Light Field und Lumigraph

Ein neuer Ansatz des Image Based Renderings

GDV Fachseminar SS 97
Patrick Diezi

Vorwort

Der Inhalt dieser Ausarbeitung bezieht sich auf zwei Paper die auf der SIGGRAPH '96 vorgestellt wurden. Hierbei handelt es sich um:

Light Field Rendering
Marc Levoy, Pat Hanrahan
(Seiten 31-42)

The Lumigraph
Steven J Gortler, Radek Grzeszczuk, Richard Szeliski, Michael F. Cohen
(Seiten 43-54)

Im folgenden soll der Inhalt dieser Paper dargestellt und zusammengefasst werden. Angefangen mit einer beide Artikel betreffenden Motivation und Einleitung geht es über in die ebenfalls fast identische Repräsentation. Anschliessend sollen die Implementationen separat behandelt werden. Auch wenn der Lumigraph im Paper auf das Light Field verweist und somit das Light Field Paper sich aufdrängt zur Erstbetrachtung, behandle ich zuerst die Implementation des Lumigraphs, da die Darstellungen dort mehr mathematisch unterlegt sind und darum meiner Meinung nach grundlegender sind und sich damit zur Erstbetrachtung besser eignen. Den Abschluss dieser Ausarbeitung bildet eine beide Artikel betreffende Diskussion, die Aussicht auf zukünftige Forschungsmöglichkeiten, sowie meine persönliche Meinung zu diesen Methoden.

Motivation

Immer mehr kommt der Wunsch auf, in Realtime realistische Szenen darstellen zu können. Realtime und Realismus sind aber zwei Wünsche, die nur schwer gleichzeitig erfüllbar sind. Soll eine Darstellung realistisch sein, so ist dies oft mit enormem Rechenaufwand verbunden. Genauso hat der Wunsch nach Realtime oft eine Herabsetzung des Realismus zur Folge. Im Laufe der Zeit wurden Methoden wie Ray Tracing, verschiedene Arten des Renderings und Radiosity-Verfahren entwickelt, die vorallem versuchen Oberflächeneigenschaften, Materialien und eine Beleuchtung möglichst realistisch zu erzeugen. Dies ist aber mit Rechenaufwand verbunden, der nicht in Realtime geleistet werden kann. So wurde versucht das Problem von einer anderen Seite her anzupacken. Es wurden Methoden entwickelt, die auf dem Image Based Rendering basieren. Hier wird versucht auf irgendeine Art aus einer Kollektion von realistischen Bildern (digital oder synthetisch), neue Bilder entsprechend einer gewünschten Kameraposition und Richtung zu generieren. Dazu wurden Methoden entwickelt, die dies mit Hilfe von Tiefeninformationen, mit der Bewegung eines Pixels zwischen zwei Bildern, mit Abhängigkeiten zwischen Bildern oder mit Environments Maps [Chen95] bewältigen. Doch auch diese Methoden haben diverse Nachteile wie zum Beispiel komplexe Berechnungen, grosse Datenmengen, nicht genügend gute Darstellung oder zu eingeschränkte Möglichkeiten für die Plazierung der virtuellen Kamera. Mit den auf dem Image Based Rendering basierenden Methoden des Light Fields und des Lumigraphs wird versucht, die Nachteile der vorhin erwähnten Methoden zu beseitigen und einen Schritt näher an den Wunsch von realistischen Bildern, die in Realtime darstellbar sind, zu kommen.

Einleitung

Das Light Field Rendering wie auch der Lumigraph sind eine Methode zur einfachen und robusten Generierung neuer Sichten aus beliebigen virtuellen Kamerapositionen ohne Verwendung von Tiefeninformationen oder sonstigem Wissen von Parametern einer Szene, nur durch Kombinieren und Wiedergeben der vorhandenen Bilder, welche von realer, synthetischer oder gemischter Herkunft sein können. Der Schlüssel darin liegt in der Interpretation von Bildern als 2D Ausschnitt einer 4D Funktion L - dem Light Field oder Lumigraph.
Diese 4D Funktion ist eine Reduzierung der 5-dimensionalen plenoptic function [Adelson91] auf vier Dimensionen. Die plenoptic function stellt den Lichtfluss an jedem Ort (x,y,z) in jede beliebige Richtung (θ,ϕ) dar. Beschränkt man sich auf Kamerapositionen im freien Raum, so kann auf Grund dar Tatsache, dass sich der Lichtfluss in der angenommenen transparenten Luft nur beim Übergang in ein anderes Medium ändert, mit nur 4 Dimensionen dargestellt werden (siehe weiter unten Darstellung). Zudem bezieht sich diese Funktion nicht explizit auf den optischen Fluss, sondern approximiert diesen.
Diese Funktion charakterisiert komplett den Lichtfluss durch den nicht versperrten Raum einer statischen Szene mit fixer Beleuchtung. Es soll eine Darstellung beschrieben werden, die gleich effizient für die Generierung und das Anzeigen von nach innen und aussen gerichteten Sichten ist.
Wie schon erwähnt basieren diese Methode auf dem Ansatz des Image Based Rendering, mit welchem versucht wird, neue Bilder aus vorgängig berechneten oder digitalisierten zu erstellen. Dies hat folgende Vorteile:

- Benötigt bescheidene Mittel an Hardware- und Softwareressourcen, was passend für Realtime Implementationen auf Workstations und PC's ist.
- Die Kosten für das interaktive Sehen der Szene ist unabhängig von der Komplexität derselben.
- Die vorgängig erworbenen Bilder können aus einer virtuellen oder realen Umgebung stammen und könne sogar gemischt werden.

Generieren eines Light Fields oder Lumigraphs entspricht dem Einfügen von 2D Bildern in die 4D Darstellung. Umgekehrt entspricht das Generieren neuer Bilder dem Extrahieren und Darstellen von 2D Ausschnitten der 4D Funktion L. Dabei wird das neue Bild aus verschiedenen Teilen diverser Inputbilder zusammengestellt und nicht einfach zwischen benachbarten Bildern gesamthaft interpoliert. Hier sind weder Tiefeninformationen noch Bildabhängigkeiten notwendig. Die Bildgenerierung basiert also nur auf Resampling, einem einfachen linearen Prozess.

Wie schon gesagt ist dieser Ansatz unabhängig von den Parametern der Szene. Es muss also weder ein geometrisches Modell vorliegen, noch Materialeigenschaften bekannt sein, noch müssen komplexe Beleuchtungsmodelle berechnet werden. Unter der Vorbedingung, dass die Objekte statisch beleuchtet werden, können mit dieser Funktion neue Bilder, unabhängig von der Geometrie wie auch von der Komplexheit der Beleuchtung, schnell erstellt werden.

