

# Rendering of Spherical Light Fields

Insung Ihm, Sanghoon Park, Rae Kyoung Lee  
Department of Computer Science  
Sogang University, Korea

Präsentiert von:  
Philipp Gehr

Betreuer:  
Reto Lütolf

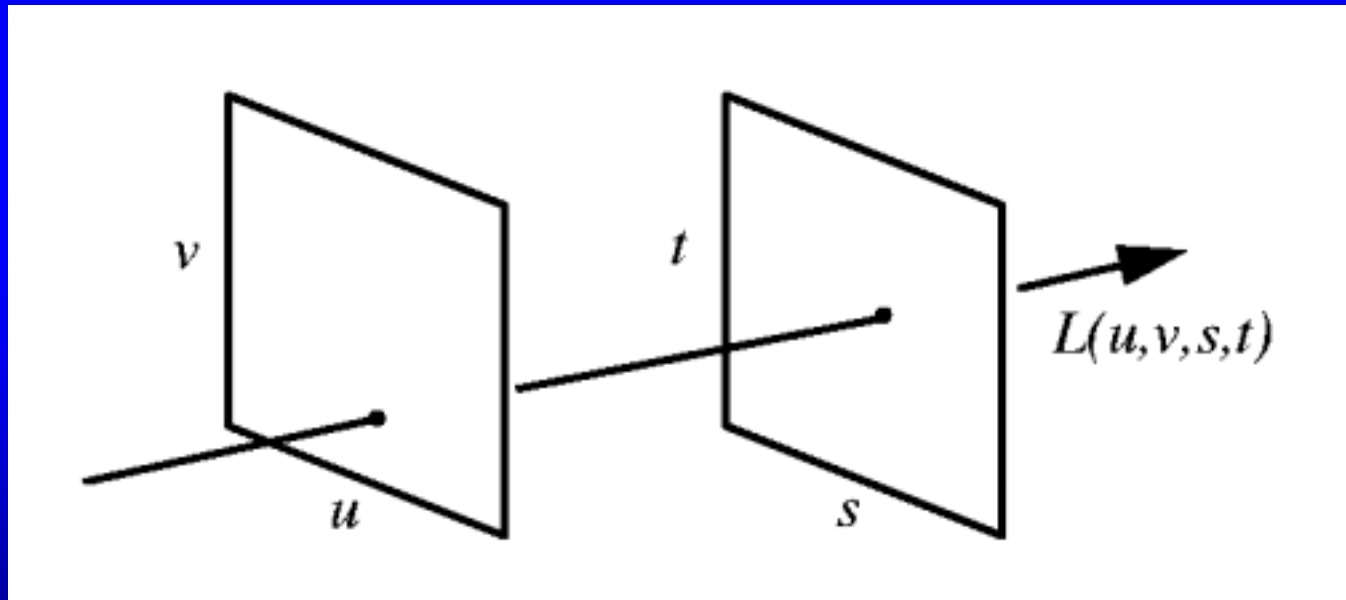
# Übersicht

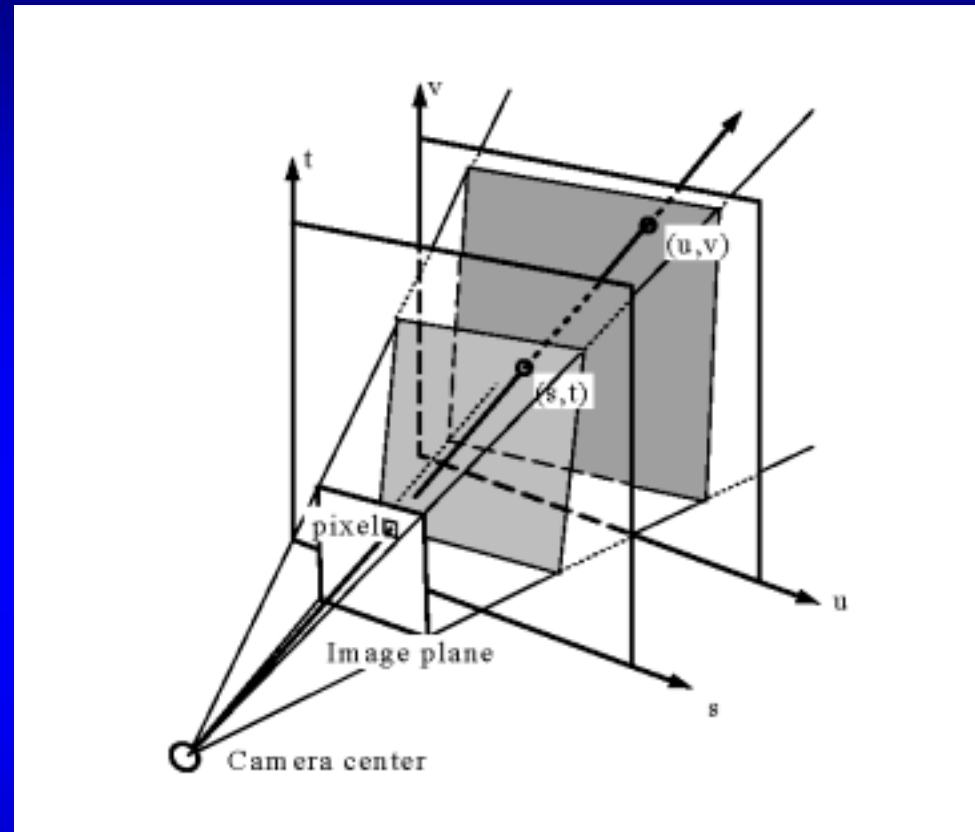
- **Einleitung und Motivation**
- **Spherical Light Field Rendering**
- **Wavelet-Kompression**
- **Experimentelle Resultate**
- **Abschliessende Bemerkungen,  
Persönliche Meinung**

