MAX, Hrsg.: *Rendering Techniques '98*, S. 117–130. Springer-Verlag, Wien.

[CATMULL 1974] CATMULL, EDWIN E. (1974). *A Subdivision Algorithm for Computer Display of Curved Surfaces*. Doktorarbeit, University of Utah.

[CHAI et al. 2000] CHAI, JIN-XIANG, X. TONG, S.-C. CHAN und H.-Y. SHUM (2000). *Plenoptic Sampling*. In: AKELEY, KURT, Hrsg.: *SIGGRAPH 2000, Computer Graphics Proceedings*, Annual Conference Series, S. 307–318. Addison Wesley Longman.

[CHAN et al. 1999] CHAN, Y., M. FOK, C. FU, P. HENG und T. WANG (1999). *A panoramic-based Walkthrough System using Real Photos*. In: *Proceedings of the Seventh Pacific Conference on Computer Graphics and Applications (Pacific Graphics)*, S. 231–241.

[CHANG et al. 1999] CHANG, CHUN-FA, G. BISHOP und A. LASTRA (1999). *LDI Tree: A Hierarchical Representation for Image-Based Rendering*. In: ROCKWOOD, ALYN, Hrsg.: *Computer Graphics Proceedings, SIGGRAPH 99*, Annual Conference Series, S. 291–298. Addison Wesley Longman.

[CHEN und NGUYEN 2001] CHEN, BAOQUAN und M. X. NGUYEN (2001). *POP: A Hybrid Point and Polygon Rendering System for Large Data*. In: ERTL, THOMAS, K. JOY und A. VARSHNEY, Hrsg.: *Proceedings Visualization 2001*, S. 45–52. ACM Press, New York.

[CHEN 1995] CHEN, SHENCHANG ERIC (1995). *Quicktime VR - An Image-Based Approach to Virtual Environment Navigation*. In: COOK, ROBERT, Hrsg.: *SIGGRAPH 95, Computer Graphics Proceedings*, Annual Conference Series, S. 29–38. Addison Wesley Longman.

[CHEN und WILLIAMS 1993] CHEN, SHENCHANG ERIC und L. WILLIAMS (1993). *View Interpolation for Image Synthesis*. Computer Graphics (SIGGRAPH '93 Proceedings), 27:279–288.

[COOK 1984] COOK, R. L. (1984). *Shade trees*. Computer Graphics (SIGGRAPH '84 Proceedings), 18(3):223–231.

[DALLY et al. 1996] DALLY, WILLIAM J., L. MCMILLAN, G. BISHOP und H. FUCHS (1996). *The Delta Tree: An Object-Centered Approach to Image-Based Rendering*. Technischer Bericht AIM-1604, Massachusetts Institute of Technology, Artificial Intelligence Laboratory.

[DAUBECHIES 1988] DAUBECHIES, I. (1988). *Orthonormal Bases of Compactly Supported Wavelets*. Communications on Pure Applied Mathematics, XLI(41):909–996.

[DAVIS und NOSRATINIA 1998] DAVIS, GEOFFREY M. und A. NOSRATINIA (1998). *Wavelet-Based Image Coding: An Overview*. Applied and Computational Control, Signals, and Circuits.

[DEBEVEC 1998] DEBEVEC, PAUL (1998). *Rendering Synthetic Objects Into Real Scenes: Bridging Traditional and Image-Based Graphics With Global Illumination and High Dynamic Range Photography*. In: COHEN, MICHAEL, Hrsg.: *SIGGRAPH 98, Computer Graphics Proceedings*, Annual Conference Series, S. 189–198. Addison Wesley Longman.

[DEBEVEC et al. 2000] DEBEVEC, PAUL, T. HAWKINS, C. TCHOU, H.-P. DUIKER, W. SAROKIN und M. SAGAR (2000). *Acquiring the Reflectance Field of a Human Face*. In: AKELEY, KURT, Hrsg.: *SIGGRAPH 2000, Computer Graphics Proceedings*, Annual Conference Series, S. 145–156. Addison Wesley Longman.

[DEBEVEC und MALIK 1997] DEBEVEC, PAUL E. und J. MALIK (1997). *Recovering High Dynamic Range Radiance Maps from Photographs*. In: WHITTED, TURNER, Hrsg.: *SIGGRAPH 97, Computer Graphics Proceedings*, Annual Conference Series, S. 369–378. Addison Wesley Longman.

[DEBEVEC et al. 1996] DEBEVEC, PAUL E., C. J. TAYLOR und J. MALIK (1996). *Modeling and Rendering Architecture from Photographs: A Hybrid Geometry- and Image-Based Approach*. In: RUSHMEIER, HOLLY, Hrsg.: *SIGGRAPH 96, Computer Graphics Proceedings*, Annual Conference Series, S. 11–20. Addison Wesley Longman.

[DEBEVEC et al. 1998] DEBEVEC, PAUL E., Y. YU und G. D. BORSHUKOV (1998). *Efficient View-Dependent Image-Based Rendering With Projective Texture-Mapping*. In: DRETTAKIS, GEORGE und N. MAX, Hrsg.: *Rendering Techniques '98*, S. 105–116. Springer-Verlag, Wien.

[DECORET et al. 1999] DECORET, XAVIER, F. SILLION, G. SCHAUFLER und J. DORSEY (1999). *Multi-layered impostors for accelerated rendering*. Computer Graphics Forum, 18(3):61–73.

[DirectX] DIRECTX. http://www.microsoft.com/directx.

[DOGGETT et al. 2001] DOGGETT, MICHAEL, A. KUGLER und W. STRASSER (2001). *Displacement Mapping using Scan Conversion Hardware Architectures*. In: DUKE, D. und R. SCOPIGNO, Hrsg.: *Computer Graphics Forum*, Bd. 20(1), S. 13–26. Blackwell Publishing.

[EISERT et al. 1999] EISERT, PETER, E. STEINBACH und B. GIROD (1999). *3-D Shape Reconstruction from Light Fields Using Voxel Back-Projection*. In: *Proceedding of the 4th Conference on Vision, Modeling, and Visualization (VMV-99)*, S. 67–74.

[FELLNER et al. 2000] FELLNER, DIETER W., S. HAVEMANN, L. KOBBELT, H. P. A. LENSCH, G. MÜLLER, I. PETER, R. SCHNEIDER, H.-P. SEIDEL und W. STRASSER (2000). *Beiträge der Computergraphik zur Realisierung eines verallgemeinerten Dokumentbegriffs*. it+ti: Informationstechnik und technische Informatik, 42(6):8–16.

[FINKELSTEIN und SALESIN 1994] FINKELSTEIN, ADAM und D. H. SALESIN (1994). *Multiresolution Curves*. SIGGRAPH 94, Computer Graphics Proceedings, 28:261–268.

