# Gliederung

# Gliederung

# 1. Einführung / Motivation

# Motivation zur Durchführung von Datenreduktionen

## Vergleich des Speicherplatzbedarfes für Text-, Bild-, Audio-, Video-Dateien:

- **Text:**
  Frage: Wieviel Seiten Text kann eine 3,5 Zoll-Diskette (1,4 Mbyte) speichern bei 60 Zeichen/Zeile und 30 Zeilen/Seite ?

  *Lösung:*
  1,4 Mbyte= 1,4 1024*1024 Byte= 1468006,4 / 60 / 30 = **815 Seiten**

- **Farbbild 1:**
  Wieviel Speicherplatz benötigt ein Farbbild in PAL –Norm (768x576 Pixel) mit 8 Bit / Farbe im RGBModus?

  *Lösung:*
  768 x 576 x 3 Byte (24Bit) = 1327104 Byte = 1296 kByte = **1,26 Mbyte**

- **Videosequenz 1 Sekunde (PAL):** = 1,26 Mbyte x 25 Vollbilder = **31,6 Mbyte**

- **Audio 1 Sekunde:** CD-Qualität (Abtastung 44 kHz), Stereo, 16 Bit Quantisierung
  = 44 kHz x 2 Kanäle x 2 Byte **= 176 kByte**

# Motivation zur Durchführung von Datenreduktionen / Forts.

## Vergleich des Speicherplatzbedarfes für Text-, Bild-, Audio-, Video-Dateien

**Farbbild 2:**

Ein (DIN)-A4- Farbbild in 600 ppi x 600 ppi – Auflösung im RGB-Modus (8 Bit /Farbe) liefert eine Datenmenge von: (DIN A4 (297 x 210 mm2) ; 1 inch = 2,54 cm) = 11,88 inch x 8,4 inch x 600 ppi x 600 ppi x 3 Byte
= 107.775.360 Byte = 105.249 kByte = **102,8 MByte**

**Farbbild 3:**

Geldschein 10 Euro mit 2400 ppi x 2400 ppi in RGB (8 Bit/Farbe) gescannt: = **215 Mbyte**

# Motivation zur Durchführung von Datenreduktionen / Forts.

## Beispiele DVD-Video

- Wieviel Minuten Video würde eine DVD5 (4,7 GByte, 1 Seite, 1 Schicht ) im PAL-Format (768 x 576 x 25 Bilder/s , bei 8 Bit/Farbe im RGB-Modus) unkomprimiert aufnehmen können?

  *Lösung:*

  1 s Video unkomprimiert PAL (768x576x 25 x 8Bit/Farbe) = 31,6 MByte (siehe oben)
  31.6 Mbyte/s = 31,6/1024 Gbyte/s = 0,0308 GByte/s
  Zeit [min] = 4,7 GByte / 0,0308 GByte/s = 152.6 s = **2,54 Minuten !**

- Wie hoch muß die Kompression sein, damit 4h Video in PAL auf eine DVD5 (4,7 GByte) passen?