# Einleitung und Motivation

- Ziel der graphischen Datenverarbeitung:  
realistische Bilder + Real-Time
- Geometry-based Rendering vs.  
Image-based Rendering
- Plenoptic Function:  $Lichtfluss = P(x, y, z, \theta, \phi)$
- Durch geeignete Parametrisierung:  
Reduktion auf 4 Dimensionen

- Light Fields:
  - Parameter  $(u,v,s,t)$
  - Light Slab





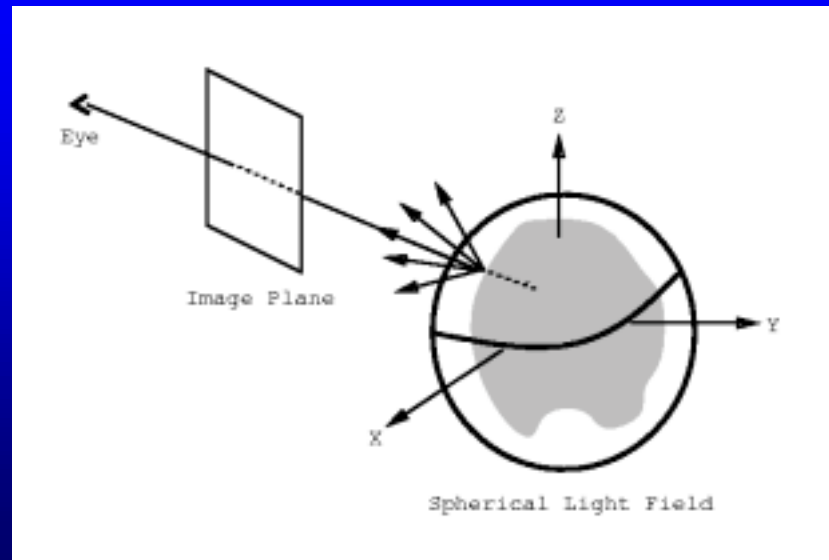
- Lumigraph

- **Spherical Light Fields: Parametrisierung mit Hilfe von sphärischen Koordinaten**
- **Image-space vs. Object-space**
- **Vorteil von Object-space Algorithmen: Einfache Einbettung in bestehende polygonale Rendering-Systeme**
  - **Hardware-Beschleunigung**
- **Hybride Rendering-Methoden**

- **Image-based Rendering Systems:**
  - Menge von Bildern
  - keine Tiefeninformationen
  - kein geometrisches Modell
  - keine Materialeigenschaften
  - keine Beleuchtungsmodelle
- **Vorteil: konstante Rendering-Kosten**
- **Nachteile:**
  - riesige Datenmengen
  - eingeschränkte Flexibilität

# Spherical Light Field Rendering

- Darstellung / Parametrisierung:
  - Punkt auf der Sphäre ( $\theta_p, \phi_p$ )
  - Lichtstrahl mit Richtung ( $\theta_d, \phi_d$ )
- Resultat:  $C = (R, G, B) = \text{Ray}(\theta_p, \phi_p, \theta_d, \phi_d)$

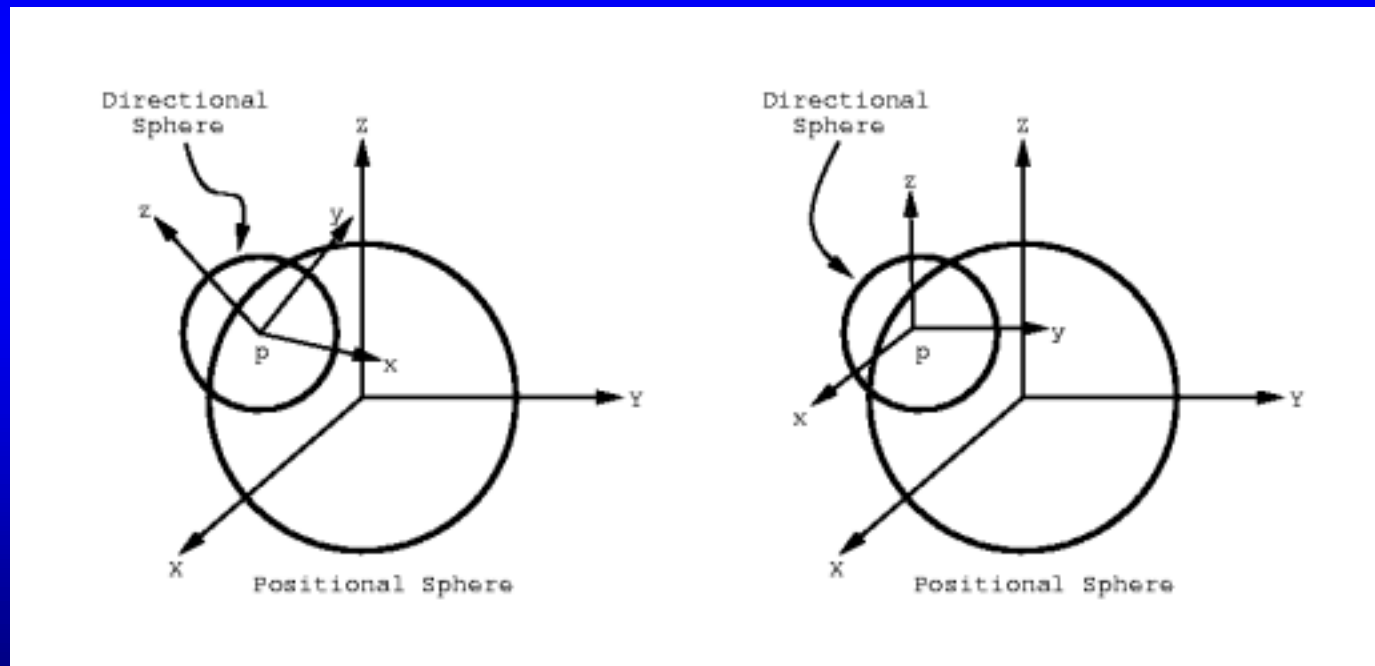




- Positions-Sphäre und Richtungs-Sphären:

Domains:  $\theta$ :  $0 \leq \theta \leq 2\pi$

$\phi$ :  $-\pi/2 \leq \phi \leq \pi/2$



- Diskretisierung:

$$\mathbf{Ray}(\theta_p, \phi_p, \theta_d, \phi_d) = \mathbf{f}_d(\theta_d, \phi_d) = (\mathbf{f}_p(\theta_p, \phi_p))(\theta_d, \phi_d)$$

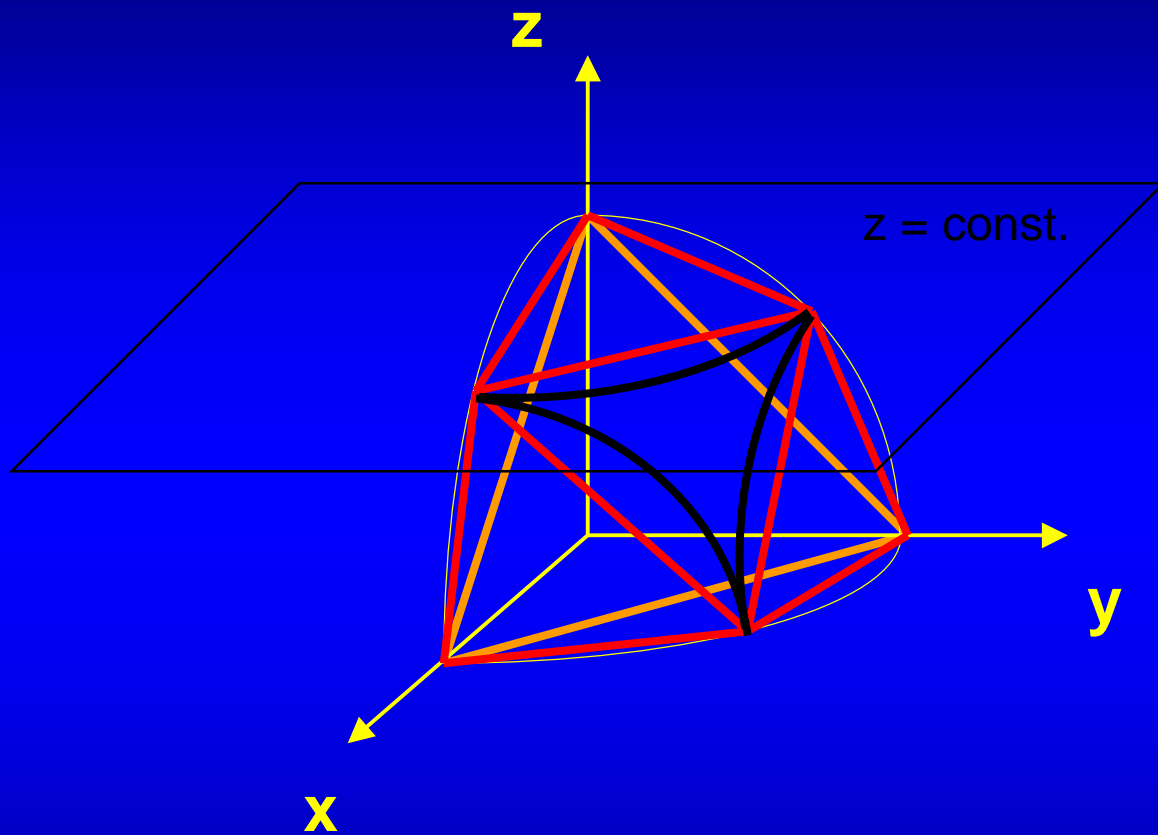
$$\mathbf{f}_p: V \rightarrow (V \rightarrow \mathbf{C})$$

$$\mathbf{f}_d: V \rightarrow \mathbf{C}$$

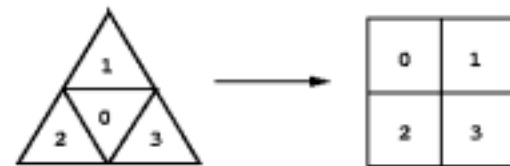
$$V = \{(\theta, \phi) \mid 0 \leq \theta \leq 2\pi, -\pi/2 \leq \phi \leq \pi/2\}$$

$$\mathbf{C} = (\mathbf{R}, \mathbf{G}, \mathbf{B})$$

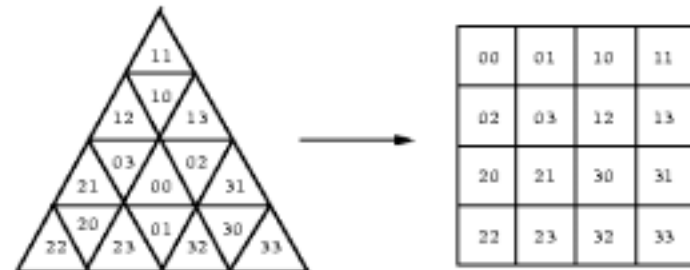
# Spherical Light Field Rendering



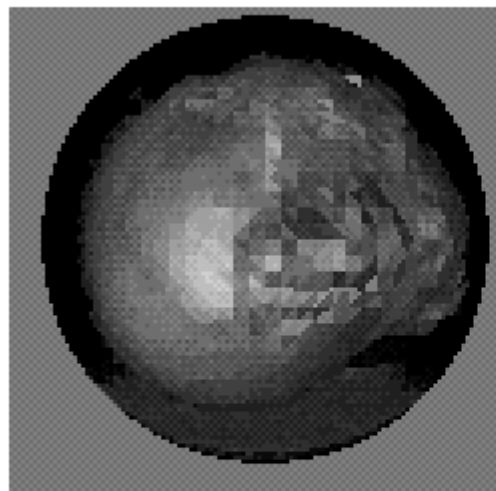
- Diskretisierung der Positions-Sphäre:  
Funktion  $f_p$  auf den Knoten des Polyeders, der die Positions-Sphäre approximiert, definiert
- Diskretisierung der Richtungs-Sphären:
  - Funktion  $f_d$  liefert Farbwerte der Dreiecke der Richtungs-Sphären
  - Daten in 2D-Array
  - Quadtree