Implementation des Lumigraphs

Darstellung

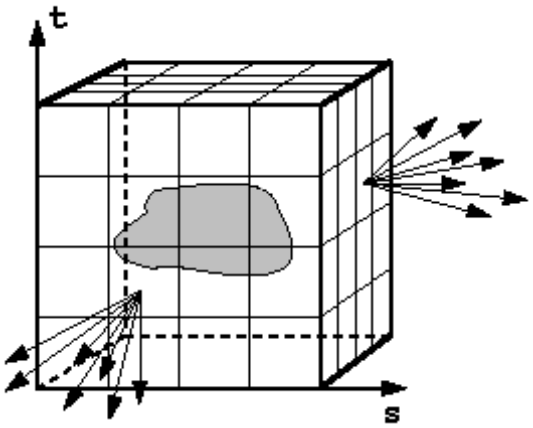
Im folgenden sollen zuerst ein Paar Gründe für die Wahl einer 4D Darstellung gegeben werden:

- Die Redundanz in der 5D Darstellung aufgrund der Konstantheit der Strahlung im freien Raum mit transparenten Luft vergrössert die Datengrösse.
- Redundanz führt auch zu einer komplizierteren Rekonstruktion der Strahlung aus den Beispielen.

Die Restriktion der Positionierung der virtuellen Kamera im freien Raum scheint zuerst ein Nachteil zu sein. Es gibt jedoch Situation, die implizit diese Voraussetzungen beinhalten:

- Geometrische Objekte sind meist gebunden.
- Bei Animationen durch Innen- oder Aussenszenen bewegt sich die Kamera im freien Raum.

Mit der Beschränkung unserer Kamerapositionen auf den Raum ausserhalb der konvexen Hülle der Szene, und der Annahme, dass Luft transparent sei, können wir uns bei der Erstellung und Darstellung des Lumigraphs auf eine Oberfläche beschränken, die sozusagen im freien Raum liegt. Aufgrund der Einfachheit wurde als Körper der Würfel gewählt. Für jeden Strahl durch einen Gitterpunkt des Würfels kann durch Zurückverfolgen bis auf die Oberfläche des Objekts, der Wert des Lumigraphs bestimmt werden (*Figur 1*).



← Figur 1

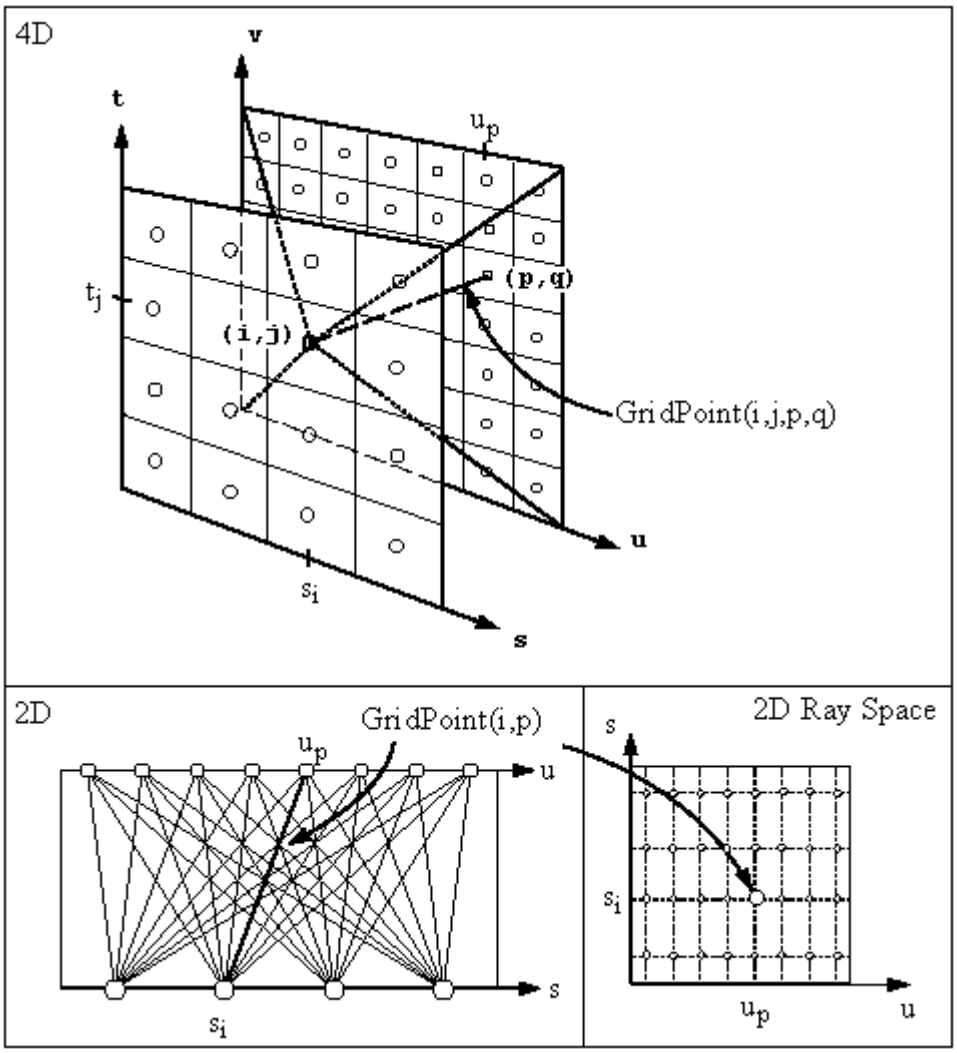
Parametrisierung

Die Parametrisierung basiert auf zwei konzentrischen Würfeln, in deren Mitte die Szenen sich befindet. Ab jetzt soll die Betrachtung auf nur ein Paar von Würfelseiten beschränkt werden. So ein Paar bildet einen Slab. Die Erweiterung auf den ganzen Raum ist analog. In beiden Würfelseiten wird ein Punkt mittels zwei Koordinaten festgelegt. Für die innere, der Szenen näheren Seite (oft Objektebene genannt) werden die Achsen u und v getauft, während auf der äusseren Seite (oft Kameraebene genannt) die Achsen s und t heissen. Somit kann jetzt ein beliebiger Strahl durch die zwei Durchstosspunkte mit den zwei Ebenen dargestellt werden. Strahl (s,t,u,v) ist also der Strahl, der die st-Ebene im Punkt (s,t) und die uv-Ebene im Punkt (u,v) durchstösst. Die uv-Ebene soll im Ursprung zentriert sein, und die st-Ebene soll sich im z = 1 Abstand davon befinden. Oft ist es hilfreich den Lumigraph mittels zwei 2D Analogien darzustellen (*Figur 2*). Man erhält den Lumigraph zum Beispiel durch Platzieren einer Kamera an jedem Punkt (s,t) mit Blickrichtung auf jeden Punkt (u,v). Das Abbilden von (x,y) auf (s,t,u,v) Koordinaten oder umgekehrt entspricht einer perspektivischen Darstellung, welche nur lineare Algebra (multiplizieren mit 3 x 3 Matrix) benötigt.