  *Lösung:* 4h = 4*3600 s =14400s
  Kompressionsfaktor 14400 / 152,6 = **94,36**

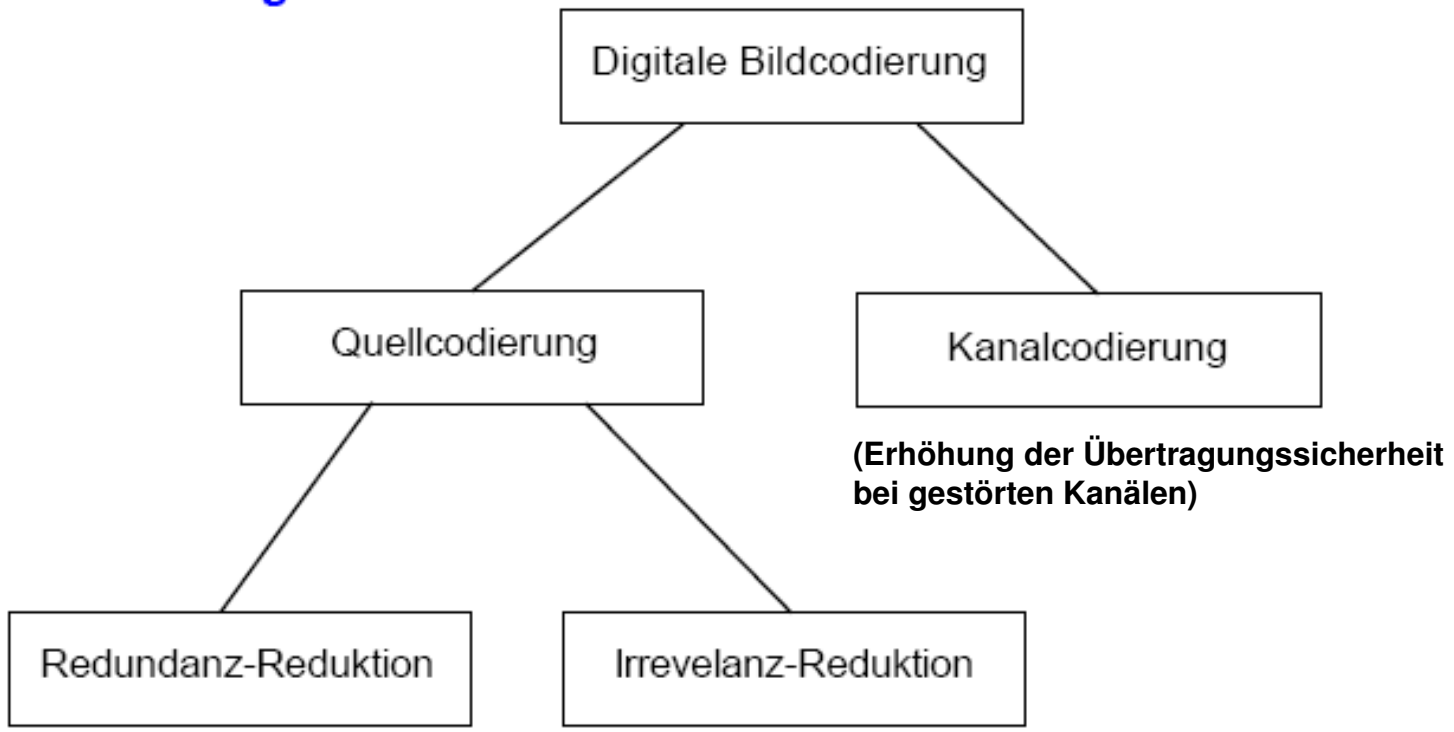
- Wieviel Minuten Video fasst eine DVD5 (4,7Gbyte) im DVD-Longplay (MPEG2, 4 MBit/s)?

  *Lösung:*
  Zeit[min] = DVD-Kap [GByte] / Datenrate [GByte/min]
  = DVD-Kap. [GByte] / (Datenrate [MBit/s]x60[Min] / 8 [MByte/s] /1024[GByte])
  = 4,7 GByte *8 * 1024/ 4 MBit/s *60 = **160 Minuten**

## Notwendigkeit der Datenreduktion mit dem Ziel

a) der platzsparende Speicherung
b) einer schmalbandige Bildübertragung
jeweils bei Erhalt des visuellen Eindrucks bzw. relevanter Informationen und/oder verlustlose Kompression bei speziellen Bildern

**Einordnung**



# Arten der Datenreduktion

## Redundanzreduktion

- RLC
- Freeman
- Quadtree
- Differenzbildcodierung (DPCM)
- Pyramiden
- Huffmann
- arithmetische Codierung
- Transformationscodierung
- JPEG
- MPEG
- Wavelet
- Fraktale

## Irrelevanzreduktion

- Kontrastempfinden
- Richtungsempfinden
- Verdeckungseffekt
- Textur-Effekt
- Frequenzgangempfindlichkeit
- Auflösungsvermögen