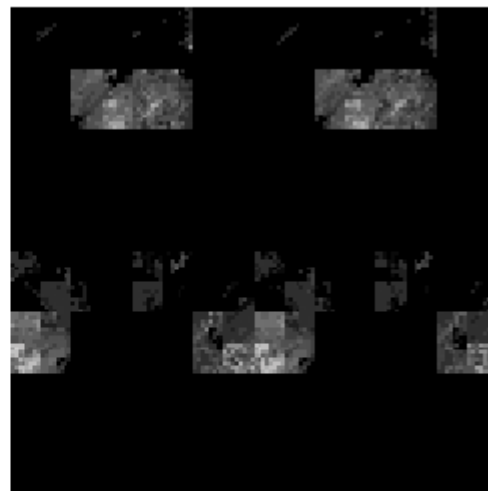
(a) Labeling order



(b) Level two subdivision



(a) A Flat-Shaded Directional Sphere



(b) Two Reordered Spheres

- **Polygonales Rendering of Spherical Light Fields:**
  - **früher: Ray Tracing**
    - **image-space Algorithmus**
  - **Ziel: object-space Algorithmus**
  - **Grund:**
    - **Integration in polygonales Rendering-System**
    - **Hardware-Beschleunigung**

- **Algorithmus:**
  - versteckte Dreiecke aussortieren
  - jedem Knoten der Positions-Sphäre wird eine Farbe zugewiesen:
    - Projektionsrichtung beim Knoten  $(\theta_p, \phi_p)$  bestimmen
    - Punkt  $(\theta_d, \phi_d)$  auf Richtungs-Sphäre
    - Funktion  $\text{Ray}(\theta_p, \phi_p, \theta_d, \phi_d)$  auswerten
    - Farbwert (R,G,B)
  - Projektion der Dreiecke auf Bildebene
  - Bilineare Interpolation

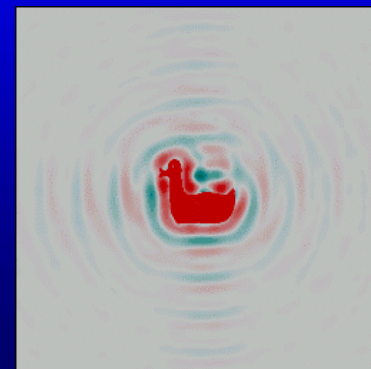
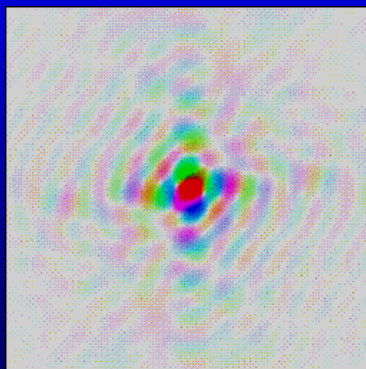
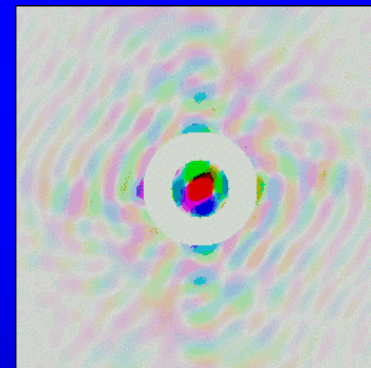
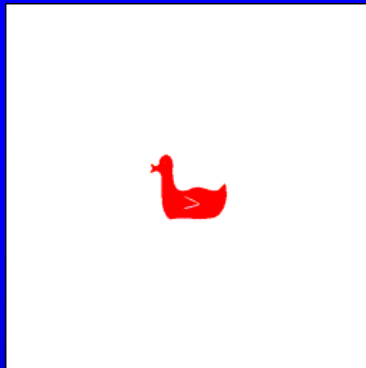
- **Beispiel für polygonales Rendering-System: OpenGL**
- **Voraussetzungen für Hardware-Beschleunigung:**
  - **Viewing**
  - **Culling**
  - **Smooth-Shading**
- **Berechnungen in Software:**
  - **Farbe eines Knotens ( $\theta_p, \phi_p$ ) auf der Positions-Sphäre**
  - **diskrete Richtungs-Sphäre → Resampling: Nearest-neighbour Resampling**



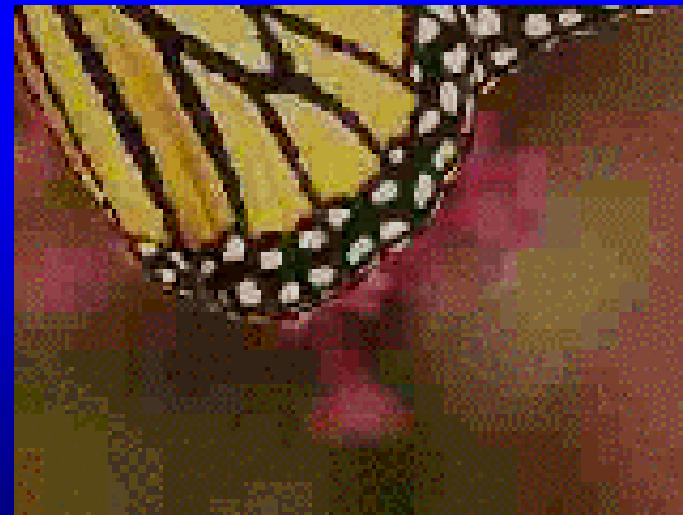
# Wavelet-Kompression

- riesige Datenmengen → Datenkompression
- Anforderungen an die Kompression:
  - Redundanz beseitigen
  - Kosten für Zugriff auf Daten minimal
- Wavelets: mathematisches Werkzeug zur hierarchischen Repräsentation von Funktionen

