JPEG2000

Seminar Datenkompression

Universität Bonn, WS 2005/2006 Prof. Dr. N. Blum, Matthias Kretschmer

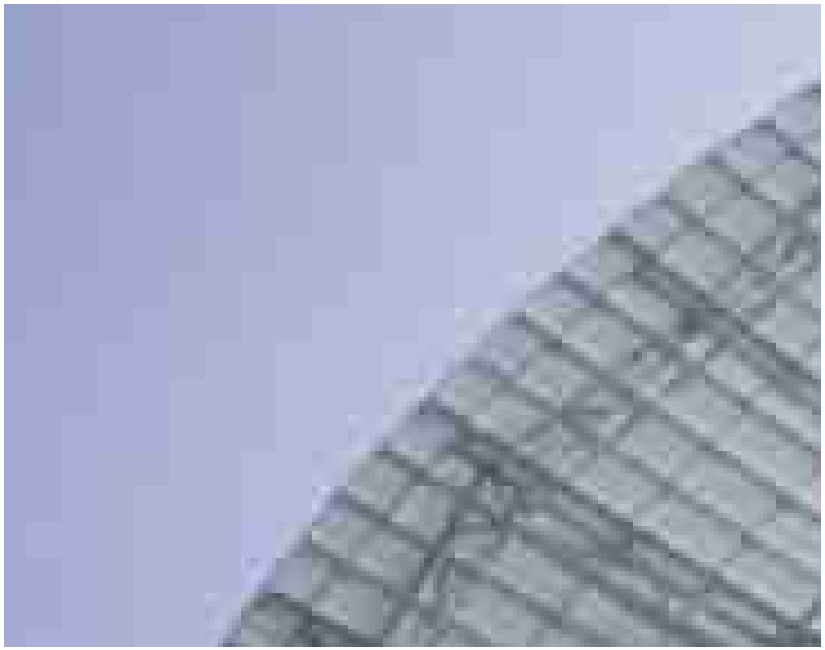
Gereon Schüller

Email:`X@Y mit X='schuelle' Y='cs.uni-bonn.de'`

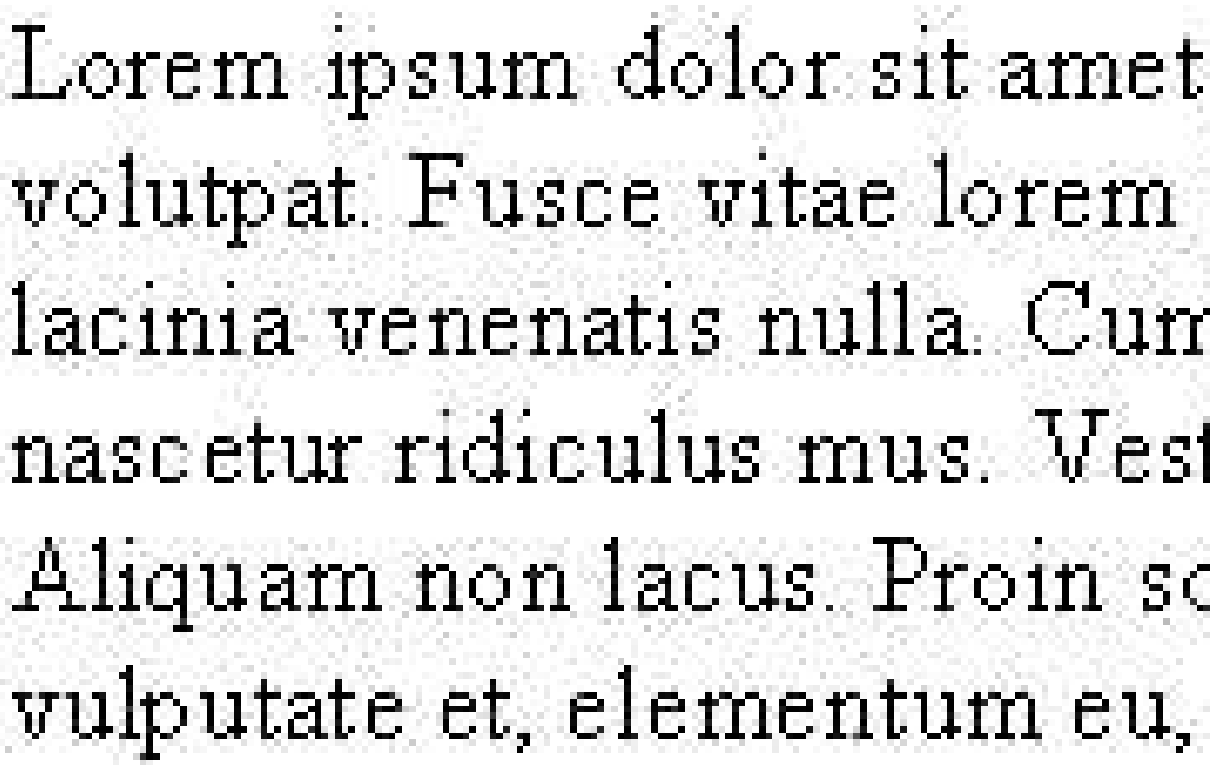