[FISCHER 1986] FISCHER, GERD (1986). *Lineare Algebra*. Vieweg, Braunschweig, 9. Aufl.

[FLEISHMAN et al. 1999] FLEISHMAN, S., D. COHEN-OR und D. LISCHINSKI (1999). *Automatic Camera Placement for Image-based Modeling*. In: *Proceedings of the Seventh Pacific Conference on Computer Graphics and Applications (Pacific Graphics)*, S. 12–20.

[FU et al. 1999] FU, CHI-WING, T.-T. WONG und P.-A. HENG (1999). *Computing Visibility of Triangulated Panorama*. In: LISCHINSKI, DANI und G. W. LARSON, Hrsg.: *Rendering Techniques '99*, S. 161–174. Springer-Verlag, Wien.

[GORTLER et al. 1996] GORTLER, STEVEN J., R. GRZESZCZUK, R. SZELISKI und M. F. COHEN (1996). *The Lumigraph*. In: RUSHMEIER, HOLLY, Hrsg.: *SIGGRAPH 96, Computer Graphics Proceedings*, Annual Conference Series, S. 43–54. Addison Wesley.

[GORTLER et al. 1997] GORTLER, STEVEN J., LI-WEI und M. F. COHEN (1997). *Rendering Layered Depth Images*. Technischer Bericht, Microsoft Research, Advanced Technology Division, Microsoft Corporation.

[GORTLER et al. 1993] GORTLER, STEVEN J., P. SCHRODER, M. F. COHEN und P. HANRAHAN (1993). *Wavelet Radiosity*. In: *SIGGRAPH 93, Computer Graphics Proceedings*, Annual Conference Series, S. 221–230.

[GREENE und KASS 1993] GREENE, NED und M. KASS (1993). *Hierarchical Z-Buffer Visibility*. In: *SIGGRAPH 93, Computer Graphics Proceedings*, Annual Conference Series, S. 231–240.

[GROSSMAN und DALLY 1998] GROSSMAN, J. P. und W. J. DALLY (1998). *Point Sample Rendering*. In: DRETTAKIS, GEORGE und N. MAX, Hrsg.: *Rendering Techniques '98*, S. 181–192. Springer-Verlag, Wien.

[GU et al. 1997] GU, XIANFENG, S. J. GORTLER und M. F. COHEN (1997). *Polyhedral Geometry and the Two-Plane Parameterization*. In: DORSEY, JULIE und P. SLUSALLEK, Hrsg.: *Rendering Techniques '97*, S. 1–12. Springer-Verlag, Wien.

[GUMHOLD und HÜTTNER 1999] GUMHOLD, STEFAN und T. HÜTTNER (1999). *A Multiresolution Hardware with Displacement Mapping*. In: KAUFMANN, ARIE und W. STRASSER, Hrsg.: *1999 Eurographics/SIGGRAPH Workshop on Graphics Hardware*, S. 55–66. Addison Wesley.

[GUSTAFSSON und TURBELL 1997] GUSTAFSSON, ANDERS und H. TURBELL (1997). *Image-Based Rendering*. Technischer Bericht, Image Processing Laboratory, Linköping University.

[GUTHE und STRASSER 2001] GUTHE, STEFAN und W. STRASSER (2001). *Real-time Dekompression and Visualization of Animated Volume Data*. In: ERTL, THOMAS, K. JOY und A. VARSHNEY, Hrsg.: *Proceedings Visualization 2001*, S. 349–356. ACM Press, New York.

[HALLE 1998] HALLE, MICHAEL (1998). *Multiple Viewpoint Rendering*. In: COHEN, MICHAEL, Hrsg.: *SIGGRAPH 98, Computer Graphics Proceedings*, Annual Conference Series, S. 243–254. Addison Wesley Longman.

[HANSON und WERNERT 1998] HANSON, ANDREW J. und E. A. WERNERT (1998). *Image-Based Rendering with Occlusions via Cubist Images*. In: EBERT, DAVID, H. HAGEN und H. RUSHMEIER, Hrsg.: *Proceeding Visualization '98*, S. 327–334. ACM Press, New York.

[HECKBERT 1986] HECKBERT, PAUL S. (1986). *Survey of Texture Mapping*. IEEE Computer Graphics and Applications, 6(11):56–67.

[HEIDRICH et al. 1999a] HEIDRICH, WOLFGANG, H. LENSCH, M. F. COHEN und H.-P. SEIDEL (1999a). *Light Field Techniques for Reflexions and Refractions*. In: LISCHINSKI, DANI und G. W. LARSON, Hrsg.: *Rendering Techniques '99*, S. 187–196. Springer-Verlag, Wien.

[HEIDRICH et al. 1999b] HEIDRICH, WOLFGANG, H. SCHIRMACHER, H. KÜCK und H.-P. SEIDEL (1999b). *A Warping-based Refinement of Lumigraphs*. In: THALMANN, N. und V. SKALA, Hrsg.: *Proceedings of WSCG '99*.

[HEIDRICH et al. 1997] HEIDRICH, WOLFGANG, P. SLUSALEK und H. SEIDEL (1997). *An Image-Based Model for Realistic Lens Systems in Interactive Computer Graphics*. In: DAVIS, WAYNE A., M. MANTEI und R. V. KLASSEN, Hrsg.: *Graphics Interface '97*, S. 68–75. Canadian Human-Computer Communications Society.

[HEIGL et al. 1998] HEIGL, B., J. DENZLER und H. NIEMANN (1998). *On the Application of Lightfield Reconstruction for Statistical Object Recognition*. In: *European Signal Processing Conference (EUSIPCO)*, S. 1101–1104.

[HEIGL und NIEMANN 1999] HEIGL, B. und H. NIEMANN (1999). *Camera Calibration from Extended Image Sequences for Lightfield Reconstruction*. In: *Proc. Vision, Modeling, and Visualization (VMV-99)*, S. 43–50.

[HOPPE 1996] HOPPE, HUGUES (1996). *Progressive Meshes*. In: RUSHMEIER, HOLLY, Hrsg.: *SIGGRAPH 96, Computer Graphics Proceedings*, Annual Conference Series, S. 99–108. Addison Wesley.

[HORRY et al. 1997] HORRY, YOUICHI, K. ICHI ANJYO und K. ARAI (1997). *Tour Into the Picture: Using a Spidery Mesh Interface to Make Animation from a Single Image*. In: WHITTED, TURNER, Hrsg.: *SIGGRAPH 97, Computer Graphics Proceedings*, Annual Conference Series, S. 225–232. Addison Wesley.

[IHM et al. 1997] IHM, I., S. PARK und R. LEE (1997). *Rendering of Spherical Light Fields*. In: WERNER, BOB, Hrsg.: *Proceedings of the Conference on Computer Graphics and Applications (Pacific-Graphics-97)*, S. 59–68. IEEE.