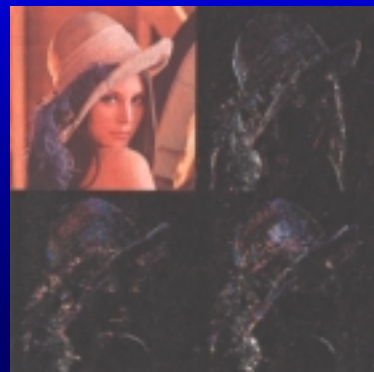
- **JPEG (Joint Photographic Experts Group):**
  - verlustbehaftete Kompression
  - Diskrete Cosinus-Transformation (DCT)
- **Signale (Bildraten) werden in Anteile unterschiedlicher Frequenzen zerlegt**



- **Quantisierung der Bildwerte**
- **Kodierung der quantisierten Koeffizienten  
+ Entropie-Kodierung**
- **Rücktransformation: Überlagerung von  
Cosinus-Schwingungen**
- **Blocking-Artefakte**

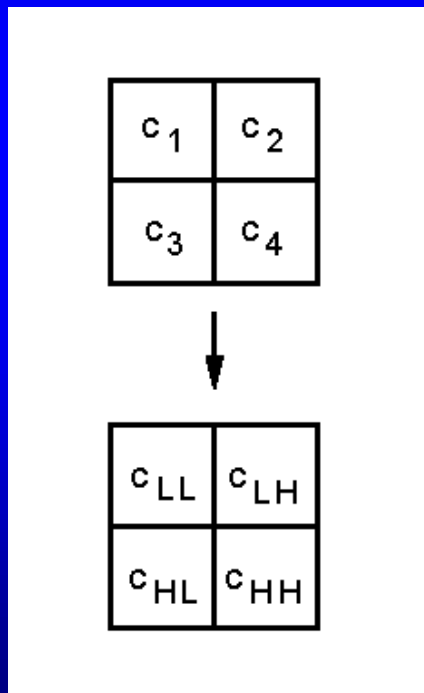


- **Wavelet-Transformation: Verwendung von kompakten Schwingung (Wavelets)**
- **Signal wird durch transformierte Versionen des Mutter-Wavelets zusammengesetzt**
- **Glättungsfilter (= Tiefpassfilter) → Mittelung**
- **Differenzenfilter (= Hochpassfilter) → Details**





- Wavelet-Kompression von Spherical Light Fields:
  - Haar-Wavelet-Zerlegung → effizient
  - konzeptioneller Quadtree

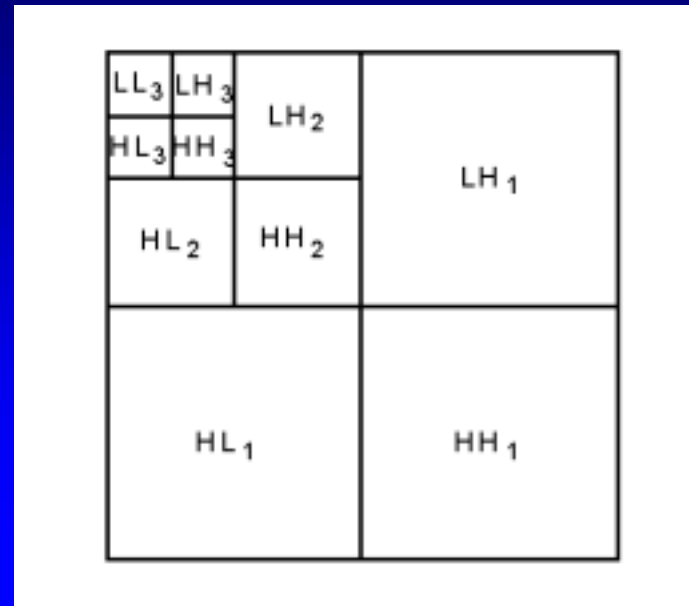
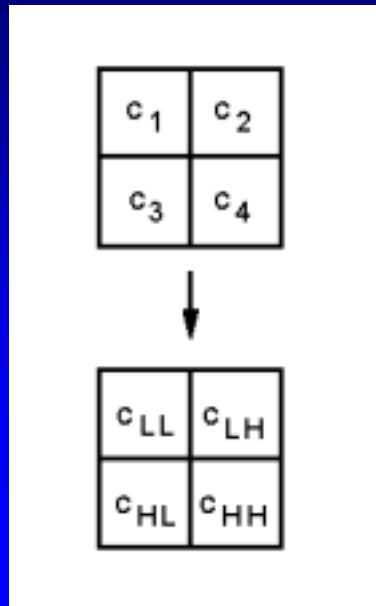