13. Dezember 2005

Probleme von JPEG

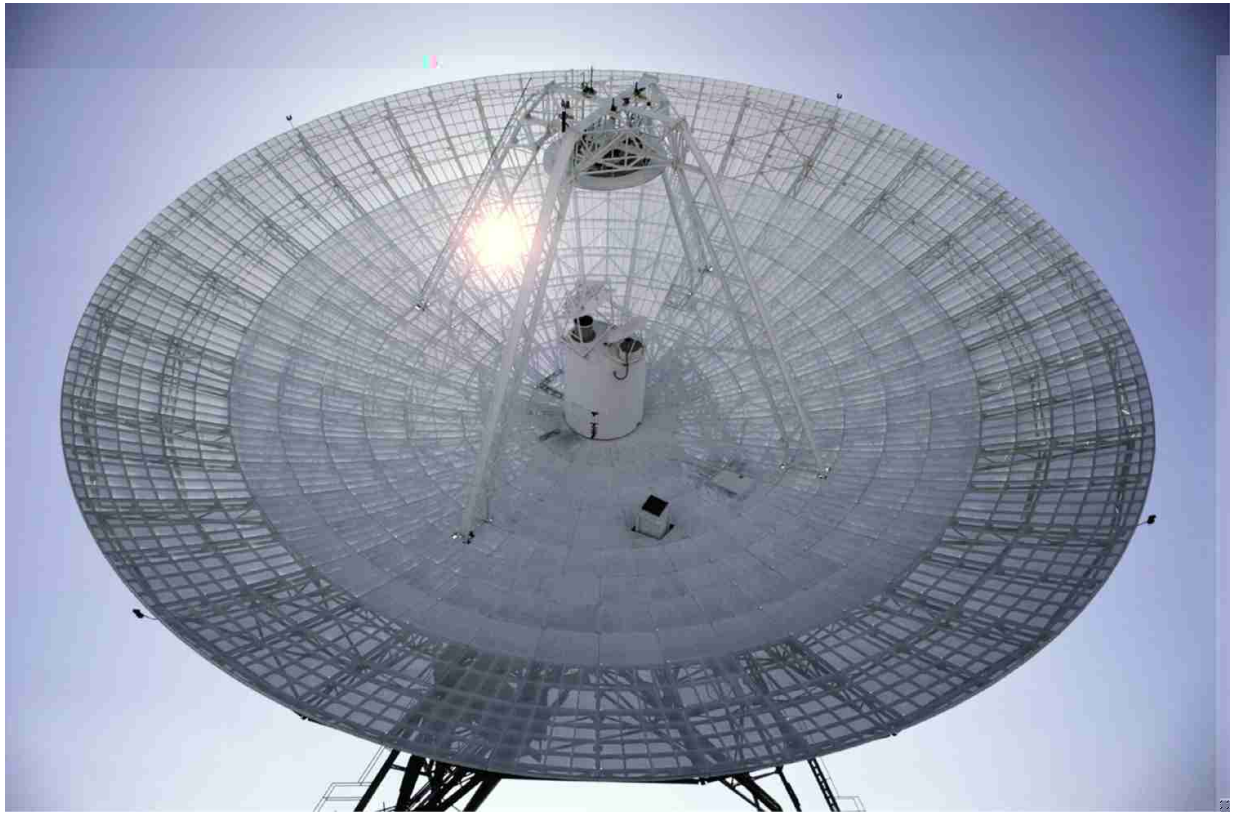
- Bildung von Block-Artefakten



- Schlechtes Ergebnis bei Texten und Computergrafiken



- Nicht robust gegen Dateifehler



(1 Byte willkürlich geändert)