[IHM und PARK 1998] IHM, INSUNG und S. PARK (1998). *Wavelet-Based 3D Compression Scheme for Very Large Volume Data*. In: *Graphics Interface*, S. 107–116.

[IHM und PARK 1999] IHM, INSUNG und S. PARK (1999). *Wavelet-Based 3D Compression Scheme for Interactive Visualization of Very Large Volume Data*. Computer Graphics Forum, 18(1):3–15.

[ISAKSEN et al. 2000] ISAKSEN, AARON, L. MCMILLAN und S. J. GORTLER (2000). *Dynamically Reparameterized Light Fields*. In: AKELEY, KURT, Hrsg.: *SIGGRAPH 2000, Computer Graphics Proceedings*, Annual Conference Series, S. 297–306. Addison Wesley Longman.

[Java 2 Platform] JAVA 2 PLATFORM. `http:java.sun.com`.

[Java3D] JAVA3D. `http:java.sun.com/products/java-media/3D`.

[JPEG 2000] JPEG 2000. *Final Committee Draft*. ISO/IEC FCD15444-1.

[KANG et al. 2001] KANG, HYUNG WOO, S. H. PYO, K. ICHI ANJYO und S. Y. SHIN (2001). *Tour into the Picture using a Vanishing Line and its Extensions to Panoramic Images*. Computer Graphics Forum (Eurographics 2001), 20(3):132–141.

[KATAYAMA et al. 1999a] KATAYAMA, AKIHIRO, Y. SAKAGAWA und H. TAMURA (1999a). *A Method of Shading and Shadowing in Image-Based Rendering*. In: *Proceedings of the 1999 International Conference on Image Processing (ICIP-99)*, S. 26–30. IEEE.

[KATAYAMA et al. 1999b] KATAYAMA, AKIHIRO, Y. SAKAGAWA, H. YAMAMOTO und H. TAMURA (1999b). *Shading and shadow casting in image-based rendering without geometric models*. In: *SIGGRAPH 99. Conference Abstracts and Applications*, Annual Conference Series, S. 275–275. ACM Press, New York.

[KAWASAKI et al. 2001] KAWASAKI, HIROSHI, H. ARITAKI, T. OOISHI, K. IKEUCHI und M. SAKAUGHI (2001). *Image-Based Rendering for Animated Deforming Objects*. In: *SIGGRAPH 2001. Conference Abstracts and Applications*, Annual Conference Series, S. 205. ACM Press, New York.

[KEMPF et al. 1997] KEMPF, RENATE, C. FRAZIER und OPENGL ARCHITECTURE REVIEW BOARD (1997). *OpenGL reference manual: the official reference document to OpenGL, version 1.1*. Addison-Wesley Developers Press, Reading, Mass., 2. Aufl.

[KILGARD] KILGARD, MARK J. *All About OpenGL Extensions, including specifications for some significant OpenGL extensions.* Technischer Bericht, NVIDIA Corporation.

[KIM und SHIN 1999] KIM, TEA-YOUNG und Y. G. SHIN (1999). *An Efficient Wavelet-based Compression Method for Volume Rendering.* In: *Proceedings of the Seventh Pacific Conference on Computer Graphics and Applications (Pacific Graphics)*, S. 147–157.

[KIU et al. 1998] KIU, MING-HOE, X.-S. DU, R. J. MOORHEAD, D. C. BANKS und R. MACHIRAJU (1998). *Two Dimensional Sequence Compression Using MPEG.* In: *SPIE Vol. 3309, SPIE/IS&T Electronic Imaging '97*, San Jose, CA.

[KLEIN 1998] KLEIN, REINHARD (1998). *Multiresolution representations for surfaces meshes based on the vertex decimation method*. Computers & Graphics, 22(1):13–26.

[KLEIN et al. 1996] KLEIN, REINHARD, G. LIEBICH und W. STRASSER (1996). *Mesh Reduction with Error Control.* In: YAGEL, RONI und G. M. NIELSON., Hrsg.: *Proceedings Visualization '96*, S. 311–318. ACM Press, New York.

[KOCH et al. 1999a] KOCH, R., B. HEIGL, M. POLLEFEYS, L. V. GOOL und H. NIEMANN (1999a). *A Geometric Approach to Lightfield Calibration.* In: SOLINA, F. und A. LEONARDIS, Hrsg.: *Computer Analysis of Images and Patterns (CAIP '99)*, Nr. 1689 in *Lecture Notes in Computer Science*, S. 596–603. Springer-Verlag, Heidelberg.

[KOCH et al. 1999b] KOCH, R., M. POLLEFEYS, B. HEIGL, L. V. GOOL und H. NIEMANNIX (1999b). *Calibration of Hand-Held Camera Sequences for Plenoptic Modeling*. In: *Proceedings of the 7th IEEE International Conference on Computer Vision (ICCV-99)*, Bd. I, S. 585–591. IEEE.

[LALONDE und FOURNIER 1999] LALONDE, PAUL und A. FOURNIER (1999). *Interactive Rendering of Wavelet Projected Light Fields.* In: *Graphics Interface 1999*, S. 107–114.

[LANG 1997] LANG, SERGE (1997). *Introduction to Linear Algebra*. Springer-Verlag, New York, Heidelberg, 2. Aufl.

[LAVEAU und FAUGERAS 1994] LAVEAU, STEPHANE und O. FAUGERAS (1994). *3-D Scene Representation as a Collection of Images and Fundamental Matrices.*

Technischer Bericht RR-2205, Inria, Institut National de Recherche en Informatique et en Automatique.

[LEE und LEE 2000] LEE, YUNJIN und S. LEE (2000). *3D Plenoptic Functions with Cylindrical Maps*. In: *SIGGRAPH 2000, Conference Abstracts and Applications*, Annual Conference Series, S. 173. ACM Press, New York.

[LEVOY und HANRAHAN 1996] LEVOY, MARC und P. HANRAHAN (1996). *Light Field Rendering*. In: RUSHMEIER, HOLLY, Hrsg.: *SIGGRAPH 96, Computer Graphics Proceedings*, Annual Conference Series, S. 31–42. Addison Wesley.

[LEVOY et al. 2000] LEVOY, MARC, K. PULLI, B. CURLESS, S. RUSINKIEWICZ, D. KOLLER, L. PEREIRA, M. GINZTON, S. ANDERSON, J. DAVIS, J. GINSBERG, J. SHADE und D. FULK (2000). *The Digital Michelangelo Project: 3D Scanning of Large Statues*. In: AKELEY, KURT, Hrsg.: *SIGGRAPH 2000, Computer Graphics Proceedings*, Annual Conference Series, S. 131–144. Addison Wesley Longman.

[LEVOY und WHITTED 1985] LEVOY, MARC und T. WHITTED (1985). *The Use of Points as a Display Primitive*. Technischer Bericht TR 85-022, Department of Computer Science, University of North Carolina - Chapel Hill.