$$c_{LL} = \frac{c_1 + c_2 + c_3 + c_4}{4}$$

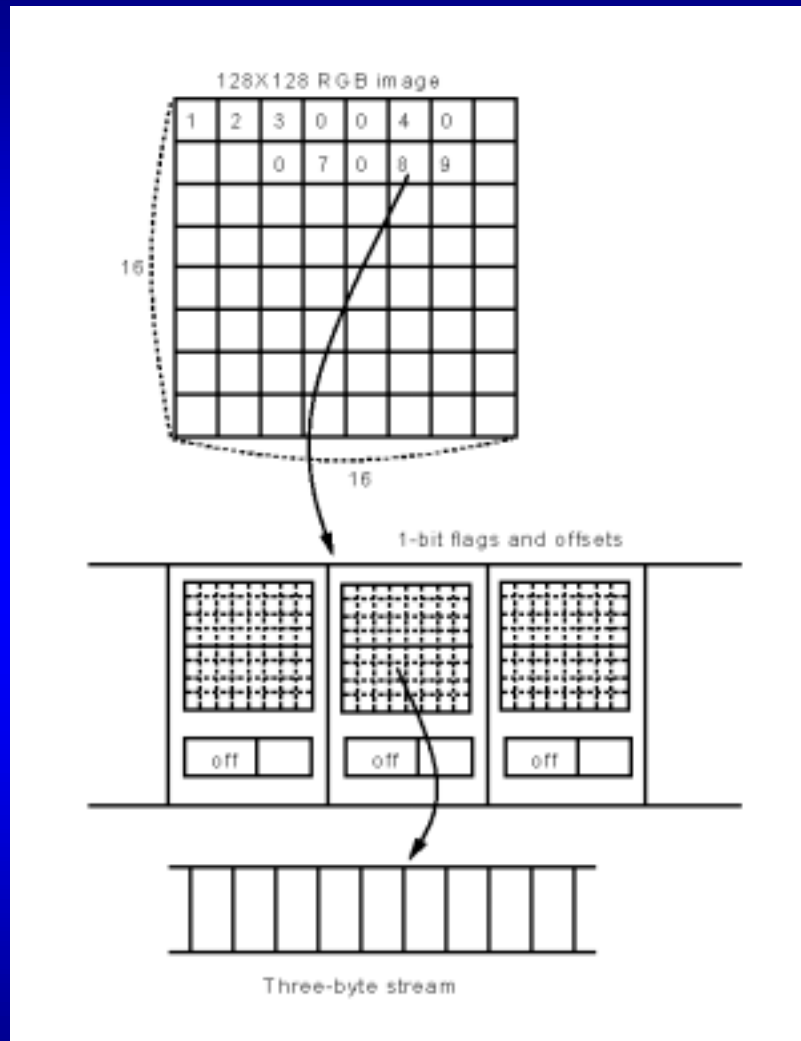
$$c_{LH} = \frac{c_1 + c_2 - c_3 - c_4}{4}$$

$$c_{HL} = \frac{c_1 - c_2 + c_3 - c_4}{4}$$

$$c_{HH} = \frac{c_1 - c_2 - c_3 + c_4}{4}$$



- **Wavelet-Zerlegung:**
  - Ausgangslage: 128x128 Bild  
3 Bytes/Pixel
  - während Zerlegung: 4 Bytes/Kanal
  - Rundung auf 1 Byte/Kanal → 3 Bytes/Pixel
  - Koeffizient  $< \tau$  → mit 0 ersetzen

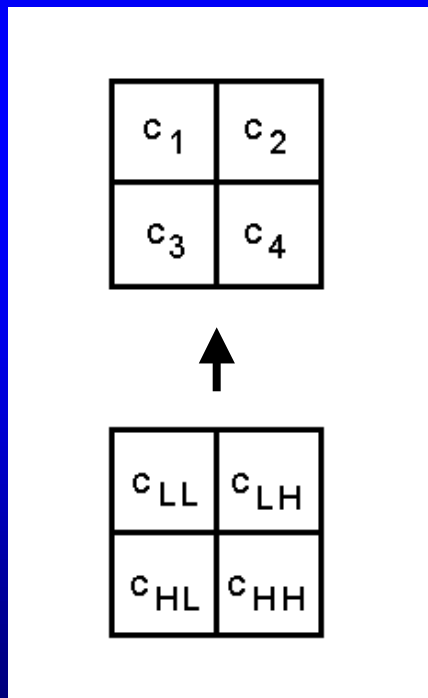


- Unterteilung in Blöcke
- Scannen und Markieren:  
1 Byte/Markierung
- Significance Map:  
8x8 1-Bit Flag Block
- Offset Information:  
2 Bytes

→ Koeffizienten-  
Quantisierung



- Farb-Extraktion aus der Richtungs-Sphäre = Farb-Rekonstruktion von wavelet-kodierten Bild = Inverse der Wavelet-Zerlegung:



$$c_1 = c_{LL} + c_{HL} + c_{LH} + c_{HH}$$

$$c_2 = c_{LL} - c_{HL} + c_{LH} - c_{HH}$$

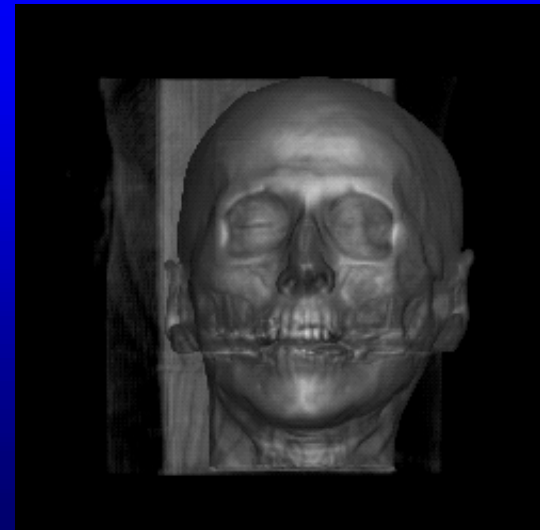
$$c_3 = c_{LL} + c_{HL} - c_{LH} - c_{HH}$$

