Ausarbeitung zu den Systementwicklungsprojekten

Image-Based Rendering using State of the Art Graphics Hardware

und

Image-Based Editing

Bearbeitung: Moulaye Ould Mehdi
Wolfgang Wirth

Betreuer: Dr. Peter Kipfer
Lehrstuhl 15
Fakultät für Informatik
Technische Universität München

1. Einführung

Image-Based Rendering (IBR) ist eine in den letzten Jahren erforschte Alternative der Erzeugung zweidimensionaler Ansichten gegenüber geometriebasierten Methoden. Dabei werden ausgehend von zweidimensionalen Bilddaten dreidimensionale Ansichten der Szene rekonstruiert, wobei ein Wissen über die Geometrie der Szene nicht notwendig ist. Das Verfahren basiert auf Farbbildern sowie Zusatzinformationen (wie z.B. Tiefenwerten) der abgebildeten Szene und ermöglicht die Verwendung von Bildern natürlicher Objekte wie Farbphotographien, so daß der bei geometriebasierten Verfahren naturbedingt begrenzte Realismus der Darstellung überwunden werden kann. Insbesondere bei der naturgetreuen Darstellung von Objekten, bei der einfache 2D Aufnahmen nicht ausreichend sind, kann Image-Based Rendering gewinnbringend eingesetzt werden. Anwendungen für das Verfahren sind in Bereichen sinnvoll, in denen eine exakte Wiedergabe der Farbinformationen und Geometrie notwendig oder erwünscht ist.

2. Image-Based-Rendering

2.1 Light Field Rendering

Auf der SIGGRAPH 1996 stellten Marc Levoy und Pat Hanrahan das Light Field Rendering als neue Methode des Image-Based Rendering vor. Dabei werden sowohl Aufnahmen natürlicher Szenen als auch digital erzeugte Bilddaten ohne Tiefenwerte zu neuen Ansichten kombiniert. Notwendig sind allerdings Informationen über die relative Lage der Kameras zueinander, was bei natürlichen Szenen durch eine protokollierte Einrichtung und Photographie der Szene erreicht werden kann und beim Rendern von digitalen Bildern ohne Zusatzaufwand ermittelt werden kann.

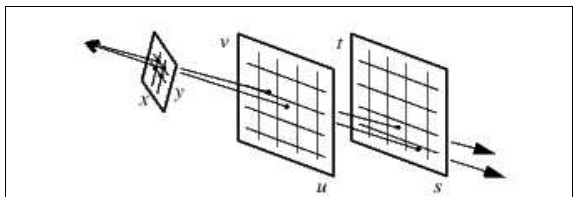

Abb. 1: Ansicht zweier Strahlen ausgehend von der virtuellen Kamera durch die beiden Ebenen u,v und s,t (Bild: Light Field Rendering, Marc Levoy und Pat Hanrahan)

Ähnlich der plenoptischen Funktion werden die in der Szene abgestrahlten Farbwerte als Funktion von Position und Richtung aufgefasst. Diese 5D Funktion kann jedoch auf 4D reduziert werden, da im freien Raum - unter der Annahme, daß das Medium Luft vollständig transparent ist - sich Intensität und Farbwerte entlang eines Strahles nicht verändern. Zur Parametrisierung der Strahlen werden 2 Ebenen verwendet, so daß ein einzelner Strahl durch die Schnitte der beiden Ebenen beschrieben wird. Die u,v-Ebene ist dabei die Kameraebene, von der aus die Szene aufgenommen wurde. Alle Kameras sind auf die s,t-Ebene ausgerichtet, die auch im Unendlichen positioniert werden kann, so daß die Strahlen als Position und Richtung aufgefaßt werden können.
Die Aufnahmen einzelner Kameras werden bei der Erzeugung des Light Field als 2D Schichtbild in den 4D Datensatz eingefügt und bei der Darstellung einer neuen Ansicht wieder als 2D Schichtbild ausgelesen. Da das Light Field Rendering jedoch auf eine hohe Abtastrate der Szene angewiesen ist, sind große Datenmengen erforderlich, die von Marc Levoy und Pat Hanrahan durch Kompression ähnlich des Algorithmus von Ziv und Lempel durch die Ausnutzung der hohen Redundanz der Daten verringert werden konnten.