- Dateigröße auf 64.000 $\times$ 64.000 Pixel beschränkt

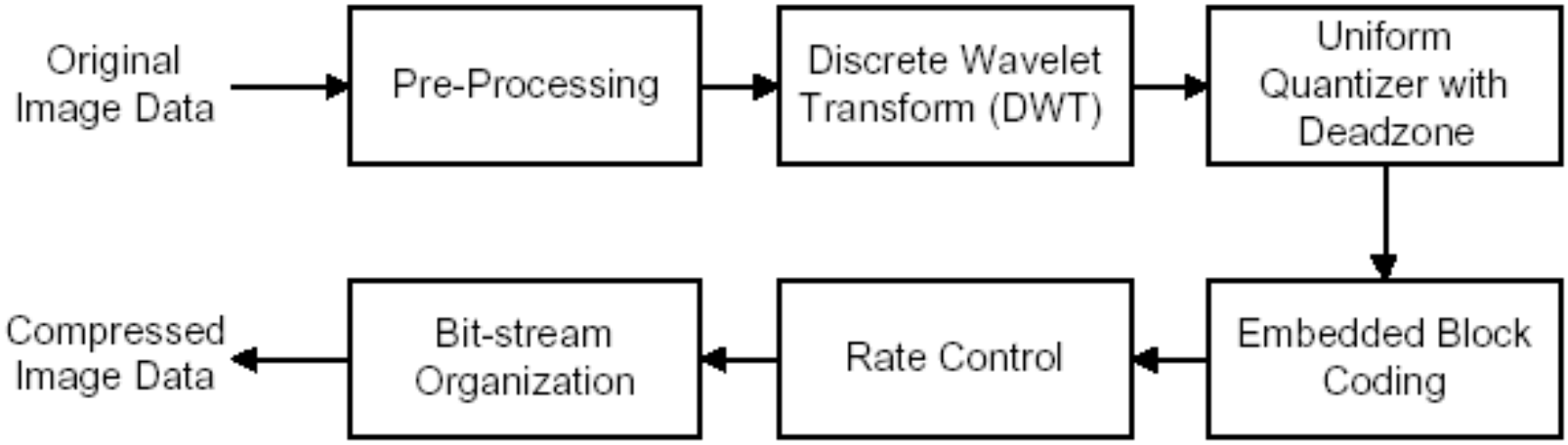
Anforderungen an JPEG2000

- Komprimierung für alle Typen von Grafiken (Fotos, computergenerierte Grafiken, Texte etc.)
- Anpassbar für viele Übertragungswege (Echtzeit, limitierte Bandbreite, Speicherung in Archiven)
- Bessere Qualität bei gleicher Dateigröße als bisherige Standards
- Schaffung von verlustlosem Kompressionsalgorithmus

Idee:

- Verwendung von diskreter Wavelet-Transformation statt diskreter Kosinus-Transformation
- Komprimierung mittels Entropieenkodierung
- Möglichkeit reversibler Komprimierung
- Speicherung in Blöcken, so dass Beschädigungen nur lokal wirken

Ablauf der Enkodierung



- Dekodierung im Wesentlichen umgekehrt

Pre-Processing

1. „Tiling“: Zerschneiden des Bildes in kleinere Stücke, falls Arbeitsspeicher nicht ausreicht
2. „Level Offset“: Addieren von -2^{B-1}, falls Bildwerte als positive Integerwerte vorliegen
3. „Irreversible Color Transform“: Umwandeln des Bildes aus dem RGB-Farbraum in den YC_bC_r-Farbraum

Farbtransformation nach YC_bC_r

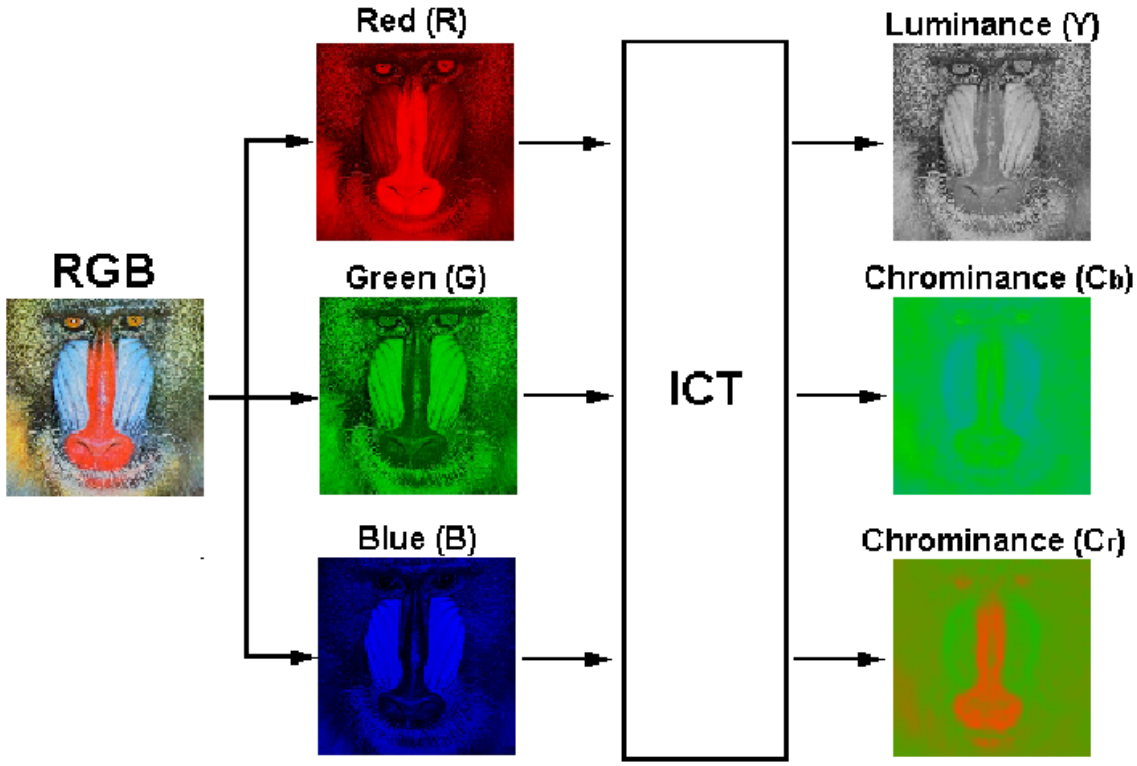
- Ursprünglich Entwicklung aus der Fernsehtechnik zur Darstellung von Farbbildern
- Bild wird in seine Luminanz, d.h. Helligkeit (=s/w-Bild) und seine Farbkomponenten (=Differenz zu blau und rot) zerlegt