Diskretisierung

Um den Lumigraph konkret implementieren zu können, müssen die st-, wie auch die uv-Ebene diskretisiert werden. Der nicht endliche Lumigraph L muss also in einen endlichen Funktionenraum L' abgebildet werden. Dabei soll die st-Ebene in M x M Quadrate, die uv-Ebene in N x N Quadrate

aufgeteilt werden. Die Gitterpunkte auf der st-Ebene sollen mit (i,j), die Gitterpunkte auf der uv-Ebene mit (p,q) indiziert werden. Ein Strahl kann also durch den 4-dimensionalen Punkt (i,j,p,q) dargestellt werden (*Figur 2*). Der Wert des Lumigraphs (genaugenommen das RGB-Tripel) an diesem Punkt wird als $x_{i,j,p,q}$ bezeichnet.



↑ Figur 2

Es wird zu jedem Gitterpunkt eine Basisfunktion $B_{i,j,p,q}$ assoziiert, mit welcher der kontinuierliche Lumigraph L wie folgt als lineare Summe rekonstruiert wird:

$$L'(s,t,u,v) = \sum_{i=0}^{M} \sum_{j=0}^{M} \sum_{p=0}^{N} \sum_{q=0}^{N} x_{i,j,p,q} \, B_{i,j,p,q}(s,t,u,v)$$

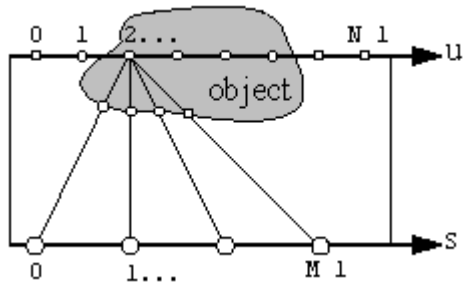
Dabei sei L' der endliche Lumigraph definiert im Raum gegeben durch die Basis B. Der Wert des diskreten Lumigraphs L' im Punkt (s,t,u,v) wird also interpoliert aus allen diskreten Punkten (i,j,p,q), gewichtet mit dem Wert der Basis im diskreten Punkt (i,j,p,q), ausgewertet für den Punkt (s,t,u,v). Ist die Basis zum Beispiel ein 4-dimensionaler Boxfilter mit dem Wert 1 im nächstliegenden diskreten Punkt (i,j,p,q) und sonst 0, so wird der Punkt(s,t,u,v) immer durch den nächstgelegenen diskreten Punkt (i,j,p,q) approximiert. Wählt man hingegen eine quadralineare Basis mit Wert 1 im diskreten Punkt und abfallend zu 0 an allen Nachbarpunkten, so wird der Wert von L' für einen Punkt aus den 16 Punkten interpoliert, welchen einen Hypercube bilden, in welchem der Punkt liegt.
Bei einem gegebenen kontinuierlichen Lumigraph L und der Wahl einer diskreten Basis B' benötigt man noch eine Projektion von L auf L' zur Bestimmung der Koeffizienten x. Diese Projektion ist gegeben durch das folgende innere Produkt:

$$x_{i,j,p,q} = < L, B'_{i,j,p,q} >$$

Dies entspricht einer Interpolation aller Werte L(s,t,u,v), welche auf L'(s,t,u,v), also auf den 4-dimensionalen Gitterpunkt (i,j,p,q) abgebildet werden, gewichtet mit der Basis $B'_{i,j,p,q}(s,t,u,v)$. Dies sieht mathematisch wie folgt aus:

$$x_{i,j,p,q} = \int L(s,t,u,v)\, B'_{i,j,p,q}(s,t,u,v)\, ds\, dt\, du\, dv$$

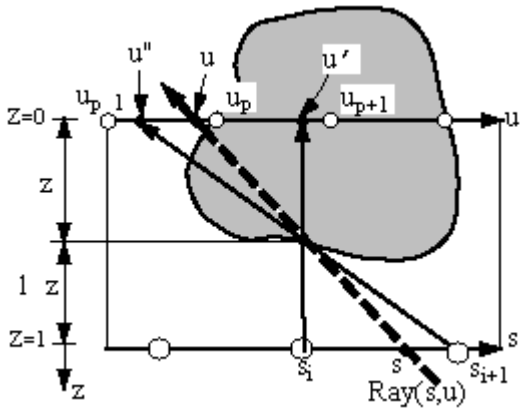
Diese Projektion kann bei der quadralinearen Basis wie folgt interpretiert werden: Eine Kamera mit bilinearer Blende wird in jedem Punkt (s_i,t_j) aufgestellt. Gerichtet auf jeden Punkt (u_p,v_q) wird ein Bild aufgenommen, das anschliessend mit einem bilinearen Filter antialiased wird.



↑ Figur 3

Eine wichtige Frage betrifft die Auflösung der st- und uv-Ebene. Die Frage hängt sehr von unserer Vorstellung ab, dass die uv-Ebene näher an der Szene liegt als die st-Ebene. Um eine gute Balance zwischen Effizienz und Qualität zu erreichen, wählen wir die Auflösung der uv-Ebene etwa entsprechend der Auflösung des gewünschten Bildes. Da eine Änderung des Gitterpunkts in der st-Ebene nur einer kleinen Änderung auf dem Objekt entspricht, kann die Auflösung der st-Ebene kleiner gewählt werden (*Figur 3*). Die Auflösung der Beispiel liegt für N zwischen 128 und 512 und für die M zwischen 16 und 64.

Liegt ein Punkt des Objekts nicht genau auf der uv-Ebene, so kann es zu Fehlern in der Darstellung kommen (*Figur 3*). Nimmt man jetzt an, dass man die Tiefe z des Punktes kennt, an dem ein Strahl das erste Mal die Oberfläche des Objekts durchstösst, so kann mit Hilfe dieses Wissens für ein gegebenes s_i der Wert u° für einen Strahl $(s_i,u°)$ bestimmt werden, der das Objekt an genau derselben Stelle durchstösst wie der originale Strahl (s,u). Für die uv-Ebene bei z = 0 und die st-Ebene bei z = 1 kann u° wie folgt bestimmt werden (*Figur 4*):



↑ Figur 4

$$u^\circ = u + (s - s_i)\frac{z}{1-z}$$

Wendet man diese Tiefenkorrektur jetzt auf u und v an, so erhält man eine korrektere Abbildung der Szenen. Die daraus neu entstehende Basisfunktion wird definiert durch vorgängiges Finden von u° und v° und anschliessendem Auswerten von B. Somit erhält man:

$$B^\circ_{i,j,p,q}(s,t,u,v) = B_{i,j,p,q}(s,t,u^\circ,v^\circ)$$

Obwohl die tiefenkorrigierte Basis jetzt komplizierter ist, ist L'(s,t,u,v) immer noch eine lineare Summe von Koeffizienten, gewichtet mit der entsprechenden Basis.