[LISCHINSKI und RAPPOPORT 1998] LISCHINSKI, DANI und A. RAPPOPORT (1998). *Image-Based Rendering for Non-Diffuse Synthetic Scenes*. In: DRETTAKIS, GEORGE und N. MAX, Hrsg.: *Rendering Techniques '98*, S. 301–314. Springer-Verlag, Wien.

[LIU et al. 2001] LIU, XINGUO, Y. YU und H.-Y. SHUM (2001). *Synthesizing Bidirectional Texture Functions for Real-World Surfaces*. In: FIUME, EUGENE, Hrsg.: *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 97–106. Addison Wesley Longman.

[LOSCOS et al. 2000] LOSCOS, C., G. DRETTAKIS und L. ROBERT (2000). *Interactive Virtual Relighting of Real Scenes*. IEEE Transactions on Visualization and Computer Graphics, 6(4):289–305.

[MACIEL und SHIRLEY 1995] MACIEL, PAULO W. C. und P. SHIRLEY (1995). *Visual Navigation of Large Environments Using Textured Clusters*. In: HANRAHAN, PAT und J. WINGET, Hrsg.: *1995 Symposium on Interactive 3D Graphics*, S. 95–102.

[MAGDA et al. 2000] MAGDA, SEBASTIAN, J. LU, P. N. BELHUMEUR und DAVID J. KRIEGMAN (2000). *Shedding Light on Image-Based Rendering*. In: *SIGGRAPH 2000. Conference Abstracts and Applications*, Annual Conference Series, S. 255. ACM Press, New York.

[MAGNOR und GIROD 1999a] MAGNOR, M. und B. GIROD (1999a). *Adaptive Block-based Light Field Coding*. Proc. International Workshop on Synthetic-Natural Hybrid Coding and Three Dimensional Imaging (IWSNHC3DI'99), S. 140–143.

[MAGNOR und GIROD 1999b] MAGNOR, M. und B. GIROD (1999b). *Hierarchical Coding of Light Fields with Disparity Maps*. Proc. IEEE International Conference on Image Processing (ICIP'99), 3:334–338.

[MAGNOR und GIROD 2000] MAGNOR, M. und B. GIROD (2000). *Data Compression for Light Field Rendering*. IEEE Trans. Circuits and Systems for Video Technology, 10(3):338–343.

[MAGNOR 2000] MAGNOR, MARCUS (2000). *Geometry-Adaptive Multi-View Coding Techniques for Image Based Rendering*. Doktorarbeit, Friedrich-Alexander-Universität Erlangen-Nürnberg.

[MAGNOR 2001] MAGNOR, MARCUS (2001). *Persönliche eMail von Marcus Magnor an Ingmar Peter am 16.11.2001*.

[MAGNOR et al. 2000] MAGNOR, MARCUS, A. ENDMANN und B. GIROD (2000). *Progressive Compression and Rendering of Light Fields*. In: *Proc. Vision, Modeling, and Visualization (VMV-2000)*, S. 199–203.

[MALZBENDER et al. 2001] MALZBENDER, TOM, D. GELB und H. WOLTERS (2001). *Polynomial Texture Maps*. In: FIUME, EUGENE, Hrsg.: *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 519–528. Addison Wesley Longman.

[MARK et al. 1997] MARK, WILLIAM R., L. MCMILLAN und G. BISHOP (1997). *Post-Rendering 3D Warping*. In: COHEN, MICHAEL und D. ZELTZER, Hrsg.: *1997 Symposium on Interactive 3D Graphics*, S. 7–16.

[MATUSIK et al. 2001] MATUSIK, WOJCIECH, C. BUEHLER und L. MCMILLAN (2001). *Polyhedral Visual Hulls for Real-Time Rendering*. In: GORTLER, S. J. und M. MYSZKOWSKI, Hrsg.: *Rendering Techniques 2001*, S. 115–126. Springer-Verlag, Wien.

[MATUSIK et al. 2000] MATUSIK, WOJCIECH, C. BUEHLER, R. RASKAR, S. J. GORTLER und L. MCMILLAN (2000). *Image-Based Visual Hulls*. In: AKELEY, KURT, Hrsg.: *SIGGRAPH 2000, Computer Graphics Proceedings*, Annual Conference Series, S. 369–374. Addison Wesley Longman.

[MAX und OHSAKI 1995] MAX, NELSON und K. OHSAKI (1995). *Rendering Tree from Precomputed Z-Buffer Views*. In: HANRAHAN, PATRICK M. und W. PURGATHOFER, Hrsg.: *Rendering Techniques '95*, S. 74–81. Springer-Verlag, Wien.

[MCALLISTER et al. 1999] MCALLISTER, LARS F. NYLAND DAVID K., V. POPESCU, A. LASTRA und C. MCCUE (1999). *Real-Time Rendering of Real World Environnements*. In: LISCHINSKI, DANI und G. W. LARSON, Hrsg.: *Rendering Techniques '99*, S. 145–160. Springer-Verlag, Wien.

[MCMILLAN 1995] MCMILLAN, LEONARD (1995). *A List-Priority Rendering Algorithm for Redisplaying Projected Surfaces*. Technischer Bericht TR95-005, Department of Computer Science, University of North Carolina - Chapel Hill.

[MCMILLAN und BISHOP 1995] MCMILLAN, LEONARD und G. BISHOP (1995). *Plenoptic Modeling: An Image-Based Rendering System.* In: COOK, ROBERT, Hrsg.: *SIGGRAPH 95, Computer Graphics Proceedings*, Annual Conference Series, S. 39–46. Addison Wesley.

[MCMILLAN und GORTLER 1999] MCMILLAN, LEONARD und S. GORTLER (1999). *Image-Based Rendering: A New Interface Between Computer Vision and Computer Graphics.* Computer Graphics, S. 61–64.

[MILLER et al. 1998] MILLER, GAVIN, S. RUBIN und D. PONCELEON (1998). *Lazy decompression of surface light fields for precomputer global illumination.* In: DRETTAKIS, G. und N. MAX, Hrsg.: *Rendering Techniques '98*, S. 281–292. Springer-Verlag, Wien.

[MÜLLER 1999] MÜLLER, HEINRICH (1999). *Image-Based Rendering: A Survey.* In: MUDUR, S. P. und D. SHIKHARE, Hrsg.: *ICVC99 - International Conference on Visual Computing*, S. 136–143. National Centre for Software Technology, Mumbai, India, Fontasey Typesetters Pvt. Ltd.

[MURAKI 1992] MURAKI, SHIGERU (1992). *Approximation and Rendering of Volume Data Using Wavelet Transforms.* In: *Proceedings Visualization '92*, S. 21–28. IEEE.

[MURAKI 1993] MURAKI, SHIGERU (1993). *Volume data and wavelet transform.* IEEE Computer Graphics and Applications, 13(4):50–56.