Mathematisch:

$$\begin{bmatrix} Y \\ C_r \\ C_b \end{bmatrix} = \begin{bmatrix} 0.299 & 0.586 & 0.114 \\ -0.163 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

- Die Matrix ist invertierbar, aber die Transformation wegen Rundung auf Integerwerte irreversibel
- Für verlustlose Komprimierung ganzzahlige Matrix

Beispiel:



Vorteil für unsere Zwecke: Dynamischer Bereich der Farbkompenenten „flacher“, daher besser komprimierbar

Diskrete Wavelettransformation

- Herzstück von JPEG2000
- 2 Wavelets stehen zur Auswahl:
  1. Daubechies-5/3-Wavelet für verlustfreie Kompression (auch „LeGall“ genannt)
  2. Daubechies-9/7-Wavelet für verlustbehaftete Kompression

Was sind Wavelets?

- Funktionen die wie „kleine Wellen“ aussehen, daher der Name

Mathematische Anforderungen:

1. Endlicher Träger:
   $\exists x, y \in \mathbb{R} : \forall \xi < x, \eta > y : \phi(\xi) = 0 \wedge \phi(\eta) = 0$
2. $\int_{-\infty}^{\infty} |\phi(t)|^2 \, dt = 1$
3. $\int_{-\infty}^{\infty} |\phi(t)| \, dt < \infty$
4. $\int_{-\infty}^{\infty} \phi(t) dt = 0$

Wofür Wavelets

- Basisfunktion für Funktionen in stetiger „Zeit“

$\Rightarrow$ Basis heißt hier Menge linear unabh. Funktionen die alle Funktionen erzeugen können

$\Rightarrow$ Erzeugung der Basis aus den Wavelets durch Stauchung und Verschiebung

Schon lange bekannt: Das Haar-Wavelet

$$f(x) = \begin{cases} 1 & 0 \leq x \leq \frac{1}{2} \\ -1 & \frac{1}{2} \leq x \leq 1 \\ 0 & sonst \end{cases}$$

In diskreter Zeit werden Funktionswerte zu Zeilenvektoren, die Wavelets zu Matrizen.

Matrix für das Haar-Wavelet und einen Vektor $\in \mathbb{R}^2$:

$$\begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Die erste Spalte fungiert als Tiefpassfilter, die zweite als Hochpassfilter

Nachteil: Haar-Wavelet unstetig

Idee von Ingrid Daubechies:

Finde ein Wavelets mit

1. kompaktem Träger
2. stetiger Funktion
3. Orthogonalität
4. mit maximalem m, so dass $\int_{-\infty}^{\infty} t^m \phi(t) dt = 0$,
   wobei m als Ordnung des „vanishing moment“ bezeichnet wird.
   Motivation: Falls ein Wavelet eine „vanishing Moment“ der
   Ordnung m hat, so approximiert es Polynome bis zum Grad m

Dies führt zu einer rekursiven Funktion

$$\phi(x) = \sum_{k=0}^{N-1} a_k \phi(2x - k)$$

Eine Auflösung dieser Funktion zusammen mit den obigen Forderungen ergibt die Koeffizienten h_i der Wavelets. Für den Fall D_4:

$$h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}} \quad h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}} \quad h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}} \quad h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$$

$$g_0 = h_3 \quad g_1 = -h_2 \quad g_2 = h_1 \quad g_3 = -h_0$$

- Schreibe dann in Filter Matrix h_i und g_i untereinander, alle 2 Zeilen um 2 Spalten verschoben
- g als Tiefpass-Filter, h als Hochpass-Filter