Lumigraph System

Das Erstellen des Lumigraphs läuft in verschiedenen Etappen ab. Als erstes werden Bilder der Szenen aufgenommen. Dies geschieht mit einer Handkamera. Mit Hilfe von bekannten Markierungen in den Bildern werden die Kameraposition, wie auch die Orientierung abgeschätzt. Daraus wird versucht ein einfaches approximiertes geometrisches Modell des Objekts zu erstellen, das zur Tiefenkorrektur verwendet werden soll. Anschliessend werden anhand der Pixel der Bilder die

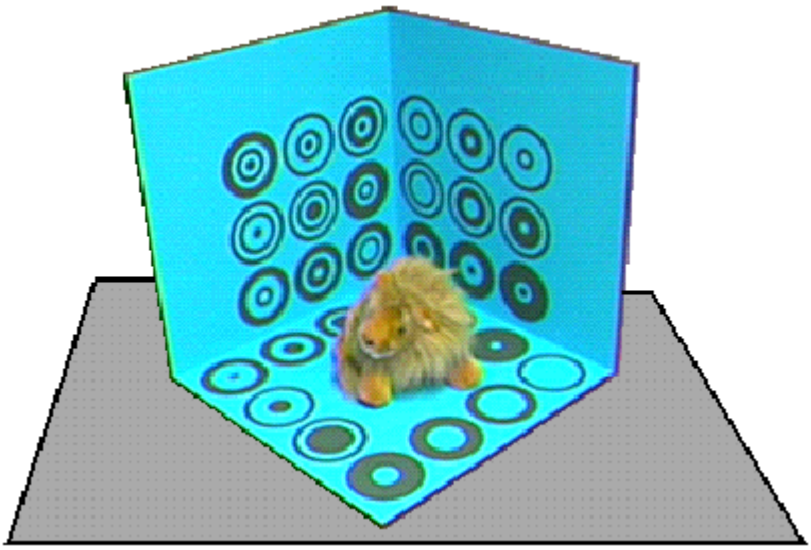
Koeffizienten des diskreten Lumigraphs L' abgeschätzt. Da der Lumigraph eine grosse Datenmenge voraussetzt, wird das ganze komprimiert. Am Schluss werden aus den gewonnen Informationen neue Bilder an einer gewünschten Position mit Orientierung erstellt. Natürlich können auch Bilder zur Erstellung des Lumigraphs mittels Rendering, Ray Tracing oder anderen Methoden gemacht werden.

Bilder aus synthetischen Szenen

Die Erstellung des Lumigraph aus synthetischen Bilder ist recht einfach: Zuerst definiert man die uv-Ebene als die Bildebene. Jetzt wird für jeden Punkt (i,j) in der st-Ebene von dem Objekt mittels einer virtuellen Kamera ein Bild generiert. Danach entspricht der Farbwert des Pixels (p,q) gerade dem Koeffizient $x_{i,j,p,q}$. Um die Qualität zu verbessern können für jeden Koeffizient mehrere Strahlen durch die gejitterte Kameraposition gelegt werden und die Pixelwerte gemittelt werden.

Bilder aus realen Szenen

Für Bilder aus realen Szenen könnte man den gleichen Ansatz machen. Dieser ist aber technisch recht aufwendig. Eine einfachere Methode ist das Aufnehmen der Szene mit einer Handkamera. Dabei muss die Szenen mit Markierungen versehen sein, damit die Position und die Orientierung der Kamera abgeleitet werden kann (*Figur 5*).



Figur 5 →

3D Oberflächen Approximation

Eine grobe Abschätzung der Oberfläche erhält man mit einem Octree-Konstruktionsalgorithmus. Dazu wird zuerst jedes Eingabe-bild mittels Blue-Screen Technik in ein binäres Objekt/Hintergrund-Bild segmentiert. Dann wird eine Octree-Repräsentation eines Würfels, der das ganzen Objekt umschliesst, initialisiert. Jetzt wird für jedes Bild der Octree auf dasselbe projiziert. Falls jetzt ein Voxel des Octrees auf ausserhalb der Objektsilhouette fällt, so wird es entfernt. Fällt ein Voxel in die Silhouette, so lässt man es. Falls aber ein Voxel auf den Rand der Silhouette fällt, so wird es für eine später Subdivision markiert. Nachdem dies für einige Bilder gemacht wurde, werden die markierten Voxel subdividiert und diese analog weiterverarbeitet. Nach einer Rekursionsstufe von etwa 4 erhält man ein genügend gutes 3D Modell des Objekts, bestehend aus einer Ansammlung von Voxels. Die externen Polygone werden zusammengefasst und bilden die Oberfläche des Objekts. Dieses Polyhedron kann auf Wunsch noch mittels eines Glättungsalgorithmus geglättet werden [Taubin95].

Erstellen des Lumigraphs

Wie der Lumigraph theoretisch erstellt wird, wurde weiter oben erwähnt. Dabei ging man aber aus, dass für das Objekte der kontinuierliche Lumigraph L vorliegt. Beim Erstellen der Bilder mit einer Handkamera ergeben sich aber zwei Probleme: Einerseits kann die Abtastdichte grosse Löcher enthalten (kann entstehen, falls das Objekt nicht von allen Richtungen aufgenommen wurde), andererseits ist die Abtastdichte nicht gleichverteilt (entsteht, falls die Kamera in gewissen Richtungen länger gehalten wird als in anderen).

Für das erste Problem gibt es von Burt [Burt88] einen Algorithmus, der auf einem hierarchischen Set von Bildern kleinerer Auflösung basiert. Dieses Set definiert eine Bilderpyramide. Jedes Bild repräsentiert das verschwommene Abbild des Bildes höherer Auflösung. Die Löcher im ursprünglichen Inputbild werden dadurch verkleinert. Diese Bilder mit tieferer Auflösung werden dazu verwendet, die Löcher in der höheren Auflösung zu füllen. Für das zweite Problem wird ein Algorithmus von Mitchell [Mitchell87] verwendet. Dieser Algorithmus behebt das Problem, dass Bereiche mit höherer Samplingdichte mehr Gewichtung bekommen. Dazu berechnet der Algorithmus Mittelwerte in Bereichen kleinerer Regionen. Der endgültige Wert des Pixels wird dann berechnet durch Mittelung der Werte an diesem Punkt in den verschiedenen Regionen. Da der Durchschnitt

nicht mit der Anzahl Samples gewichtet wird, die in diese Regionen fallen, beeinflusst die Nicht-Gleichverteilung der Samples das Resultat nicht.
Der Algorithmus zum Erstellen des Lumigraphs arbeitet also in drei Phasen: In der Splatting-Phase werden die Samples dazu verwendet, das Integral zur Berechnung der $x_{i,j,p,q}$, sowie entsprechender Gewichte $w_{i,j,p,q}$ zu berechnen. Für den 1-dimensionalen Fall und der höchsten Auflösung (Auflösung r = 0, Auflösung r > 0: tiefere Auflösung) sieht die Berechnung mittels Monte-Carlo Integration [Powell66] wie folgt aus:

$$w_i^0 = \sum_k B'_i(s_k)$$

$$x_i^0 = \frac{1}{w_i^0} \sum_k B'_k(s_k) L(s_k)$$

In der zweiten Phase, der Pull-Phase, werden dann die Bilder tieferer Auflösung berechnet:

$$w_i^{r+1} = \sum_k h'_{k-2i} \min(w_k^r, 1)$$

$$x_i^{r+1} = \frac{1}{w_i^{r+1}} \sum_k h'_{k-2i} \min(w_k^r, 1) x_k^r$$

Die letzte Phase, die Push-Phase, dient dazu, die Information aus tieferen Auflösungen in die höheren zu bringen. Hier werden die Regionen in der höheren Auflösung, deren Gewichte zu klein sind, mit den Informationen aus den tieferen Auflösungen aufgefüllt. Um diese Informationen einblenden zu können, muss die Funktion aus der tieferen Auflösung in der Basis der höheren Auflösung ausgedrückt werden. Dies geschieht mittels der folgenden temporären Variablen:

$$tw_i^r = \sum_k h_{i-2k} \min(w_k^{r+1}, 1)$$

$$tx_i^r = \frac{1}{tw_i^r} \sum_k h_{i-2k} \min(w_k^{r+1}, 1) x_k^{r+1}$$

Diese temporären Variablen können jetzt mit den schon vorhandenen Werten der höheren Auflösung überblendet werden:

$$x_i^r = tx_i^r (1 - w_i^r) + w_i^r x_i^r$$

$$w_i^r = tw_i^r (1 - w_i^r) + w_i^r$$

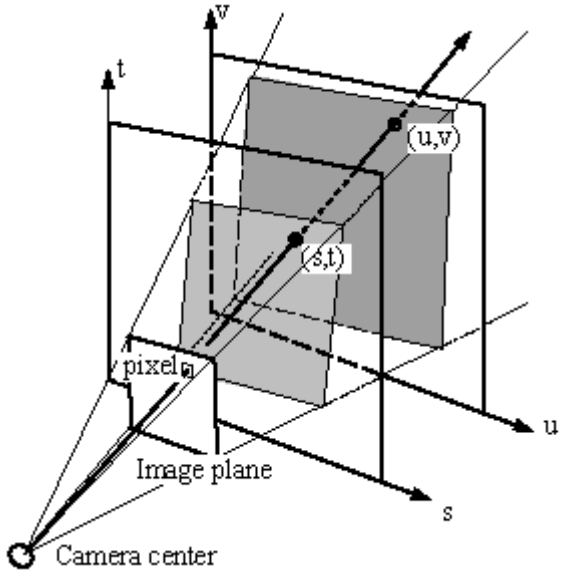
Der ganze Algorithmus muss leicht angepasst werden, falls mit Tiefenkorrektur gerechnet werden soll. So muss in der Splatt-Phase für jeden Strahl $L(s_k,t_k,u_k,v_k)$ die u und v Werte korrigiert werden. Genauso müssen in der Push- und Pull-Phase anstatt einfach die Koeffizienten mit der Basisfunktion und benachbarten Indizes zu kombinieren, die tiefenkorrigierten Indizes verwendet werden.

Komprimierung
Ein typisches Beispiel mit einem 32 x 32 Sampling in der st-Ebene und einem 256 x 256 Sampling in der uv-Ebene benötigt beim Speichern aller sechs Slabs mit je 24 Bit pro Pixel 1,125 GByte. Aufgrund von sehr viel Redundanz in den Bildern wird erwartet, dass Komprimierverhältnisse von bis zu 200:1 möglich sind, ohne dabei grosse Qualtitätsverluste zu erhalten. Dies würde die Grösse dieses Lumigraphs auf unter 6 MByte drücken.

Rekonstruktion von Bildern ...
Gegeben durch eine gewünschte virtuelle Kamera (Position, Orientierung, Auflösung) färbt die Rekonstruktionsphase jeden Pixel des Outputbildes so, als ob eine reale Kamera in dieser Situation ein Bild des Objekts machen würde.

... mittels Ray Tracing

Für jeden Strahl durch einen Pixel des Outputbildes wird anhand der berechneten (s,t,u,v) Koordinaten die naheliegenden Gitterpunkte berechnet und deren Werte entsprechend der Filterfunktion interpoliert (*Figur 6*). Bei der tiefenkorrigierten Basisfunktion werden vor der Interpolation die korrigierten (u°,v°) Koordinaten berechnet. Dies kann eventuell unter Zuhilfenahme von Hardware realisiert werden.



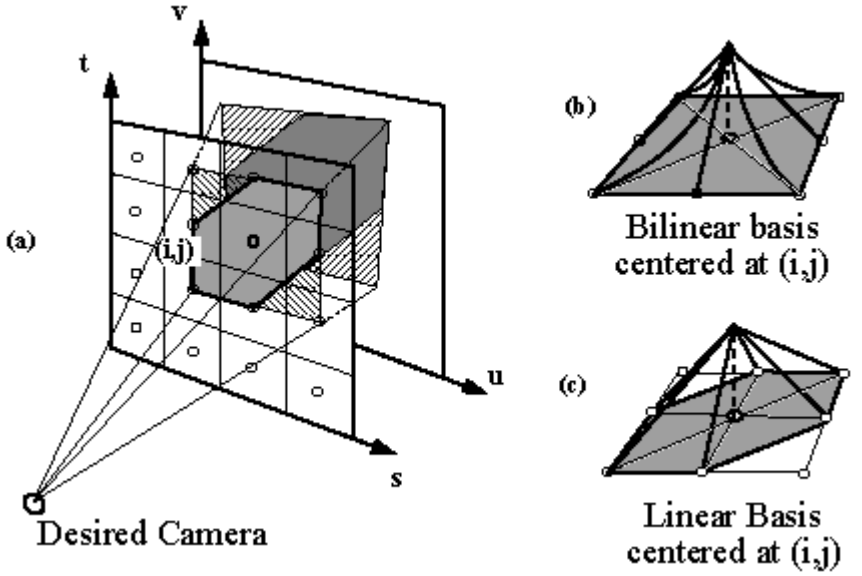
← Figur 6

... mittels Texture Mapping

Die Kosten für das Behandeln jedes Pixels separat kann verhindert werden unter Verwendung von Texture Mapping. Die st-Ebene selbst ist ja belegt mit Texturen definiert durch $tex_{i,j}(u_p,v_q) = x_{i,j,p,q}$, d.h. der Texel in der Texture (i,j) an der Position (p,q) entspricht gerade $x_{i,j,p,q}$.