2.2 The Lumigraph

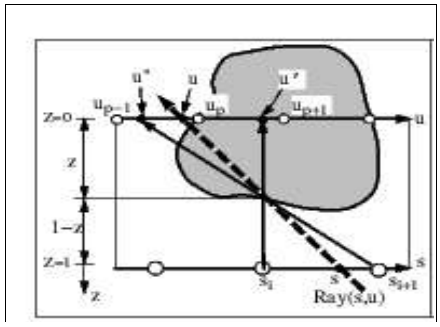

Abb. 2: Tiefenkorrektur für den Strahl durch s,u. Die Punkte u' und u'' schneiden die selbe Geometrie. (Bild: The Lumigraph)

Ähnlich dem Light Field Rendering ist auch das Prinzip des Lumigraph von Steven Gortler et al, bei dem jedoch Näherungswerte der Geometrie verwendet werden, um eine höhere Darstellungsqualität bei geringerer Abtastungsrate der Szene zu erreichen. Das Verfahren ist wiederum sowohl auf natürliche als auch künstliche Szenen anwendbar und beschreibt den Lichtfluss an allen Positionen in allen Richtungen.
Unter der Vorraussetzung, daß das Licht nach dem Abstrahlen von einem Objekt durch ein transparentes Medium fließt ist wiederum die Reduktion der plenoptischen 5D-Funktion auf 4D möglich. Zur Parametrisierung der Strahlen wird ein Würfel verwendet, wobei die Bedeutung der s,t- und u,v Ebenen (als gegenüberliegende Seitenflächen des Würfels) genau entgegengesetzt der Bedeutung beim Light Field ist.

Bei der Rekonstruktion einer neuen Ansicht für eine virtuelle Kamera wird, nachdem für alle Bildstrahlen die Schnittkoordinaten mit den beiden Ebenen ermittelt wurden, die Tiefe,bei der der Strahl als erstes auf ein Objekt trifft, ausgelesen oder berechnet. Mit diesem Wert können für die der virtuellen Kamera am nächsten gelegenen bildgebenden Kameras besser geeignete Koordinaten in der u,v-Ebene gefunden werden, bei denen die Farbwerte der Farbe des Objekts an der geschnittenen Stelle entsprechen.

3. Implementierung

3.1 Hardwarebeschleunigtes Rendern eines Lumigraphs

3.1.1 Triangulierung

Die direkte Verwendung der Pixel der Tiefenkarten ist in zweierlei Hinsicht unvorteilhaft, da zum einen die notwendigen Berechnungen für alle Tiefenpunkte individuell durchgeführt werden müssen und zudem bei der Verwendung von punktförmigen Objekten eine kontinuierliche Oberflächendarstellung nicht garantiert werden kann.

Besser geeignet ist die Verwendung einer adaptiven Triangulierung, bei der für jede Tiefenkarte ein Triangle Strip erzeugt wird. Für die Triangulierung wurde das in [5] beschriebene Verfahren ausgewählt, da es die Erzeugung generalisierter sequentieller Triangle Strips ermöglicht.

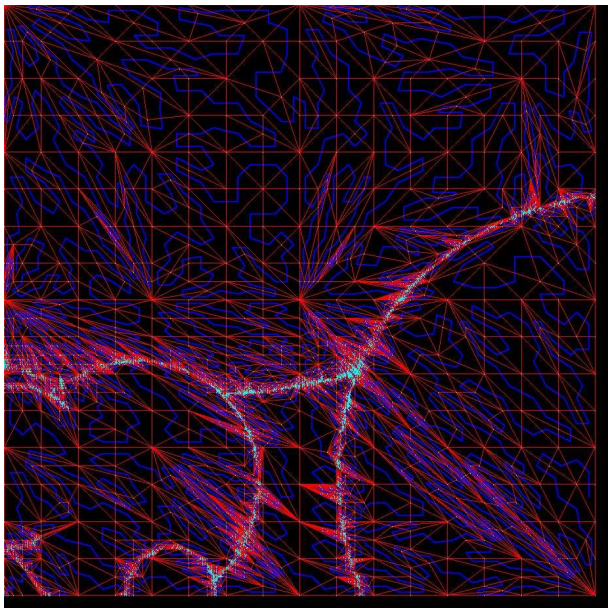

Abb. 3: Beispielhafte Unterteilung von Eingabedaten: Rot sind die Dreieckskanten in 2D. Blau ist die Space-Filling-Curve die den Weg durch die Dreiecke markiert. Gut sichtbar sind auch die, sich durch die adaptive Unterteilung ergebenden, Unterschiede zwischen der Bereichen mit geringem und den harten Kanten mit grossem Tiefenunterschied.