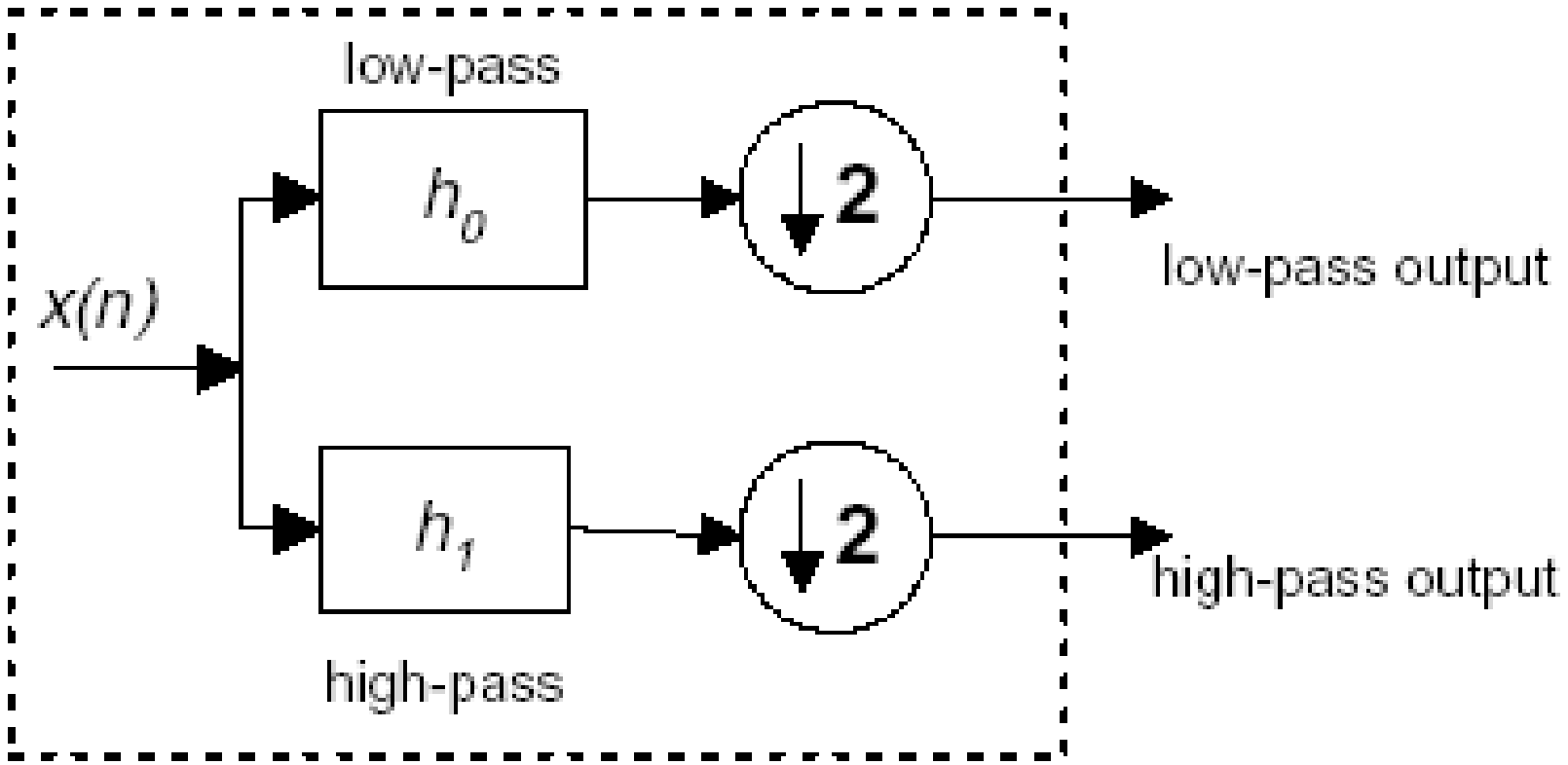
Im Folgenden werden Tiefpass- und Hochpass-Filter als getrennte Vorgänge gesehen!

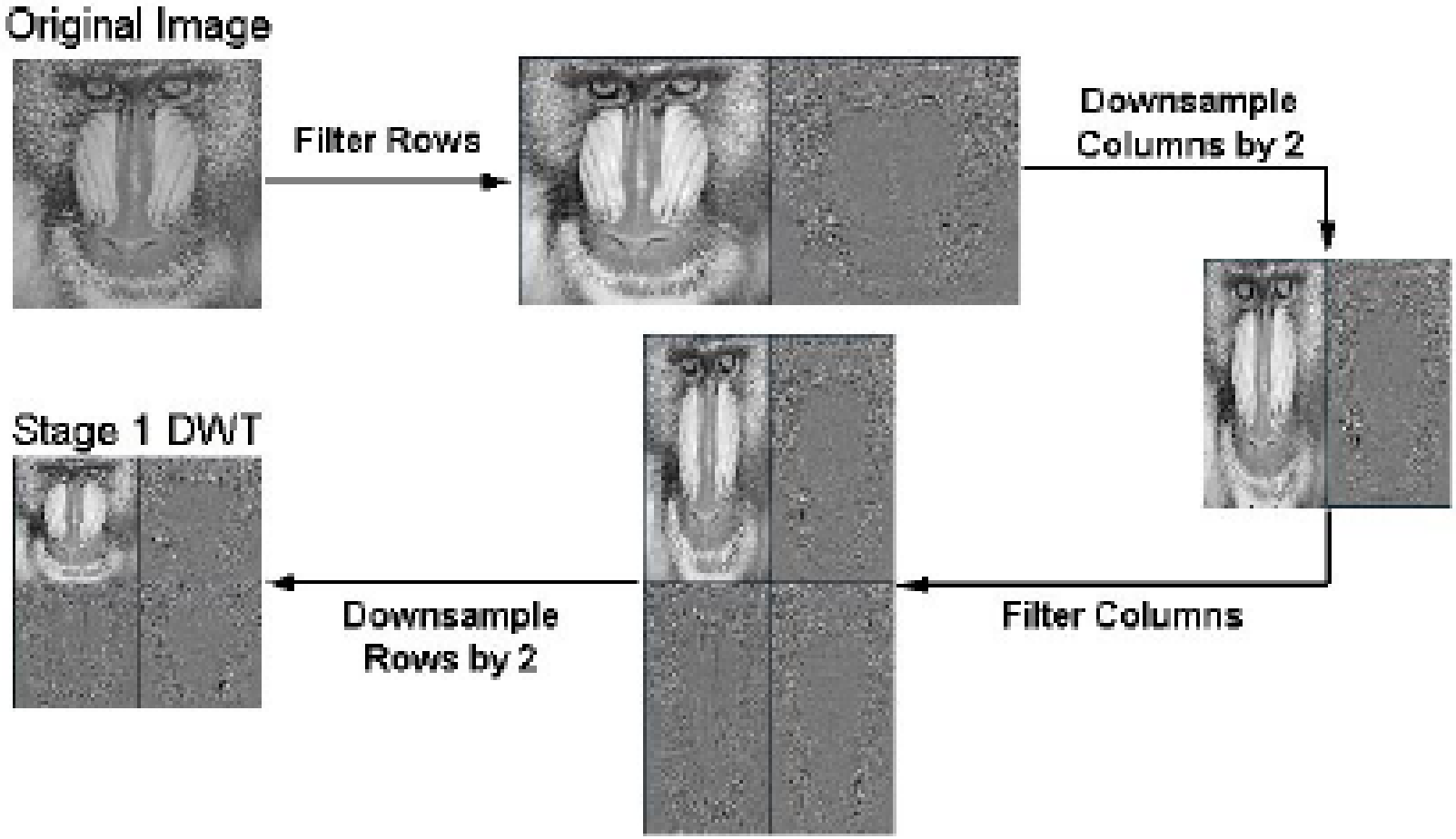
Vorgehen:

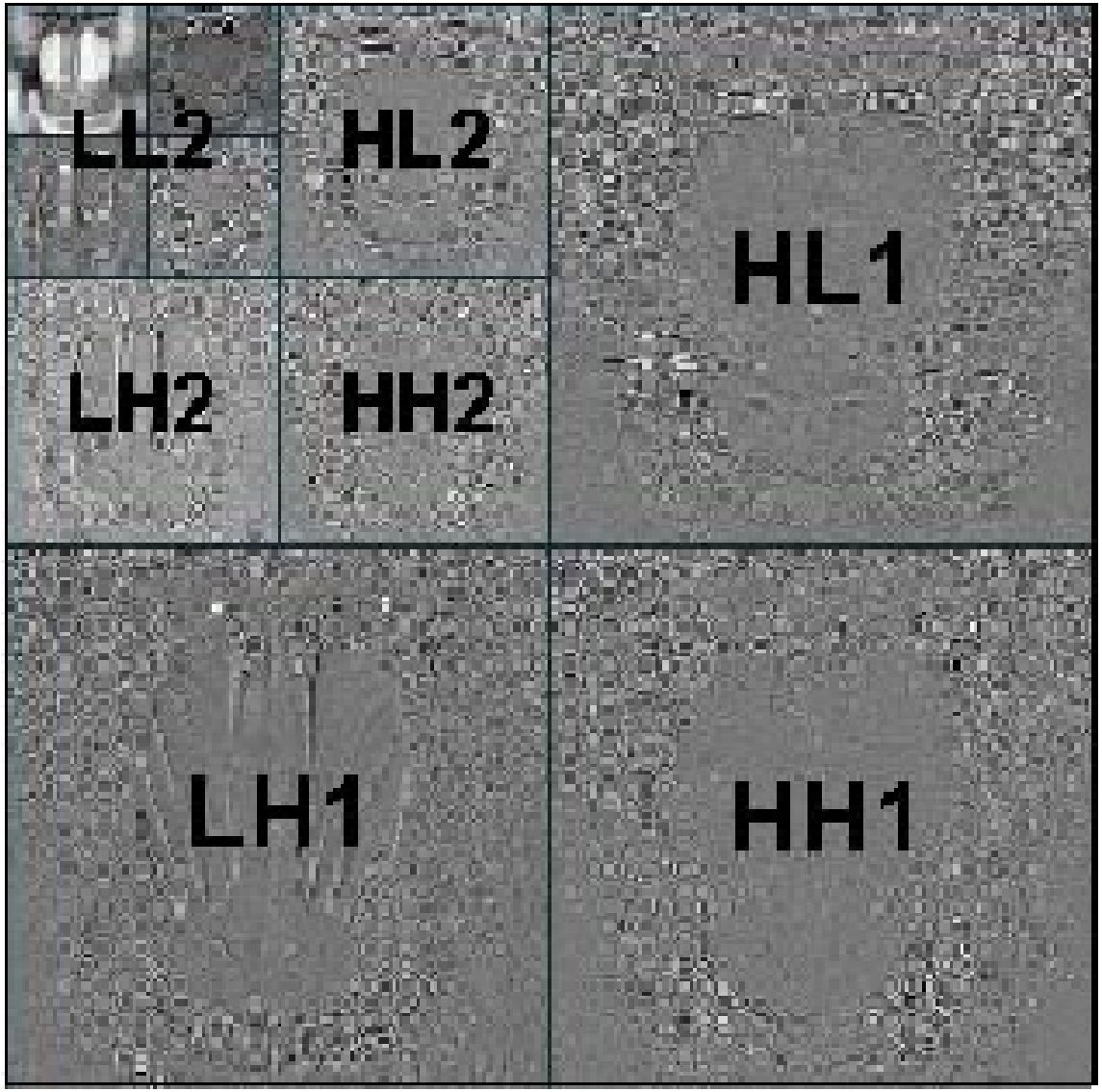
- Die Zeilen und Spalten des Bildes werden mittels eines Hoch- und eines Tiefpassfilters gefiltert
- Anschließend werden Zeilen und Spalten einem „Downsampling“ unterzogen, d.h. jeder zweite Wert wird gelöscht