Die Darstellung im Falle der konstanten Basis läuft wie folgt ab: Das Set von Strahlen durch die st-Ebene, welche auf die nächstgelegene Koordinate (i,j) abgebildet werden, schneiden aus der uv-Ebene eine Rechteck mit Koordinaten $(u,v)_0$, $(u,v)_1$, $(u,v)_2$ und $(u,v)_3$ aus. Diese Koordinaten liegen im Sample (i,j) und bilden die Texture, die als ganzes für alle Strahlen, welche auf die Koordinaten (i,j) in der st-Ebene abgebildet werden, gemappt werden kann. Jetzt können einfach für jede (i,j) Koordinate die Texturkoordinaten bestimmt werden. Anschliessend werden alle Texturen, die benötigt werden um das komplette Outputbild zu zeichnen, gemappt. Dabei wird natürlich geachtet, dass Texturen nicht mehrfach gezeichnet werden. Da bei konstanter Basis sich die Texturen nicht überschneiden, sondern aneinandergrenzen, ist kein Zusatzaufwand nötig.

Figur 7 →



Beim quadralinearen Filter wird ein Pixelwert aus 16 4D Gitterpunkten interpoliert. Das hat zur Folge, dass sich die Texturen überlappen (*Figur 7*). Der quadralineare Filter kann durch bilineares Filtern in der st- und uv-Ebene erreicht werden. Die bilineare Interpolation in der uv-Ebene kann oft durch die Hardware erledigt werden. In der st-Ebene ist dies nicht möglich. Hier kann die bilineare Pyramide (*Figur 7b*) durch eine lineare Pyramide approximiert werden (*Figur 7c*). Durch Zuweisung von α-Werten zu jeder Ecke ($\alpha = 1$ in der Mitte, $\alpha = 0$ aussen) und Benützung von α-Blending kann die quadralineare Interpolation des Bildes durch eine linear-bilineare Interpolation approximiert werden. Leider summiert so eine Basisfunktion nicht zu Eins, was zu Artefakten führen kann. Durch Verwendung einer hexagonalen Region können diese aber verhindert werden (*Figur 7c*). Die so entstehende lineare Interpolation von α-Werten mit einer bilinearen Filterung der Texturen resultiert in einer Approximation des quadralinearen Filters.

Texture Mapping ist dank Hardware schnell. Der ganze Prozess benötigt höchstens $6M^2$ Texturen, die zu mappen sind und aus α-geblendeten Dreiecken bestehen. Natürlich können die (u,v) Koordinaten tiefenkorrigiert werden, bevor sie gemappt werden.

Resultate

Über die Zeit für das Erstellen des Lumigraphs aus synthetischen Bildern sind keine Daten im Paper vorhanden. Je nach Aufwand der betrieben wird, kann dies aber schon mehrere Tage in Anspruch nehmen. Die Verarbeitung der digitalisierten Bilder zu einem Lumigraph dauerte aber weniger als einen Tag auf einer SGI Indy Workstation. Ist ein Lumigraph einmal vorhanden, so können neue Bilder mit einer Auflösung von 450 x 450 Pixel mittels Ray Tracing auf einer SGI Reality Engine in wenigen Sekunden generiert werden. Unter Verwendung von Texture Mapping konnten mehrere Bilder in einer Sekunde generiert werden.

Implementation des Light Fields

Darstellung

Die Darstellung ist analog zum Lumigraph abgesehen von folgenden zwei Unterschieden: Der erste Unterschied ist eine vertauschte Benennung der Ebenen, also von keiner grösseren Bedeutung. Der zweite Unterschied liegt in der freien Wahl der Positionierung der Ebenen im Light Field. So kann zum Beispiel die eine Ebene im Unendlichen liegen, was die Möglichkeit eröffnet, Light Fields aus orthogonalen Aufnahmen von Bildern zu erstellen. Falls die Berechnungen in homogenen Koordinaten getätigt werden, so können beide Fälle ohne Zusatzaufwand behandelt werden. Auch hier ist es oft hilfreich das Light Field im 2D Linien Raum zu betrachten.

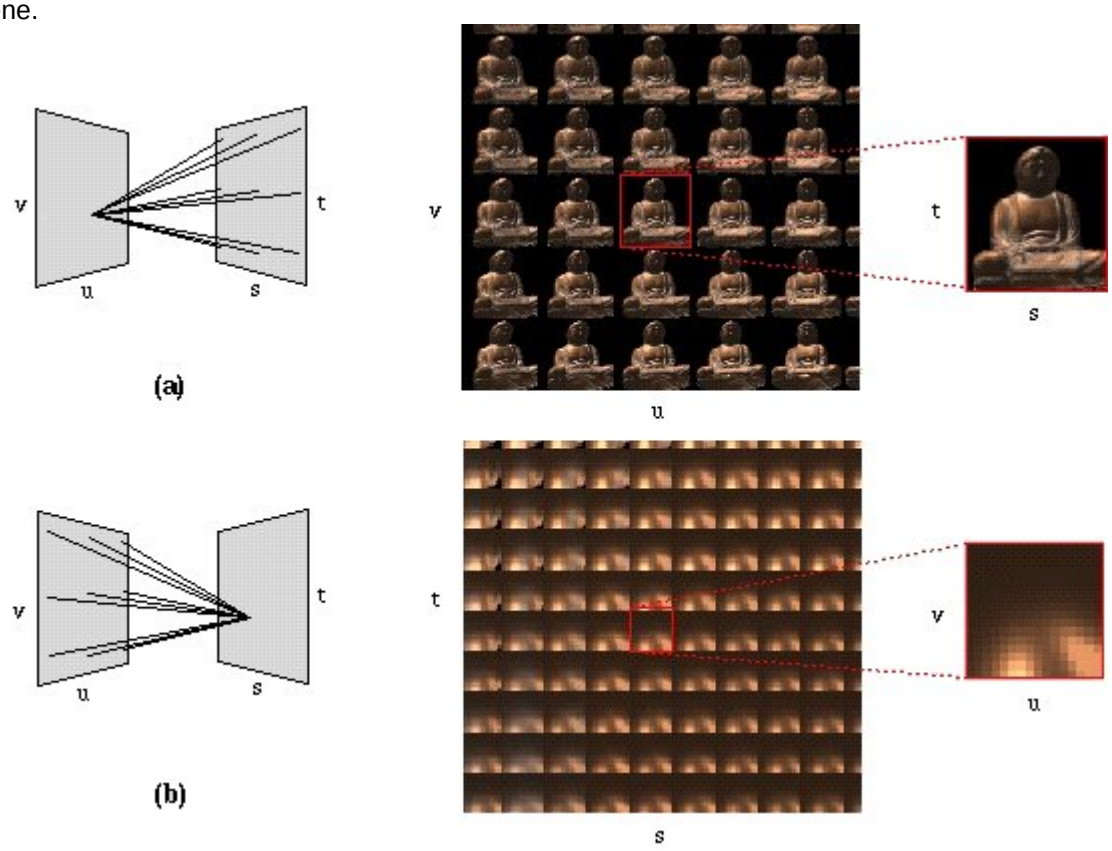
Das Light Field Rendering System

Wie beim Lumigraph könnte man auch das Light Field aus einer Kollektion von Bildern, die zum Beispiel mittels eines Ray Tracers erstellt wurden, generieren. Analog könnte man eine reale Szene mit einem Radiometer (Strahlungsmesser) abtasten und so die entsprechenden Bilder erstellen. Dies ist aber zu ungenau und viel zu aufwendig.