Die Zerlegung von 2D Eingabedaten in einen generalisierten sequentiellen Triangle Strip liegt genau dann vor, wenn die erzeugte Triangulierung einen Hamiltonpfad enthält, so daß jedes Dreieck exakt einmal auf diesem Pfad durch die gesamte Triangulierung vorkommt. In jede solche Kette von Dreiecken, die einen Hamiltonpfad enthält, kann eine weitere Kette eingefügt werden, um die Abtastung der Eingabedaten zu verfeinern. Dies entspricht einer Unterteilung eines Dreiecks in mehrere Teildreicke.

Um einerseits eine adaptive Triangulierung zu erhalten und andererseits den Hamiltonpfad nicht zu unterbrechen, muß die Unterteilung von Dreiecken abhängig von der Umgebung (Auswahl der zu unterteilenden Kanten, Eingangskante/Ausgangskante) erfolgen. Dafür gibt es mehrere Musterunterteilungen, die in der jeweiligen Situation beide Bedingungen erfüllen. Unterteilt werden die Dreiecke wenn mindestens eine von zwei Vorraussetzungen erfüllt sind: Zum einen kann die Länge einer Kante einen gewissen Schwellwert überschreiten oder zwischen zwei Vertices einer Kante besteht ein Tiefenunterschied, der den festgesetzten oder prozentualen Schwellwert überschreitet. Desweiteren können Dreiecke durch Unterteilungen ihrer Nachbarn gezwungen werden sich selbst weiter zu teilen.

Aufgrund der Tatsache daß der bei der Triangulierung erreichte Triangle Strip ein generalisierter Triangle Strip ist, ist die alternierende Folge von A- und B-Kanten wie es von der OpenGL Bibliothek gefordert wird leider nicht garantiert. Um diesen Effekt auszugleichen können einzelne Vertices wiederholt werden um eine Richtungsänderung zu erreichen.

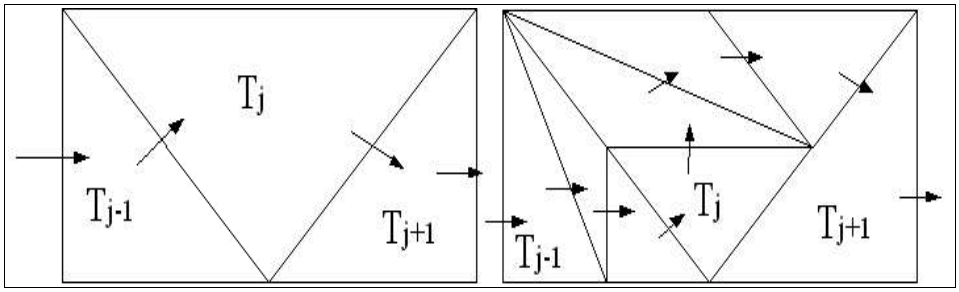

Abb. 3: Beispielhafte Unterteilung von Eingabedaten: Rot sind die Dreieckskanten in 2D. Blau ist die Space-Filling-Curve die den Weg durch die Dreiecke markiert. Gut sichtbar sind auch die, sich durch die adaptive Unterteilung ergebenden, Unterschiede zwischen der Bereichen mit geringem und den harten Kanten mit grossem Tiefenunterschied.

3.1.2 Darstellung einer rekonstruierten Ansicht

Bei jedem anzuzeigenden Bild wird zuerst die Position der virtuellen Kamera C_v in der Kameraebene bestimmt. Anhand der ermittelten Koordinaten können die für den aktuellen Frame am besten geeigneten Triangulierungen der Tiefenkarten sowie die zugehörigen Farbbilder ausgewählt werden. Das Kriterium hierbei ist die Distanz der zwischen C_v und den Kameras, mit denen die Szene ursprünglich aufgenommen wurde.