$\Rightarrow$ Dabei ist das Filtern nach Zeilen und Spalten kommutativ

- Das Verfahren wird rekursiv bis zu 32 mal angewendet (für natürliche Bilder 4 bis 8 mal)


low-pass
h_0
2
low-pass output
x(n)
h_1
2
high-pass output
high-pass


Original Image
Filter Rows
Downsample
Columns by 2
Stage 1 DWT
Downsample
Rows by 2
Filter Columns


LL2
HL2
HL1
LH2
HH2
LH1
HH1

Quantisierung

Die berechneten Signal-Koeffizienten liegen nun als Dezimalzahlen vor und müssen als Integerwerte gespeichert werden

- Es wird ein sog. *Quantifizierer* eingesetzt. Dabei werden die Werte folgt gerundet:

$$q = sign(y) \left\lfloor \frac{|y|}{\Delta_b} \right\rfloor$$

$\Rightarrow$ Das Intervall $[-\Delta_b; \Delta_b]$ wird auf 0 abgebildet, daher werden besonders kleine Werte sehr effektiv komprimiert

- Dieses Intervall wird auch „Todeszone“ *(deadzone)* genannt.

Eingebettete Block-Kodierung

- Die quantisierten Koeffizenten werden nun in „Blöcke“ à 32×32 Werte eingeteilt

$\Rightarrow$ Idee: Die Blöcke werden einzeln kodiert und gespeichert, dann sind bei Beschädigungen nur kleine Bereiche betroffen

- Jeder Block wird Streifenweise nach Spalten von links nach rechts gescannt, wobei die Spalte nach 4 Werten in die nächst Spalte gewechselt wird.

- Dabei werden die einzelnen Werte nicht auf Byte- sondern auf Bitebene betrachtet. Jede Bit-Stelle wird als eigene Ebenene *(bit-plane)* betrachtet. Dabei wird zuerst die Ebene mit der größten Stelle (MSB-Plane) kodiert

Kontext-Signifikanz-Kodierung

Die *Kontext-Signifikanz* eines Bits berechnet sich aus den Signifikanzen σ seiner horizontalen, vertikalen und diagonalen Nachbarn. Dabei ist:

$$K^h[j] = \sigma[j_r, j_c - 1] + \sigma[j_r, j_c + 1] \quad (1)$$

$$K^v[j] = \sigma[j_r - 1, j_c] + \sigma[j_r + 1, j_c] \quad (2)$$

$$K^d[j] = \sum_{k_r=\pm 1} \sum_{k_c=\pm 1} \sigma[j_r + k_r, j_c + k_c] \quad (3)$$

K^{sig} errechnet sich aus K^h,K^v,K^d.

1. Durchlauf: Signifikanz-Propagation

Kodiere alle insignifikanten Bits, die kontext-signifikant sind. Dabei seien

$v[j]$:=Wert von Bit an Stelle j, $v^{(p)}$:=Byte an Stelle j ohne die letzten p Bits

$\sigma[j]$:=Signifikanz von Bit an Stelle j, inital 0

$\pi[j]$:=Markierung, dass Bit schon kodiert wurde

falls der Wert des Bits 1 ist, so kodiere sein Vorzeichen und setze die Signifikanz.

1: **for all** j **do**
2: **if** $\sigma[j] = 0 \wedge K^{sig} > 0$ **then**
3: MQ-ENCODE($v[j], K^{sig}[j]$) ▷ Kodiere Sample arithmetisch
4: **if** $v[j] = 1$ **then**
5: $\sigma[j] \leftarrow 1$ ▷ Setze Signifikanz
6: ENCODESIGN(x) ▷ Kodiere Vorzeichen
7: **end if**
8: $\pi[j] \leftarrow 1$ ▷ Markiere Bit als kodiert
9: **end if**
10: **end for**

2. Durchlauf: Magnituden-Verfeinerung

Kodiere nun alle Bits, die erst im vorherigen Signifikanz-Propagierungs-Durchlauf signifikant geworden sind, und kodiere zusätzlich den *Magnituden-Kontext*