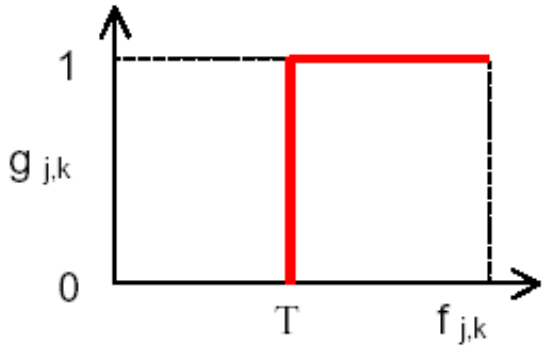
# 2. Datenreduktion durch Digitalisierung und Binarisierung

**1. Digitalisierung**

1. Datenreduktionsstufe
   Aus einer unendlichen Info-Menge
   --> durch Rasterung u. Quantisierung --> endl. Info-Menge
   --> Abtasttheorem !

**2. Binarisierung**

Aus GW-Bild wird Binärbild




$$g_{j,k} = \begin{cases} 1 & \text{für } f_{j,k} >= \text{Schwelle T} \\ 0 & \text{sonst} \end{cases}$$

z. B. GW-Bild 512 x 512 Pixel x 8 Bit =256 kByte
Binärbild =32 kByte

Verlustbehaftet – unumkehrbar !

# 3. Lauflängencodierung

Beispiel:

Die 34. Zeile eines Binärbildes trägt folgende Information:

**0 0 0 0 0 0 1 1 1 1 1 0 0 0 0 0 1 1 1 0 0 0 0 0 0 0**

bessere Darstellung: (6x0),(5x1),(5x0),(3x1),(7x0)

RLC (Variante): (6:5),(16:3) → hier: Position und Länge der „1“-Blöcke

verlustlos !

## 3. Lauflängencodierung

## Beispiele zur Lauflängencodierung von Binärbildern

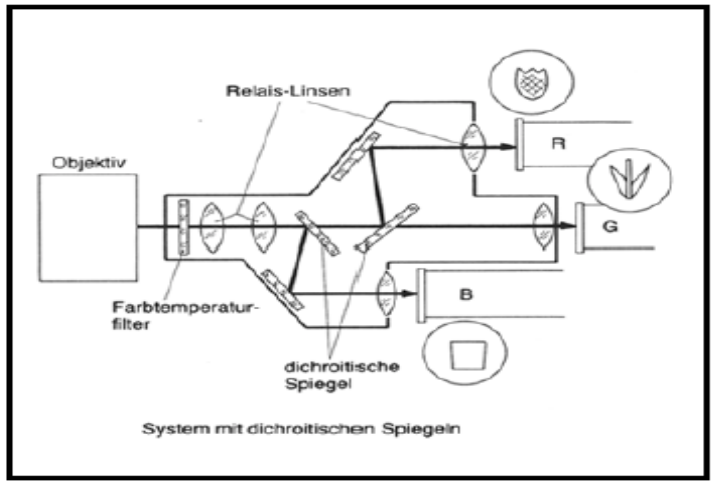
a)

Bildgröße 512 x 512 Pixel (normale Speicherung 1 Bit / Pixel --> 32 kByte)

Lauflängencodierung mit 3 Byte / Kette

a) 5,97 kByte, Kompressionsfaktor 0,19
b) 19,20 kByte, -“- 0,6
c) 34,53 kByte, -“- 1,08

b)




System mit dichroitischen Spiegeln

c)

Die Effektivität der Lauflängencodierung
Bildern, in denen sich die Bildbereiche
ergeben sich wenige Ketten pro Zeile,
Zur Kodierung von feingliedrigen Bildern
kurze Ketten notwendig, was einen ungü
5.5 zeigt drei Beispiele zur Lauflängenc
512 x 512 Bildpunkte, die bei einer nor
KByte benötigen. In der Lauflängencodi
5.97 KByte, der Ausschnitt aus einer Str
Textausschnitt in Bild 5.5-c 34,53 KByte
0,60 und 1,08. Es ist offentsichtlich, daß
Lauflängencodierung mit 3 Byte / Kette
Lauflängencodierung von Binärbildern