Um die ursprüngliche Tiefe der Szene im Betrachtungsbereich von C_v zu rekonstruieren werden die ausgewählten Triangle Strips ausgehend von C_v reprojeziert und damit die benötigten Informationen für die spätere Tiefenkorrektur bestimmt.

Die Farbinformationen der ausgewählten Kameras werden der Grafikhardware als Textur zur Verfügung gestellt. Für jedes darzustellende Fragment werden anhand der Tiefeninformationen der Szene die geeigneten Texturkoordinaten ermittelt und die derart erhaltenen Farbwerte zu einem Ausgabewert kombiniert. Die Gewichte für hierfür ergeben sich dabei aus der relativen Lage zur virtuellen Kamera.

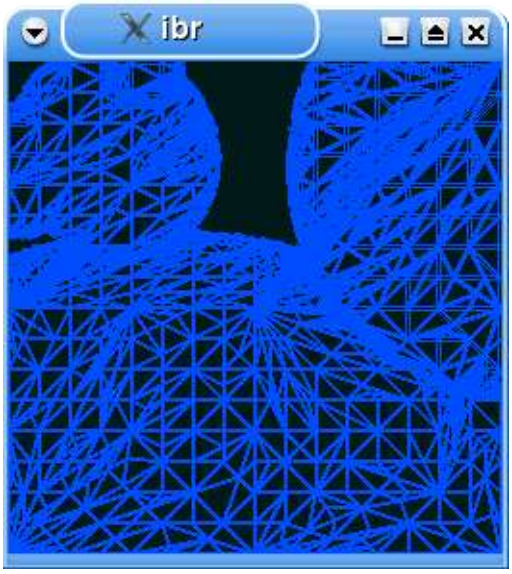

Abb. 4: Ausgehend von Cv reprojeziertes Dreiecksgitter wie es für die Bestimmung der Tiefen verwendet wird

3.1.3 Hardwarebeschleunigung

Da die Rechenleistung der CPU nicht ausreicht, um die oben aufgeführten Schritte mit akzeptabler Bildwiederholrate durchzuführen, werden die Berechnungen soweit möglich auf die GPU ausgelagert, da diese aufgrund ihrer parallelen Datenverarbeitung sowohl für die Berechnung der Tiefenwerte als auch für die Tiefenkorrektur hervorragend geeignet ist.

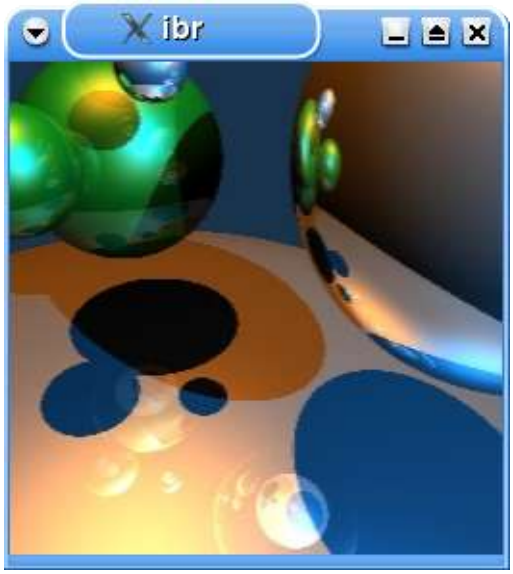

Abb. 5: Rekonstruktion einer Szene

Bei der Berechnung der Tiefe der Szene für die einzelnen Fragmente wird für alle Vertices des Triangle Strips im Vertexshader anhand der Positionen der jeweiligen Kameras sowie von C_v die korrekte Lage berechnet. Durch den Depth-Test von OpenGL wird anhand der Dreiecke für jedes Fragment der geringste Abstand zur virtuellen Kamera bestimmt und in den Depthbuffer gespeichert. Um diese Informationen den weiteren Berechnungsschritten zugänglich zu machen wird der Inhalt des Depthbuffer mittels “Copy To Texture” in eine Textur umgewandelt. Die Tiefeninformationen müssen dabei nicht in den Hauptspeicher zurückgelesen werden, so daß zeitintensive Lesevorgänge auf den Speicher der Grafikhardware vermieden werden können.