[OH et al. 2001] OH, BYONG MOK, M. CHEN, J. DORSEY und F. DURAND (2001). *Image-Based Modelling and Photo Editing.* In: FIUME, EUGENE, Hrsg.: *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 433–442. Addison Wesley Longman.

[OLIVEIRA und BISHOP 1999a] OLIVEIRA, MANUEL und G. BISHOP (1999a). *Relief Textures.* Technischer Bericht TR99-015, Department of Computer Science, University of North Carolina - Chapel Hill.

[OLIVEIRA und BISHOP 1998] OLIVEIRA, MANUEL M. und G. BISHOP (1998). *Dynamic Shading in Image-Based Rendering.* Technischer Bericht TR98-023, Department of Computer Science, University of North Carolina - Chapel Hill.

[OLIVEIRA und BISHOP 1999b] OLIVEIRA, MANUEL M. und G. BISHOP (1999b). *Image-Based Objects.* In: SPENCER, STEPHEN N., Hrsg.: *Proceedings of the Conference on the 1999 Symposium on interactive 3D Graphics*, S. 191–198. Addison Wesley Longman.

[OLIVEIRA et al. 2000] OLIVEIRA, MANUEL M., G. BISHOP und D. MCALLISTER (2000). *Relief Texture Mapping.* In: AKELEY, KURT, Hrsg.: *SIGGRAPH 2000, Computer Graphics Proceedings*, Annual Conference Series, S. 359–368. Addison Wesley Longman.

[PALL 1993] PALL, MICHAEL (1993). *Einsatz von Subbandcodierung und Wavelets bei der Kompression von Bilddaten.* Diplomarbeit, Institut für Algorithmen und Kognitive Systeme, Fakultät für Informatik, Universität Karlsruhe (TH).

[PETER und GUMHOLD 1999] PETER, INGMAR und S. GUMHOLD (1999). *Teaching Computer Graphics with Java 3D.* In: SYED, M. R. und O. R. BAIOCCHI, Hrsg.: *Advances in Multimedia and Distance Education (Proceedings of ISIMADE '99)*, S. 95–99. University of Windsor.

[PETER und PIETREK 1998] PETER, INGMAR und G. PIETREK (1998). *Importance Driven Construction of Photon Maps.* In: GEORGE DRETTAKIS, NELSON MAX, Hrsg.: *Rendering Techniques '98*, S. 269–280. Springer-Verlag, Wien.

[PETER und STRASSER 1999] PETER, INGMAR und W. STRASSER (1999). *The Wavelet Stream: Progressive Transmission of Compressed Light Field Data*. In: VARSHNEY, AMITABH, C. M. WITTENBRINK und H. HAGEN, Hrsg.: *IEEE Visualization 1999 Late Breaking Hot Topics*, S. 69–72.

[PETER und STRASSER 2001a] PETER, INGMAR und W. STRASSER (2001a). *Multi-Resolution Light Field Rendering with the Wavelet Stream.* In: *SIGGRAPH 2001, Conference Abstracts and Applications*, Annual Conference Series, S. 234. ACM Press, New York.

[PETER und STRASSER 2001b] PETER, INGMAR und W. STRASSER (2001b). *The Wavelet Stream: Interactive Multi Resolution Light Field Rendering*. In: GORTLER, S. J. und M. MYSZKOWSKI, Hrsg.: *Rendering Techniques 2001*, S. 127–138. Springer-Verlag, Wien.

[PETER und STRASSER 2001c] PETER, INGMAR und W. STRASSER (2001c). *The Wavelet Stream: Interactive Multi Resolution Light Field Rendering.* Technischer Bericht WSI-2001-09, Wilhelm-Schickard-Institut für Informatik, Graphisch-Interaktive Systeme (WSI/GRIS), Universität Tübingen.

[PFISTER et al. 2000] PFISTER, HANSPETER, M. ZWICKER, J. VAN BAAR und M. GROSS (2000). *Surfels: Surface Elements as Rendering Primitives*. In: AKELEY, KURT, Hrsg.: *SIGGRAPH 2000, Computer Graphics Proceedings*, Annual Conference Series, S. 335–342. Addison Wesley Longman.

[PG MOCART 1996] PG MOCART (1996). *Ein objektorientiertes Programmsystem für Monte-Carlo Bildsynthese*. Interner Bericht, Lehrstuhl VII, Fachbereich Informatik, Universität Dortmund.

[PHONG 1975] PHONG, BUI-TUONG (1975). *Illumination for computer generated pictures*. Communications of the ACM, 18(6):311–317.

[PIETREK 2001] PIETREK, GEORG (2001). *Verfahren zur verbesserten Approximation von Lichtverteilungen in der fotorealistischen Bildsynthese*. Doktorarbeit, Fachbereich Informatik, Universität Dortmund.

[PIETREK und PETER 1999] PIETREK, GEORG und I. PETER (1999). *Adaptive Wavelet Densities for Monte Carlo Ray Tracing*. In: THALMAN, N. und V. SKALA, Hrsg.: *Proceedings of WSCG '99*, Bd. 1, S. 217–224.

[POPESCU et al. 2000] POPESCU, VOICU, J. EYLES, A. LASTRA, J. STEINHURST, N. ENGLAND und L. NYLAND (2000). *The WarpEngine: An Architecture for the*

Post-Polygonal Age. In: AKELEY, KURT, Hrsg.: *SIGGRAPH 2000, Computer Graphics Proceedings*, Annual Conference Series, S. 433–442. Addison Wesley Longman.

[POPESCU et al. 1998] POPESCU, VOICU S., A. LASTRA, D. G. ALIAGA und M. M. DE OLIVEIRA NETO (1998). *Efficient Warping for Architectural Walkthroughs using Layered Depth Images*. In: *Proceedings Visualization '98*, S. 211–216. ACM Press, New York.

[POULIN et al. 1998] POULIN, PIERRE, M. OUIMET und M. C. FRASSON (1998). *Interactively Modeling with Photogrammetry*. In: DRETTAKIS, GEORGE und N. MAX, Hrsg.: *Rendering Techniques '98*, S. 93–104. Springer-Verlag, Wien.

[PULLI et al. 1997] PULLI, KARI, M. COHEN, T. DUCHAMP, H. HOPPE, L. SHAPIRO und W. STUETZLE (1997). *View-based Rendering: Visualizing Real Objects from Scanned Range and Color Data*. In: DORSEY, JULIE und P. SLUSALLEK, Hrsg.: *Rendering Techniques '97*, S. 23–34. Springer-Verlag, Wien.

[RADEMACHER und BISHOP 1998] RADEMACHER, PAUL und G. BISHOP (1998). *Multiple-Center-of-Projection Images*. In: COHEN, MICHAEL, Hrsg.: *SIGGRAPH 98, Computer Graphics Proceedings*, Annual Conference Series, S. 199–206. Addison Wesley.