## Lauflängencodierung von GW-Bildern, z. B.:

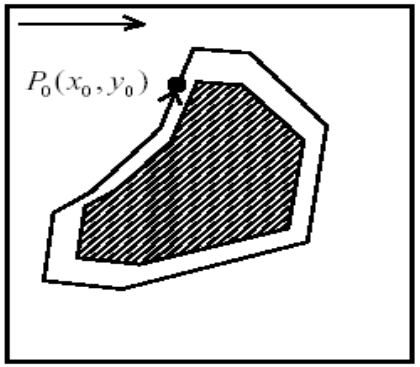
→ (vorlaufende Nullen, GW), sinnvoll bei sehr vielen Nullen im GW-Profil

Beliebige Zeile: 0 0 0 0 15 0 0 0 3 7 5 0 0 0 0 2

Code (4,15) (3,3) (0,7) (0,5) (4,2)

# 4. Konturfolge-Verfahren

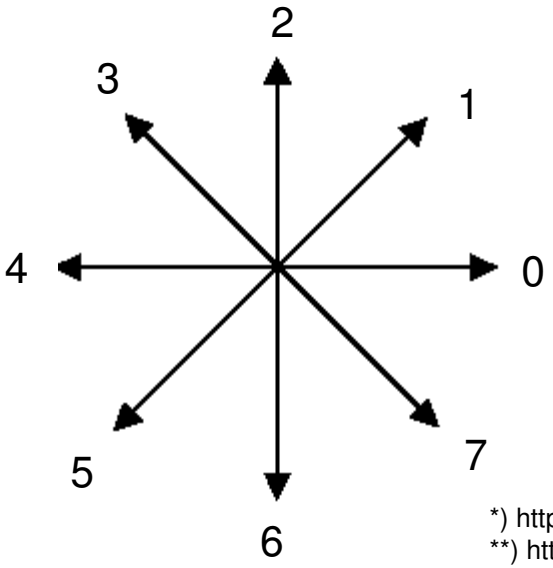
Binärbild




inhomogen, sequentiell

Vorgehensweise

1. Durchsuchen des Bildes nach Konturpunkten
   Abstand Suchzeilen =1 oder >1
2. Erster gefundener Konturpunkt
   =1 → Startpunkt
3. Konturverfolgungsalgorithmus z.B. nach Pavlidis, siehe *) und **)
4. Abspeicherung FREEMAN-Code
5. Algorithmus liefert Objektkontur, gleichzeitig können Primärmerkmale generiert werden → siehe Merkmalsextraktion




Verlustlos !

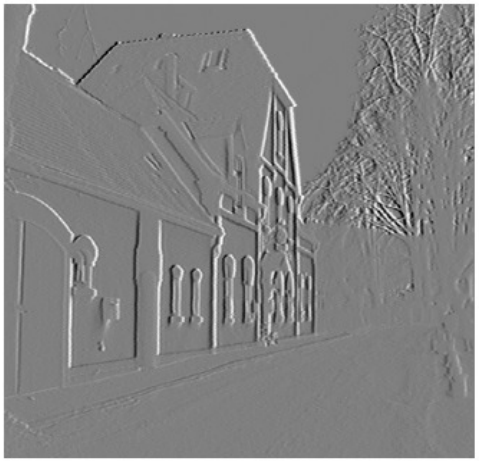
zum Beispiel:

Startpunkt P0(55,22);
7744411122223333444777.....

*) http://www.fbmn.fh-darmstadt.de/home/heckenkamp/bv1/vorlesung/konts/konturverfolgung.html
**) http://www.fbmn.fh-darmstadt.de/home/heckenkamp/bv1/vorlesung/konturcode/bv1_6.html#pcode

# 5. Differenzencodierung

Original



Differenzbild

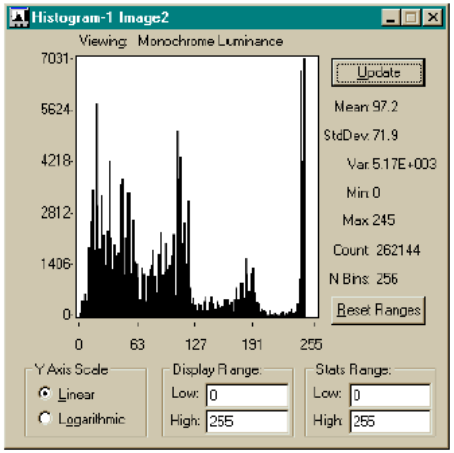
Neuer Grauwert ergibt sich z.B. jeweils
als Differenz zum linken Nachbarpixel