Erstellen aus synthetischen Bildern
Dies läuft analog zu den Erklärungen im Lumigraph ab. Die Visualisierung des Light Fields kann auf zwei Arten geschehen: Einerseits als uv-Array von st-Bildern (vorallem bei nach innen gerichteten Sichten $\rightarrow$ Objektansichten, bei denen man von jedem uv Punkt aus das Objekt sieht) oder andererseits als st-Array von uv-Bildern (bei nach aussen gerichteten Sichten $\rightarrow$ Ansicht von Räumen und Umgebungen, bei der man von jedem st-Punkt aus die Umgebung sieht oder die Strahlung des st-Punktes nach allen uv-Punkten) (*Figur 8*).
Da die Darstellung infolge von hohen Frequenzen im Light Field zu Aliasing führt, muss im 4D Raum gefiltert werden. Die Filterung entspricht dabei der quadralinearen Filterung des Lumigraphs. Die dort erwähnte Interpretation einer Kamera mit bilinearer Blende und anschliessendem bilinearen Filtern über dem Bild ist hier passend. Durch Verwenden einer Blende mit einem Durchmesser entsprechend dem Abstand der Gitterpunkte auf der Kameraebene (uv-Ebene) ersetzt den zu Aliasing führenden Sprung zwischen zwei Bildern von benachbarten Gitterpunkten durch eine Sequenz von verschwommenen Bildern. Wird der Durchmesser des Filters zu klein gewählt, so äussert sich dies in Sprüngen beim interaktiven Betrachten, wird er zu gross gewählt, so erscheinen die Bilder extrem verschwommen.

Erstellen aus digitalisierten Bildern
Das Erstellen eines Light Fields aus einer realen Szene ist ein schwieriges Problem. Die Anzahl Bilder ist gross und eine automatische oder computerunterstützte Generierung drängt sich auf. Zudem muss geachtet werden, dass zum Beispiel keine unerwünschten Schatten infolge der Kamera in der statisch beleuchteten Szene entstehen. Im weiteren sollen nur Ansichten rund um ein Objekt betrachtet werden, da diese Problem einfacher ist als die Darstellung von Aussenansichten. Ebenfalls aus Gründen der Einfachheit wird eine Apparatur verwendet, in der die Kamera sich nur auf einer Ebene (uv-Ebene) bewegen kann. Eine Gesamtabtastung führt dann zu einem Würfel, welche dem Light Field Modell näher ist als eine Abtastung auf einer Kugel. Die Kamera kann sich also auf der uv-Ebene horizontal und vertikal bewegen und dabei geneigt und geschwenkt werden. So wird in vier

mal 90° Abständen das Objekt aufgenommen. Auf die Abtastung von oben und unten wird verzichtet. Da sich beim Schwenken und Neigen der Kamera die Objektebene (st-Ebene) ebenfalls bewegt (sie bleibt ja immer normal zur Kamerarichtung), ist anzunehmen, dass dadurch Verzerrungen entstehen, da das Light Field ja auch von einer parallelen Ausrichtung der uv- und st-Ebene ausgeht. Durch das sorgfältige quadralineare Filtern sind aber in den Beispielen die erwarteten Fehler nicht aufgetreten. Um eine der Translation entsprechenden Blende zu bekommen, wurde der Ansatz gemacht, einige angrenzende Bilder zu mitteln. Das so aufgenommene Set von Bilder entspricht dem Light Field der Szene.



↑ Figur 8

Komprimierung

Light Field Arrays können wie auch im Lumigraph enorme Grösse erreichen. Um Generierung und Darstellung von Light Fields praktisch zu machen, müssen sie komprimiert werden. Folgende Charakteristiken des Light Fields leiteten die Wahl einer Komprimiermethode:

- Datenredundanz im Light Field
- Erwünschter Random Access auf Samples der uv-Ebene
- Möglichkeit des Verwendens asymmetrischer Verfahren (langsam beim Packen, schnell beim Entpacken)
- Wunsch nach reiner Softwareimplementierung

Dies führte zu einer Zwei-Schritt Pipeline: Zuerst wird eine Fix-Rate Vector Quantization Komprimierung angewendet. Auf diese folgt eine Lempel-Ziv Entropie Kodierung.
Die Vector Quantization erstellt in einer Lernphase mit einigen Bildern für verschiedene Bildausschnitte ein Muster, welches diesem Ausschnitt möglichst gleicht. Diese Muster werden mittels Indizes in ein Codebuch geschrieben. Bei der Kodierung wird jetzt jedes Bild in Samples gleicher Grösse unterteilt, für jedes Sample im Codebuch das passendste Muster gesucht und somit das gesamte Bild in Form von Indizes gespeichert. In den Beispielen wurde auf diese Art eine Komprimierung von 24:1 erreicht. Durch Komprimieren des Codebuchs und der Indizes mittels Lempel-Ziv Kodierung wurde eine zusätzliche Komprimierung von 5:1 erreicht. Dies führte zu einer Gesamtkomprimierung von 120:1.

Dekomprimierung
Diese erfolgt ebenfalls in zwei Schritten: Beim Laden des Light Fields wird dieses Lempel-Ziv dekodiert. Der Output ist das Codebuch und die Indizes. Der zweite Schritt der Dekomprimierung erfolgt dann während des interaktiven Betrachtens der Szene: Anhand der neuen Kameraposition und der Blickrichtung werden die entsprechenden (s,t,u,v) Koordinaten berechnet. Eine darauffolgende Berechnung resultiert in der Adresse des Samples in den Indizes. Der Index wird ausgelesen worauf das Muster im Codebuch bestimmt wird. Auf diese Weise wird Schritt für Schritt das Outputbild generiert.

Rekonstruktion von Bildern
Das Erzeugen neuer Bilder ist ebenfalls analog zu dem Verfahren beim Lumigraph, wobei noch erwähnt werden sollte, dass im ganzen Light Field System keine Tiefenkorrektur angewendet wird. Das Erzeugen von Bildern mittels Ray Tracing oder mittels Texture Mapping wird hier im Light Field einfach durch den zweiten Schritt der Dekomprimierung ergänzt.