Die Tiefentextur wird dann zusammen mit den Farbtexturen, die nach Bedarf in den Speicher der Grafikhardware gelesen werden, im abschliessenden Schritt in einem Fragmentshader Programm verarbeitet, bei dem anhand der Tiefe die vier Texturkoordinaten errechnet und die ausgelesenen Farbwerte zum anzuzeigenden Wert kombiniert werden.

3.2 Image-Based Editing

3.2.1 Allgemein

Die Segmentierung stellt in vielen Bildverarbeitungsaufgaben einen wichtigen Schritt vor der analytischen Auswertungsphase dar. Ihr Ziel ist es, das Bild in eine Menge von Regionen zu zerlegen, die in Bezug auf gewisse Eigenschaften (Grauwert, Textur, Kanten, statistische Eigenschaften etc.) homogen sind. Eine universelle Strategie mit einem Algorithmus alle gesuchten Strukturen einwandfrei zu identifizieren, gibt es nicht, da die Vorgehensweisen, ihre Parameter und ihr Einsatzbereich zu unterschiedlich sind. Je unabhängiger dabei ein Algorithmus von wissensbasierter Vorinformation ist, desto breiter ist sein Einsatzbereich und somit auch sein allgemeiner Nutzen. Um Objekte, die andere Breiche des Szene zu verdecken zu entfernen braucht man die Möchlichkeit diese mittels eines Segmentierungsalgorithmus zu

markieren.
In der implementierten Variante des Image-Based-Editing ist es möglich ohne Wissen von geometrischen Formen Objekte mit dem Mause auszuwählen und gegebenenfalls auszublenden.

3.2.2 3D Segmentierungsalgorithmus

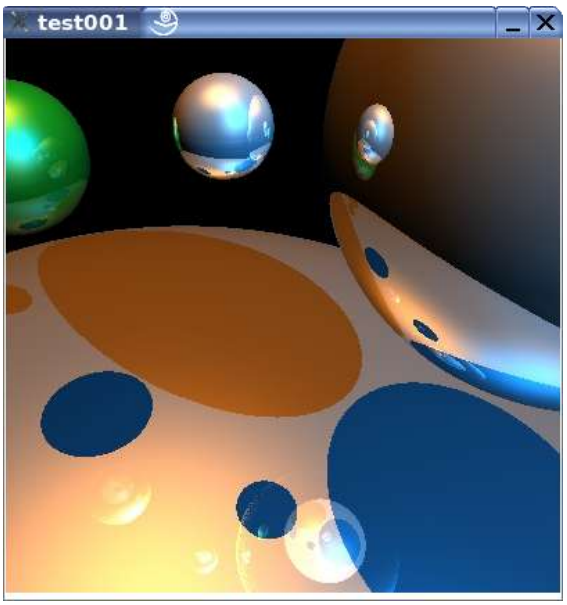

Abb. 6: Ansicht einer unmodifizierten Beispielszene

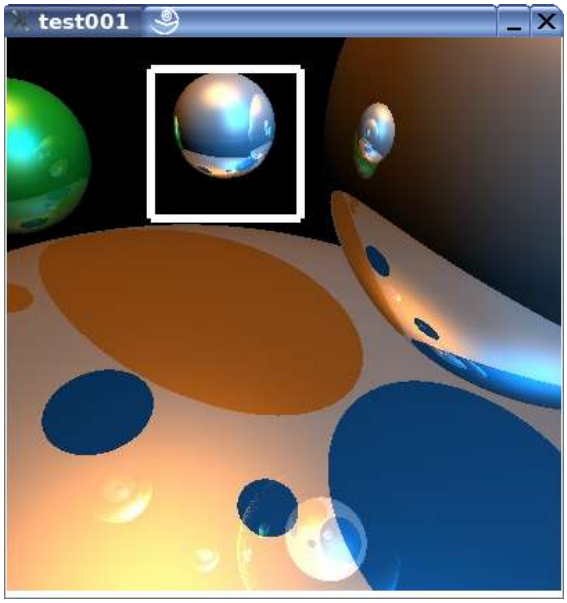

Abb. 7: Mit der Maus wird in der Szene eine Box markiert, die das zu entfernende Objekt enthält. Dabei werden die Koordinaten der Box gespeichert. Durch einen Mausklick wird dann in einem neuen Fenster eine Kopie der Szene erstellt.