$$c_4 = c_{LL} - c_{HL} - c_{LH} + c_{HH}$$

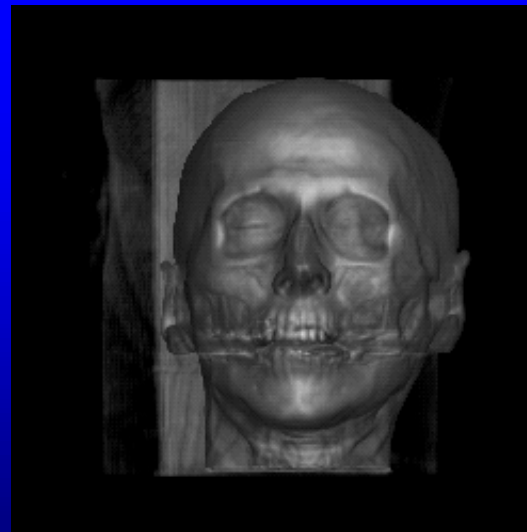
- **“gewöhnliche“ Dekodierung:**  
Quadtree traversieren → Farbe eines Pixels
- **Inkrementelle Dekodierung:**
  - **effizientere Berechnung**
  - **Pfad 2er benachbarter Rekonstruktions-Prozesse sind oft ähnlich**  
→ **Kopie des gerade berechneten Pfades**
  - **Man vergleicht nun lediglich den gespeicherten mit dem neuen Pfad und übernimmt die Farbwerte der gemeinsamen Knoten des Quadrees**

# Experimentelle Resultate

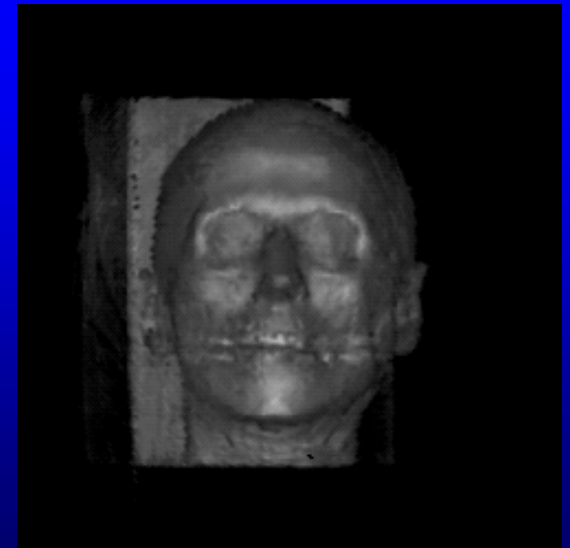
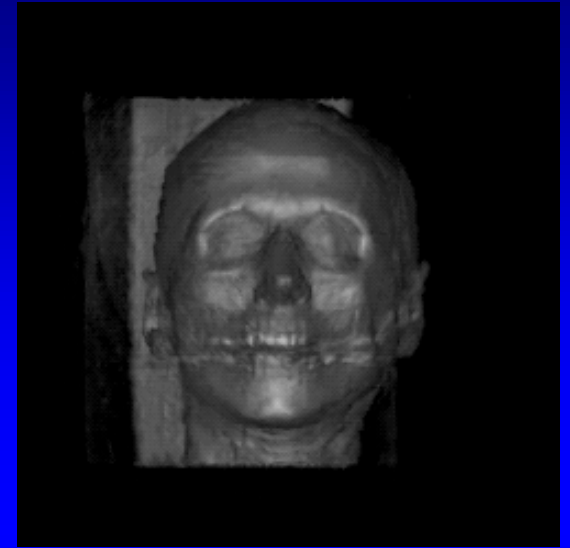
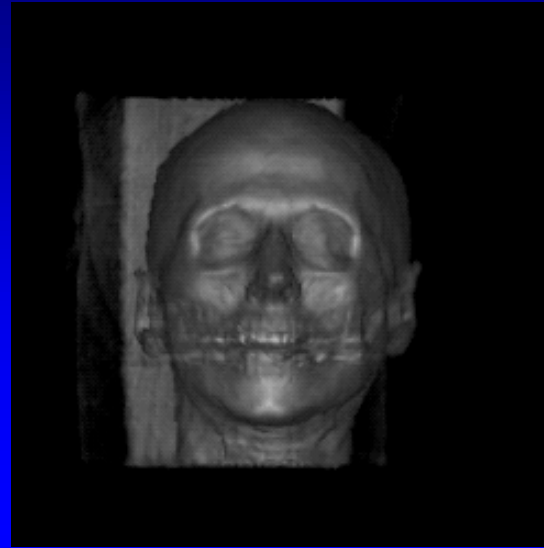
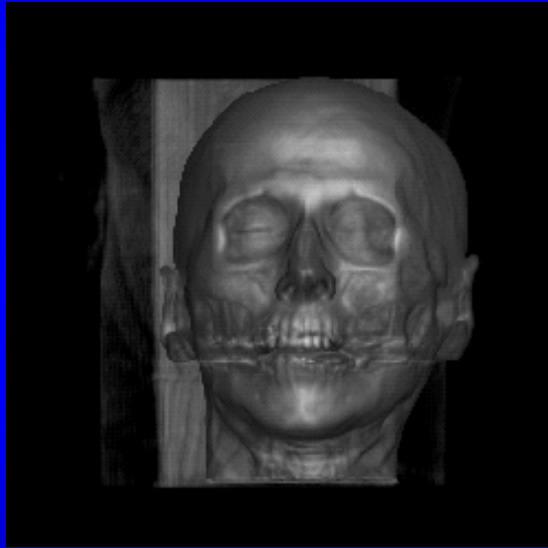
- Implementation auf einer SGI Indigo 2 (200 MHz R4400 CPU, 256 MB Main Memory, High IMPACT Graphikkarte)
- Rohdaten: CT-Scan eines menschlichen Kopfes
- Volume Ray Casting Software, um die Spherical Light Fields zu erzeugen