[RAFFERTY et al. 1998a] RAFFERTY, MATTHEW M., D. G. ALIAGA und A. A. LASTRA (1998a). *3D Image Warping in Architectural Walkthroughs*. In: *Proceedings of VRAIS '98*, S. 228–233.

[RAFFERTY et al. 1998b] RAFFERTY, MATTHEW M., D. G. ALIAGA, V. POPESCU und A. A. LASTRA (1998b). *Images for Accelerating Architectural Walkthroughs*. IEEE Computer Graphics & Applications, 18(6):38–45.

[RAMAMOORTHI und HANRAHAN 2001] RAMAMOORTHI, RAVI und P. HANRAHAN (2001). *A Signal-Processing Framework for Inversive Rendering*. In: FIUME, EUGENE, Hrsg.: *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 117–128. Addison Wesley Longman.

[REGAN et al. 1999] REGAN, MATTHEW J. P., G. S. P. MILLER, S. M. RUBIN und C. KOGELNIK (1999). *A Real Time Low-Latency Hardware Light-Field Renderer*. In: ROCKWOOD, ALYN, Hrsg.: *SIGGRAPH 99, Computer Graphics Proceedings*, Annual Conference Series, S. 287–290. Addison Wesley Longman.

[RODLER 1999] RODLER, FLEMMING FRICHE (1999). *Wavelet Based 3D Compression with Fast Random Access for Very Large Volume Data*. In: *Proceedings of the Seventh Pacific Conference on Computer Graphics and Applications (Pacific Graphics)*, S. 108–117.

[RUMBAUGH et al. 1999] RUMBAUGH, JAMES, I. JACOBSON und G. BOOCH (1999). *The Unified Modeling Language Reference Manual*. Object Technology Series. Addison Wesley Longman, Reading, Mass.

[RUSINKIEWICZ und LEVOY 2000] RUSINKIEWICZ, SZYMON und M. LEVOY (2000). *QSplat: A Multiresolution Point Rendering System for Large Meshes*. In: AKELEY, KURT, Hrsg.: *SIGGRAPH 2000, Computer Graphics Proceedings*, Annual Conference Series, S. 343–352. Addison Wesley Longman.

[SAHA 2000] SAHA, SUBHASIS (2000). *Image Compression - from DCT to Wavelets: A Review*. ACM Crossroads Magazine. `http:www.acm.org/crossroads`.

[SAID und PEARLMAN 1996] SAID, AMIR und W. A. PEARLMAN (1996). *A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees*. IEEE Transactions on Circuits and Systems for Video Technology, 6:243–250.

[SALOMON 2000] SALOMON, DAVID (2000). *Data Compression: The Complete Reference*. Springer-Verlag, Berlin, London, 2. Aufl.

[SCHAUFLER 1998] SCHAUFLER, GERNOT (1998). *Per-Object Image Warping with Layered Impostors*. In: DRETTAKIS, GEORGE und N. MAX, Hrsg.: *Rendering Techniques '98*, S. 145–156. Springer-Verlag, Wien.

[SCHAUFLER und PRIGLINGER 1999] SCHAUFLER, GERNOT und M. PRIGLINGER (1999). *Efficient Displacement Mapping by Image Warping*. In: LISCHINSKI, DANI und G. W. LARSON, Hrsg.: *Rendering Techniques '99*, S. 175–186. Springer-Verlag, Wien.

[SCHIRMACHER 2000] SCHIRMACHER, HARTMUT (2000). *Warping Techniques for Light Fields*. In: *Procedings of Grafiktag 2000*, S. 1–7. Gesellschaft für Informatik.

[SCHIRMACHER et al. 1999a] SCHIRMACHER, HARTMUT, W. HEIDRICH, M. RUBICK, D. SCHIRON und H.-P. SEIDEL (1999a). *Image-Based BRDF Reconstruction*. In: *Proceeding of the 4th Conference on Vision, Modeling, and Visualization (VMV-99)*, S. 285–292.

[SCHIRMACHER et al. 1999b] SCHIRMACHER, HARTMUT, W. HEIDRICH und H.-P. SEIDEL (1999b). *Adaptive Acquisition of Lumigraphs from Synthetic Scenes*. In: BRUNET, P. und R. SCOPIGNO, Hrsg.: *Computer Graphics Forum (Eurographics '99)*, Bd. 18(3), S. 151–160. The Eurographics Association and Blackwell Publishers.

[SCHIRMACHER et al. 2000] SCHIRMACHER, HARTMUT, W. HEIDRICH und H.-P. SEIDEL (2000). *High-Quality Interactive Lumigraph Rendering Through Warping*. In: *Graphics Interface*, S. 87–94.

[SCHIRMACHER et al. 2001] SCHIRMACHER, HARTMUT, L. MING und H.-P. SEIDEL (2001). *On-the-Fly Processing of Generalized Lumigraphs*. Computer Graphics Forum (Eurographics 2001), 20(3):165–173.

[SCHMALSTIEG et al. 1996] SCHMALSTIEG, D., A. L. FUHRMANN, M. GERVAUTZ und Z. SZALAVÁRI (1996). *'Studierstube' - An Environment for Collaboration in Augmented Reality'*. In: *Proceedings of Collaborative Virtual Environments '96*.

[SCHMALSTIEG et al. 1999] SCHMALSTIEG, DIETER, L. M. ENCARNAÇÁO und Z. SZALAVÁRI (1999). *Using transparent props for interaction with the virtual table*. In: SPENCER, STEPHEN N., Hrsg.: *Proceedings of the Conference on the 1999 Symposium on Interactive 3D Graphics*, S. 147–154. ACM Press, New York.

[SCHRÖDER et al. 1994] SCHRÖDER, P., S. J. GORTLER, M. F. COHEN und P. HANRAHAN (1994). *Wavelet Projections for Radiosity*. Computer Graphics Forum, 13(2):141–151.

[SGI Open Inventor] SGI OPEN INVENTOR. `http:www.sgi.com/software/inventor`.

[SHADE et al. 1996] SHADE, JONATHAN, D. LISCHINSKI, D. SALESIN, T. DEROSE und J. SNYDER (1996). *Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments*. In: RUSHMEIER, HOLLY, Hrsg.: *SIGGRAPH 96, Computer Graphics Proceedings*, Annual Conference Series, S. 75–82. Addison Wesley.

[SHADE et al. 1998] SHADE, JONATHAN W., S. J. GORTLER, L. HE und R. SZELISKI (1998). *Layered Depth Images*. In: COHEN, MICHAEL, Hrsg.: *SIGGRAPH 98, Computer Graphics Proceedings*, Annual Conference Series, S. 231–242. Addison Wesley.

[SHANNON 1948] SHANNON, C. E. (1948). *A mathematical theory of communication.*. Bell Syst. Technical Journal, 27:379–423, 623–656.