Resultate

Im Paper über das Light Field Rendering sind viele Daten von mehreren Beispielen vorhanden. Um nicht alle Tabellen zu kopieren, will ich ein Beispiel herausgreifen. Da vom Lion Beispiel die meisten Daten vorhanden sind, habe ich mich für dieses Beispiel entschieden. Das Lion Beispiel basiert auf digitalisierten Bildern und wurde mit der erwähnten Apparatur erstellt. Die folgende Tabelle enthält eine Zusammenfassung der Resultate und Daten dieses Beispiels.

Allgemein	
Anzahl Slabs (Basisanordnungen der uv-Ebene)	4 (von allen Seiten ausser von oben und unten)
Bilder pro Slab	32 x 16
Totale Anzahl Bilder	2048
Pixels pro Bild	256^2
Rohgrösse	402.7 MByte
Filterung	quadralinear
Erstellung des Light Fields	
Zeit pro Bild	3 s
Totale Erstellungszeit	4 h
Vector Quantization	
Rohgrösse	402.7 MByte
Bruchteil im Trainingsset	3 %
Anzahl Codewörter	16348
Codebuchgrösse	0.8 MByte
Indexarraygrösse	16.8 MByte
Totale Grösse	17.6 MByte
Komprimierverhältnis	23:1
Entropy Coding	
Codebuchgrösse	0.6 MByte
Indexarraygrösse	2.8 MByte
Totale Grösse	3.4 MByte
Komprimierverhältnis	5:1
Totale Komprimierung	118:1
Kompressionsleistung	
Trainingszeit	4 h
Kodierzeit	8 min
Darstellung	
Koordinatenberechnung	13 ms
Sample Extrahierung	214 ms
Overhead (Maus, Kopieren in Framebuffer, ...)	3 ms
Total	230 ms

Die Darstellungszeiten beziehen sich auf eine reine Softwareimplementation mit einer Bildauflösung von 192 x 192 Pixel auf einem 250 MHz MIPS 4400 Prozessor.

Diskussion und zukünftige Forschungsmöglichkeiten

Diese Methoden eröffnen die Möglichkeit komplexe Szenen und realistische Beleuchtung real widerzuspiegeln und dies in einer hohen Geschwindigkeit.

Verglichen mit tiefenbasierten oder abhängigkeitsbasierten Sichtinterpolationsmethoden benötigt das Light Field wie auch der Lumigraph eine grössere Anzahl Bilder. Infolge von Komprimierverhältnissen von über 100:1 und der simpleren Erstellung der Bilder wird dieser Nachteil aber wettgemacht. Die Einfachheit des Algorithmus, die hohe Geschwindigkeit, die Nichtnotwendigkeit des Findens und Speicherns einer 3D Struktur und die Unabhängigkeit von der Komplexheit der Szene und deren Beleuchtung führen zu einem Vorteil für die hier behandelten Methoden gegenüber den anderen Image Based Rendering Verfahren.
Es darf dabei nicht vergessen werden, dass die Vorteile dieser Methoden auf gewissen Einschränkungen, wie statische Szene, fixe Beleuchtung und der Einschränkung der Positionierung der virtuellen Kamera im freien Raum, basieren. Zudem muss die Sampling Dichte hoch sein, um die Verschwommenheit der Bilder einzugrenzen. Es kann auch sein, dass die Einschränkung des Betrachters auf den freien Raum nicht erwünscht ist. Dieses Problem kann reduziert werden durch Aufteilung einer komplexen Szene in verschiedene konvexe Hüllen und durch Erstellen eines Light Fields für jede dieser Regionen. Bei der Darstellung werden diese Light Fields passend überlagert. Erweitert man das Light Field mit einem Tiefenwert, so müssen die Regionen nicht mehr konvex sein. Eine weitere Erweiterung um Informationen über die Oberflächennormalen und -beschaffenheit könnte die Limitation auf statische Beleuchtung aufheben.
Betreffend der Komprimierung ist noch zu erwähnen, dass erst Komprimierverhältnisse von über 200:1 zu unangenehmen Artefakten in Folge von zu schlechter Approximation der Samples führen.

Eine Möglichkeit einer zukünftigen Forschung könnte das Entwickeln einer Apparatur von mehreren auf einem Gitter aufgestellten Kameras zur parallelen Aufnahme eines realen Light Fields auf einem Parallelrechner sein.

Meine persönliche Meinung

Meine persönliche Meinung schliesst hauptsächlich an die obige Diskussion an. Unter den gegeben Voraussetzung sind die beiden Methoden schnelle Verfahren zur Erstellung von realistischen Bildern in Realtime. Betreffend Realtime ist aber zu sagen, dass die Resultate im richtigen Verhältnis gesehen werden müssen. So sind einerseits die Bilder nur 192 x 192 Pixel gross (Lion Beispiel). Eine Darstellungszeit von 230 ms pro Bild ergibt etwa 4 Bilder pro Sekunde, was in einer Animation noch lange nicht flüssig erscheint. Zudem wurde die Zeiten auf einem 250 MHz MIPS 4400 Prozessor gemessen. Das Ziel von einer Realtime-Animation auf dem PC ist also noch nicht erreicht. Das Nichtverwenden von Tiefeninformationen ist ja ein Vorteil dieser Methoden, doch kann mir vorstellen, dass es Situationen gibt, in denen ein Benutzer vielleicht gerade diese Informationen haben will.
Wie auch immer, die hier beschriebenen Methoden sind wirklich ein Schritt in eine realistische Darstellung in Realtime, welche in Zukunft sicher noch weitere Beachtung verdienen.

Referenzen

[Adelson91] Adelson, E.H., Bergen, J.R., "The Plenoptic Function and the Elements of Early Vision", In *Computation Models of Visual Processing*, M. Landy and J.A. Movshon, eds., MIT Press, Cambridge, 1991

[Burt88] Burt, P.J., "Moment images, polynomial fit filters, and the problem of surface interpolation", In *Proceedings of Computer Vision and Pattern Recognition*, IEEE Computer Society Press, June 1988, pp. 144-152

[Chen95] Chen, S.E., "QuickTime VR - An Image Based Appraoch to Virtual Environment Navigation", In *Proc. SIGGRAPH '95* (Los Angeles, CA, August 6-11, 1995), In *Computer Graphics* Proceedings, Annual Conference Series, ACM SIGGRAPH, 1995, pp. 29-38

[Mitchell87] Mitchell, D.P., "Generating antialiased images at low sampling densities", In *Computer Graphics 21*, 4, 1987, pp. 65-72

[Powell66] Powell, M.J.D., and Swann, J., "Weighted uniform sampling - a monte carlo technique for reducing variance", In J. Inst. Maths Applics, 2, 1966, pp. 228-236

[Taubin95] Taubin, G., "A signal processing approach to fair surface design", In *Computer Graphics, Annual Conference Series*, 1995, pp. 351-358