1: **for all** j **do**
2: **if** $\sigma[j] = 1 \wedge \pi[j] = 0$ **then**
3: $K^{mag} = \ldots$
4: MQ-ENCODE($v[j], K^{mag}[j]$)
5: **end if**
6: **end for**

wobei

$$K^{mag}[j] = \begin{cases} 0 & \text{falls } v^{(p+1)}[j] = 1 \wedge K^h[j] + K^v[j] + K^d[j] = 0 \\ 1 & \text{falls } v^{(p+1)}[j] = 1 \wedge K^h[j] + K^v[j] + K^d[j] > 0 \\ 2 & \text{falls } v^{(p+1)}[j] > 1 \end{cases}$$

Phase 3: Aufräumphase

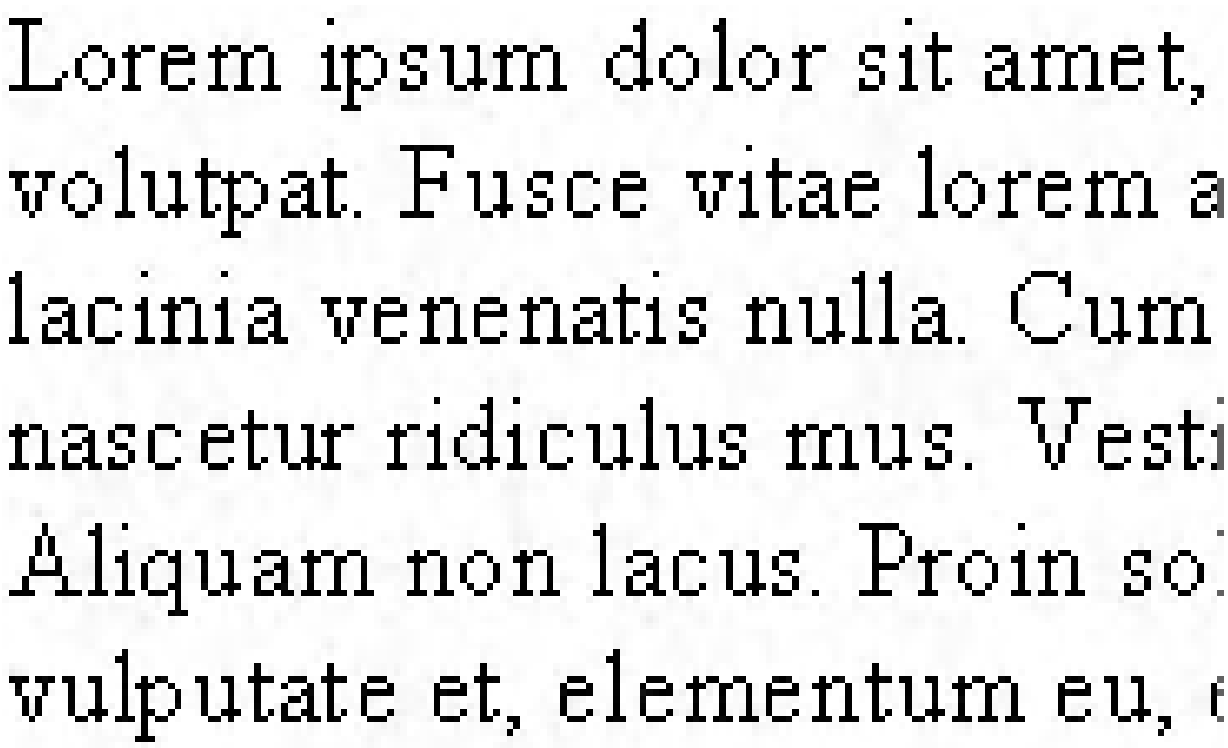
- Kodiere nun alle verbliebenen Bits (die nunmehr insignifikant sein müssen)
- Falls in einem Streifen 4 oder mehr Bits sowie ihre Nachbarn insignifikant sind, schalte in den sog. *Run-Mode* und vermerke lediglich diese Tatsache

Dekodierung

- Verläuft prinzipiell genau umgekehrt
- Neu bei JPEG2000: Einzelne Schritte können übersprungen werden, z.B.
  - Farbtransformation für s/w-Ausgabe
  - Dekompression von Layern mit niedriger Qualität, z.B. für kleine Displays
- Weiterhin können Bildbereiche weniger stark komprimiert werden, sog. *Regions Of Interest (ROI)*. Dazu wird zu den Koeffizienten ein Offset addiert
- Da jeder Block einzeln kodiert wird, gehen bei Übertragungsfehlern höchstens kleine Bildinformationen verloren

Beispiele

Text vom Anfang als JPEG2000 komprimiert



Vergleich JPEG zu JPEG2000

Quellen

1. Gray, Karen L.: „The JPEG2000 Standard“, TU München, o.J.

2. Marcellin, Michael W.: „JPEG2000“, University of Arizona, Tucson, Dezember 2002

3. Rabbani, Majid: „JPEG-2000: Background, Scope, And Technical Description“, Eastman Kodak Research Laboratories, Rochester NY, Dezember 1998

4. Taubman, D. et. al.: „Embedded Block Coding in JPEG2000“, HP Laboratories, Palo Alto, Februar 2001

Bildnachweis

Folien 7, 10, 19, 20, 21: Aus Quelle 1

Folien 2, 4, 30 (Quellbild): NASA