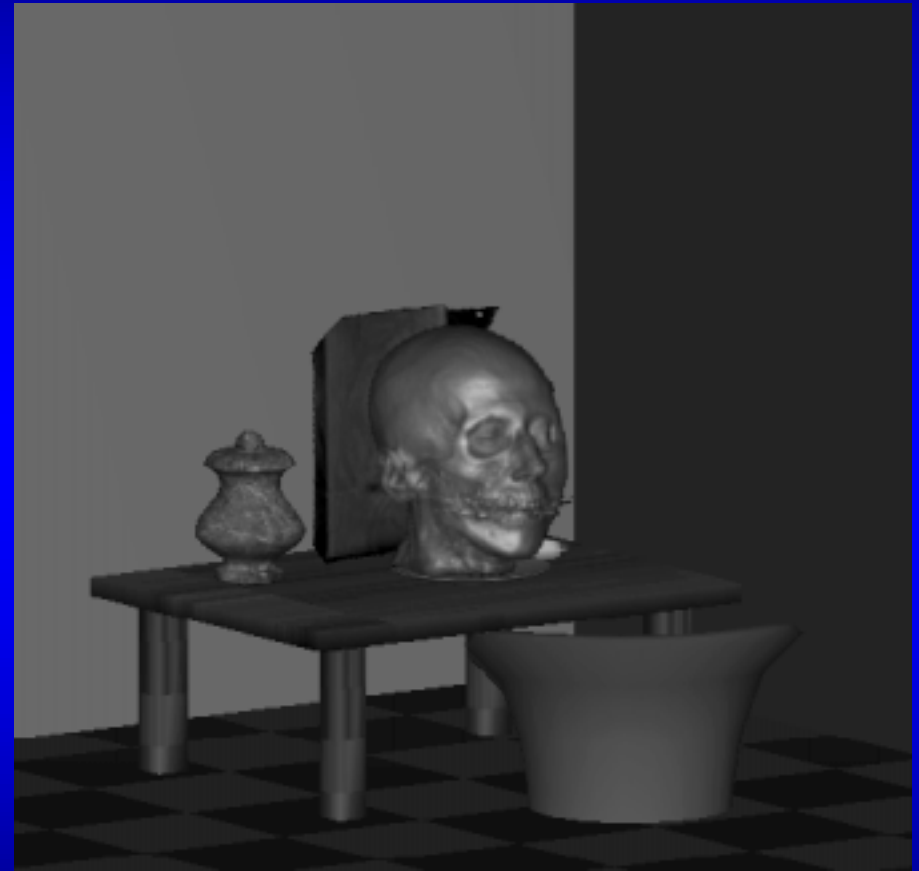
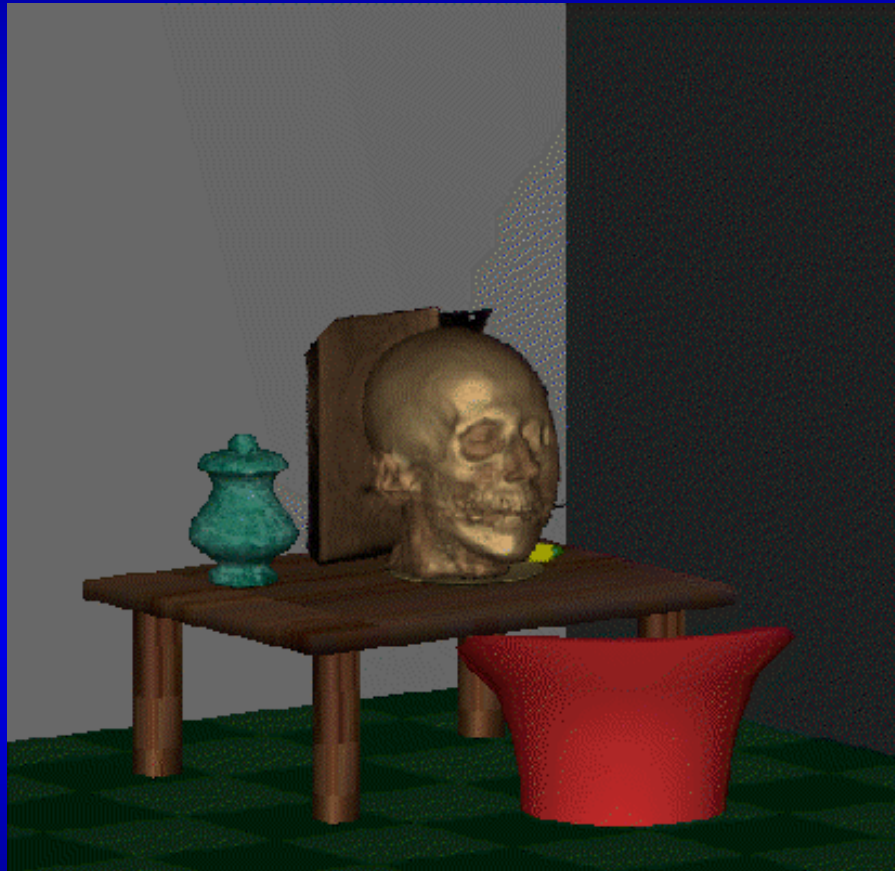
- **Spherical Light Fields: 380 MB bis 1.5 GB**
- **Speicherplatz proportional zu:**
  - **Anzahl Knoten der Positions-Sphäre**
  - **Anzahl Dreiecke der Richtungs-Sphären**



# Experimentelle Resultate



# Experimentelle Resultate



# **Abschliessende Bemerkungen, persönliche Meinung**

- **zukünftige Arbeiten:**
  - Anzahl Knoten der Positions-Sphäre reduzieren, ohne jedoch die Qualität zu senken
    - flüssigere Animationen
    - weniger Speicherplatz
- **Vorteile:**
  - realistische Darstellung in Real-time
  - Hardware-Beschleunigung
  - hybrides Rendering
- **Nachteil:**
  - Real-time wird (noch) nicht wirklich erreicht