[SHAPIRO 1993] SHAPIRO, JEROME M. (1993). *Embedded Image Coding Using Zerotrees of Wavelet Coefficients*. IEEE Transactions on Signal Processing, 41(12):3445–3462.

[SHEN et al. 2001] SHEN, CHEN, H.-Y. SHUM und J. F. O'BRIEN (2001). *Image-Based Rendering and Illumination using Spherical Mosaics*. In: *SIGGRAPH 2001, Conference Abstracts and Applications*, Annual Conference Series, S. 202. ACM Press, New York.

[SHUM und HE 1999] SHUM, HEUNG-YEUNG und L.-W. HE (1999). *Rendering with Concentric Mosaics*. In: ROCKWOOD, ALYN, Hrsg.: *SIGGRAPH 99, Computer Graphics Proceedings*, Annual Conference Series, S. 299–306. Addison Wesley Longman.

[SIEGEL und HOWELL 1981] SIEGEL, ROBERT und J. R. HOWELL (1981). *Thermal Radiation Heat Transfer*. Hemisphere Publishing Corp., Washington, DC.

[SILLION et al. 1997] SILLION, FRANÇOIS, G. DRETTAKIS und B. BODELET (1997). *Efficient Impostor Manipulationfor Real-Time Visualization of Urban Scenery*. Computer Graphics Forum (Eurographics '97), 16(3):207–218.

[SLFA] SLFA. *The Stanford Light Fields Archive*. `www-graphics.stanford.edu/software/lightpack/lifs.html`.

[SLOAN et al. 1997] SLOAN, PETER-PIKE, M. F. COHEN und S. J. GORTLER (1997). *Time Critical Lumigraph Rendering*. In: COHEN, MICHAEL und D. ZELTZER, Hrsg.: *1997 Symposium on Interactive 3D Graphics*, S. 17–24.

[SLOAN und HANSEN 1999] SLOAN, PETER-PIKE und C. HANSEN (1999). *Parallel Lumigraph Reconstruction*. In: SPENCER, STEPHAN N., Hrsg.: *Proceedings of the 1999 IEEE Parallel Visualization and Graphics Symposium (PVGS'99)*, S. 7–14. ACM Siggraph.

[STAMMINGER und DRETTAKIS 2001] STAMMINGER, MARC und G. DRETTAKIS (2001). *Interactive Sampling and Rendering for Complex and Procedural Geometry*. In: GORTLER, S. J. und M. MYSZKOWSKI, Hrsg.: *Rendering Techniques 2001*, S. 151–162.

[STOAKLEY et al. 1995] STOAKLEY, RICHARD, M. J. CONWAY und R. PAUSCH (1995). *Virtual Reality on a WIM: Interactive Worlds in Miniature*. In: *Proceedings of ACM CHI'95 (Conference on Human Factors in Computing Systems)*, Bd. 1 d. Reihe *Papers: Innovative Interaction I*, S. 265–272.

[STOEV et al. 2000] STOEV, STANISLAV L., I. PETER und W. STRASSER (2000). *The Multi LDI: An Image Based Rendering Approach for Interaction, Navigation, and Visualization in Complex Virtual Environments*. Technischer Bericht WSI-2000-23, Wilhelm-Schickard-Institut für Informatik, Graphisch-Interaktive Systeme (WSI/GRIS), Universität Tübingen.

[STOEV et al. 2001] STOEV, STANISLAV L., I. PETER und W. STRASSER (2001). *Interaction, Navigation, and Visualization Props in Complex Virtual Environments Using Image Based Rendering Techniques*. In: *Proceedings of the IEEE Virtual Reality*.

[STOLLNITZ et al. 1995a] STOLLNITZ, ERIC J., T. D. DEROSE und D. H. SALESIN (1995a). *Wavelets for Computer Graphics: A Primer, Part 1*. IEEE Computer Graphics and Applications, 15(3):76–84.

[STOLLNITZ et al. 1995b] STOLLNITZ, ERIC J., T. D. DEROSE und D. H. SALESIN (1995b). *Wavelets for Computer Graphics: Part 2*. IEEE Computer Graphics and Applications, 15(4):75–85.

[STOLLNITZ et al. 1996] STOLLNITZ, ERIC J., T. D. DEROSE und D. H. SALESIN (1996). *Wavelets for Computer Graphics: Theory and Applications*. Morgan Kaufmann, San Francisco, CA.

[SUDARSKY und GOTSMAN 1996] SUDARSKY, ODED und C. GOTSMAN (1996). *Output-Sensitive Visibility Algorithms for Dynamic Scenes with Applications to Virtual Reality*. Computer Graphics Forum (Eurographics '96), 15(3):249–258.

[SZELISKI und SHUM 1997] SZELISKI, RICHARD und H.-Y. SHUM (1997). *Creating Full View Panoramic Mosaics and Environment Maps*. In: WHITTED, TURNER, Hrsg.: *SIGGRAPH 97, Computer Graphics Proceedings*, Annual Conference Series, S. 251–258. Addison Wesley.

[TARINI et al. 2000] TARINI, M., P. CIGNONI, C. ROCCHINI und R. SCOPIGNO (2000). *Computer Assisted Reconstruction of Buildings from Photographic Data*. In: *Proceedings Vision, Modeling, and Visualization (VMV-2000)*, S. 213–220.

[TELLER und HANRAHAN 1993] TELLER, SETH und P. HANRAHAN (1993). *Global Visibility Algorithms for Illumination Computations.* In: *SIGGRAPH 93, Computer Graphics Proceedings*, Annual Conference Series, S. 239–246.

[TERFLOTH 2001] TERFLOTH, KIRSTEN (2001). *Implementierung eines LightField-Viewers in Java 3D.* Studienarbeit, WSI/GRIS, Universität Tübingen.

[TGS Open Inventor] TGS OPEN INVENTOR. `http:www.tgs.com/pro_div/oiv_main.htm`.

[TONG und GRAY 1999] TONG, XIN und R. M. GRAY (1999). *Compression of Light Fields using Disparity Compensation and Vector Quantization.* In: *Proceedings of the Second IASTED International Conference - Computer Graphics and Imaging.*

[TONG und GRAY 2000] TONG, XIN und R. M. GRAY (2000). *Coding of Multi-View Images for Immersive Viewing.* Proc. IEEE International Conference on Acoustic, Speech and Signal Processing (ICASSP-2000), 4:1879–1882.

[Trolltech] TROLLTECH. `http:www.troll.no`.

[TSANG et al. 1998] TSANG, GLENN, S. GHALI, E. L. FIUME und A. N. VENETSANOPOULOS (1998). *A Novel Parameterization of the Light Field.* In: *Proceedings of the Tenth Image and Multidimensional Digital Signal Processing Workshop (IMDSP '98)*, S. 319–322.