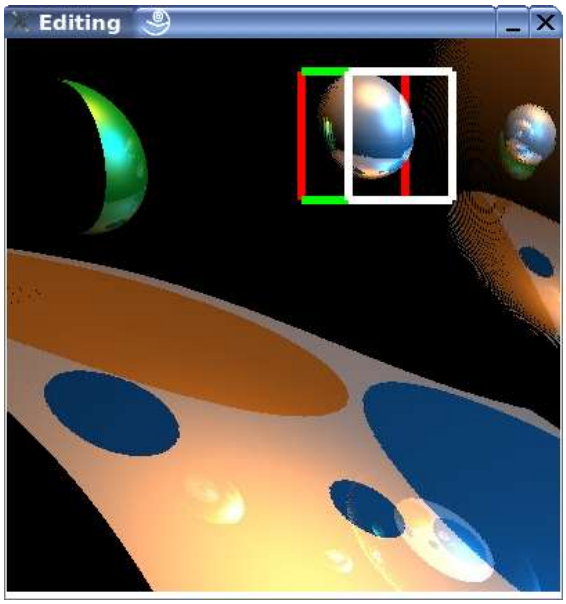

Abb. 8: Dieses Bild zeigt die Box, die den auszublendenden Bereich markiert. Das Bild wurde zur besseren Veranschaulichung rotiert.

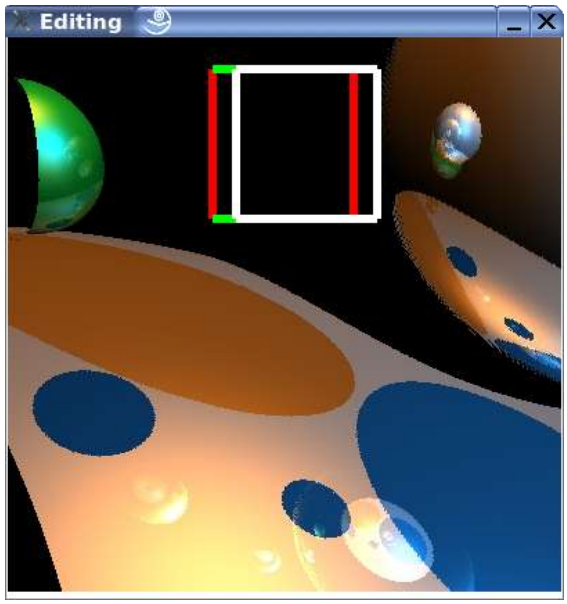

Abb. 9: Auf Wunsch kann nun der markierte Bereich ein- oder ausgeblendet werden

4. Zusammenfassung

Die Nutzung von hardwareunterstützen Grafikbibliotheken wie OpenGL oder DirectX ist für die Erzeugung neuer Ansichten eines Lumigraphs oder Light Fields hervorragend geeignet, da sich für viele Operationen Entsprechungen in den Bibliotheken (wie zum Beispiel Texture Mapping) finden lassen. Die Nutzung von aktueller Grafikhardware wie NVIDIA GForce 59x/6xxx bietet darüberhinaus aufgrund der verbesserten Programmierbarkeit mit Shaderprogrammen die Möglichkeit, die CPU durch Auslagerung vieler Berechnungen auf die GPU stark zu entlasten und durch die optimale Verteilung des Rechenaufwandes eine hohe Bildwiederholrate zu erreichen.

5. Literaturverzeichnis

[1] ADELSON, E.H., BERGEN, J. R. The plenoptic function and the elements of early vision. In Computational Models of Visual Processing, Landy and Movshon, Eds. MIT Press, Cambridge, Massachusetss, 1991, ch.1

[2] GORTLER, S.J., GRZESZCZUK, R., SZELISKI, R., COHEN, M.F. The Lumigraph. In Proc. Siggraph '96.

[3] LEVOY, M., HANRAHAN, P. Light field rendering. In Proc. Siggraph '96.

[4] ROST, R. J. OpenGL Shading Language. Addison-Wesley, 2004.

[5] VELHO, L., DEFIGUEIREDO, L.H., GOMES, J. Hierarchical Gerneralized Triangle Strips. The Visual Computer 15, 1 (1999), 21-35.

[6] ZIV, J., LEMPEL, A. A universal algorithm for sequential data compression. IEEE Transactions on Information Theory. IT-23:337-343, 1997.