[VirTü] VIRTÜ. *Virtual Tübingen, Prof. Bülthoff, MPI Tübingen.* `www.kyb.tuebingen.mpg.de/bu/projects/vrtueb`.

[VOGELSANG et al. 2000] VOGELSANG, C., B. HEIGL, G. GREINER und H. NIEMANN (2000). *Automatic Image-Based Scene Model Acquisition and Visualization.* In: *Proceedings Vision, Modeling, and Visualization (VMV-2000)*, S. 189–198.

[WALD et al. 2001] WALD, INGO, P. SLUSALLEK und C. BENTHIN (2001). *Interactive Distributed Ray Tracing of Highly Complex Models.* In: GORTLER, S. J. und M. MYSZKOWSKI, Hrsg.: *Rendering Techniques 2001*, S. 277–288. Springer-Verlag, Wien.

[WAND et al. 2001] WAND, MICHAEL, M. FISCHER, I. PETER, F. M. AUF DER HEIDE und W. STRASSER (2001). *The Randomized z-Buffer Algorithm: Interactive Rendering of Highly Complex Scenes.* In: FIUME, EUGENE, Hrsg.: *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 361–370. Addison Wesley Longman.

[WERNECKE 1994] WERNECKE, JOSIE (1994). *The Inventor Mentor*. Addison-Wesley, Reading, Mass.

[WIMMER et al. 1999] WIMMER, MICHAEL, M. GIEGL und D. SCHMALSTIEG (1999). *Fast walkthroughs with image caches and ray casting.* Computers and Graphics, 23(6):831–838.

[WONG et al. 1997a] WONG, T., P. HENG, S. OR und W. NG (1997a). *Illuminating Image-based Objects*. In: WERNER, BOB, Hrsg.: *Proceedings of the Conference on Computer Graphics and Applications (Pacific-Graphics-97)*, S. 69–79.

[WONG et al. 2001] WONG, TIEN-TSIN, P.-A. HENG und C.-W. FU (2001). *Interactive Relighting of Panoramas*. IEEE Computer Graphics and Applications, 21(2):32–41.

[WONG et al. 1997b] WONG, TIEN-TSIN, P. HENG, S. OR und W. NG (1997b). *Image-based Rendering with Controllable Illumination*. In: DORSEY, JULIE und P. SLUSALLEK, Hrsg.: *Rendering Techniques '97*, S. 13–22. Springer-Verlag, Wien, New York.

[WONKA und SCHMALSTEIG 1999] WONKA, PETER und D. SCHMALSTEIG (1999). *Occluder Shadows for Fast Walkthroughs of Urban Environments*. Computer Graphics Forum, 18(3):51–60.

[WOOD et al. 2000] WOOD, DANIEL N., D. I. AZUMA, K. ALDINGER, B. CURLESS, T. DUCHAMP, D. H. SALESIN und W. STUETZLE (2000). *Surface Light Fields for 3D Photography*. In: AKELEY, KURT, Hrsg.: *SIGGRAPH 2000, Computer Graphics Proceedings*, Annual Conference Series, S. 287–296. Addison Wesley Longman.

[XU et al. 2001] XU, YING-QING, Y. CHEN, S. LIN, H. ZHONG, E. WU, B. GUO und H.-Y. SHUM (2001). *Photorealistic Rendering of Knitware Using the Lumislice*. In: FIUME, EUGENE, Hrsg.: *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 391–398. Addison Wesley Longman.

[YANG et al. 2000] YANG, JASON C., C. LEE, A. ISAKSEN und L. MCMILLAN (2000). *A Low-Cost, Portable Light Field Capture Device*. In: *SIGGRAPH 2000, Conference Abstracts and Applications*, Annual Conference Series, S. 224. ACM Press, New York.

[YU et al. 1999] YU, YIZHOU, P. DEBEVEC, J. MALIK und T. HAWKINS (1999). *Inverse Global Illumination: Recovering Reflectance Models of Real Scenes From Photographs*. In: ROCKWOOD, ALYN, Hrsg.: *SIGGRAPH 99, Computer Graphics Proceedings*, Annual Conference Series, S. 215–224. Addison Wesley Longman.

[YU und MALIK 1998] YU, YIZHOU und J. MALIK (1998). *Recovering Photometric Properties of Architectural Scenes from Photographs*. In: COHEN, MICHAEL, Hrsg.: *SIGGRAPH 98, Computer Graphics Proceedings*, Annual Conference Series, S. 207–218. Addison Wesley.

[ZHANG et al. 1997] ZHANG, HANSONG, D. MANOCHA, T. HUDSON und K. E. HOFF III (1997). *Visibility Culling Using Hierarchical Occlusion Maps*. In: WHITTED, TURNER, Hrsg.: *SIGGRAPH 97, Computer Graphics Proceedings*, Annual Conference Series, S. 77–88. Addison Wesley.

[ZIV und LEMPEL 1977] ZIV, J. und A. LEMPEL (1977). *A universal algorithm for sequential data compression.*. IEEE Transactions on Information Theory, 23:337–343.

[ZIV und LEMPEL 1978] ZIV, J. und A. LEMPEL (1978). *Compression of Individual Sequences via Variable-Rate Coding*. IEEE Transactions on Information Theory, 24(5).

[ZWICKER et al. 2001] ZWICKER, MATTHIAS, H. PFISTER, J. VAN BAAR und M. GROSS (2001). *Surface Splatting*. In: FIUME, EUGENE, Hrsg.: *SIGGRAPH 2001, Computer Graphics Proceedings*, Annual Conference Series, S. 371–378. Addison Wesley Longman.

Tabellarischer Lebenslauf

26. September 1970	geboren in Essen
1977 - 1981	Altfriedschule in Essen-Frintrop
1981 - 1990	Don-Bosco Gymnasium, Essen
1990	Abitur
1990 - 1991	Zivildienst
1991 - 1993	Studium der Informatik an der Universität Kaiserslautern
1993 - 1998	Studium der Informatik an der Universität Dortmund
1998	Abschluß als Diplom-Informatiker (Dipl.-Inform.)
1992	Werkstudent bei der Ruhrgas AG in Essen
1994 - 1998	Studentische Hilfskraft am Lehrstuhl für Computer Graphik (LS 7) im Fachbereich Informatik der Universität Dortmund (Prof. Dr. Heinrich Müller)
1996 - 1997	Studentische Hilfskraft am Lehrstuhl Klimagerechte Architektur am Fachbereich Bauwesen an der Universität Dortmund (Prof. Dr.-Ing. Helmut Müller)
seit 1998	Wissenschaftlicher Mitarbeiter am Lehrstuhl für Graphisch Interaktive Systeme am Wilhelm-Schickard-Institut für Informatik (WSI/GRIS) der Eberhard-Karls-Universität Tübingen (Prof. Dr.-Ing. W. Straßer)