(reihenweise Differenzbildung)

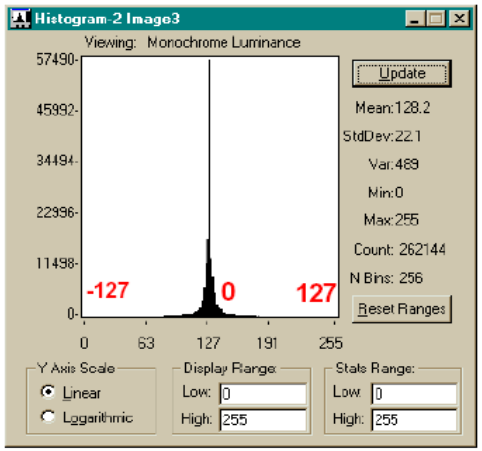
Einfache Teststruktur

links: Original

rechts: Differenzbild




Histogramm zum Originalbild




Histogramm zum Differenzbild

Entropie =7,6

Entropie = 5,8

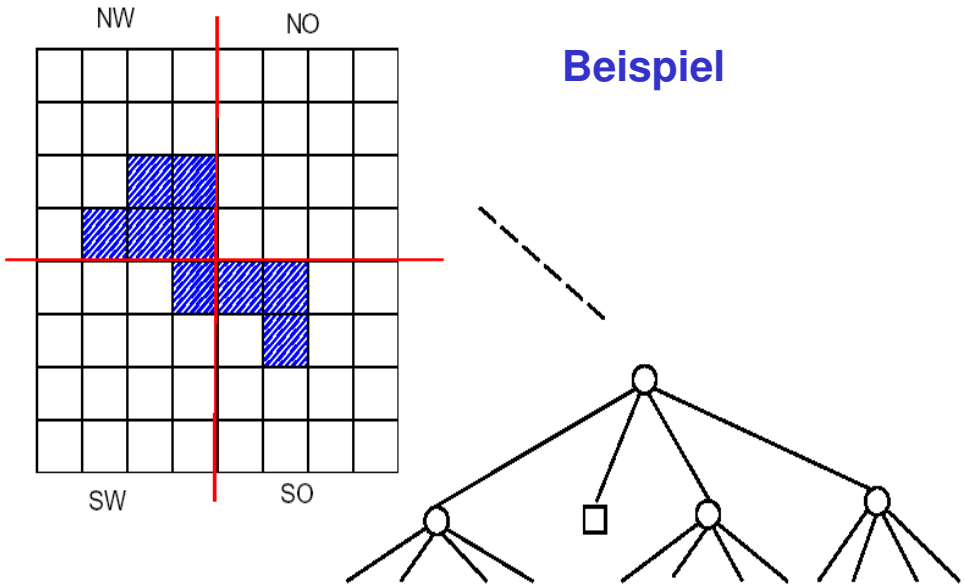
Verlustlos !

# 6. Quadtree-Codierung

Prinzip:
Untersuch. des Bildes auf homogene Gebiete → effektive Abspeicherung dieser Gebiete

**Ablauf:**

1. Gesamtbild = Wurzel
2. Einteilung des Bildes → 4 Teilbilder (NW,NO,SW,SO) → Knoten (Äste)
3. Untersuchung auf homogene Gebiete
   homogen: Abspeicherung als Blatt (Untergrund = 0, Objekt = 1)
   inhomogen: Abspeicherung als neuer Knoten, erneute Einteilung in 4 Teilbilder usw.




**Beispiel**

**Quadtree-Notierung**

1. Durchquerung des Quadtree von links → rechts
2. Abstieg → Klammer auf ‘(‘ oder auch übl.: ‘q(‘
3. Aufstieg → Klammer zu ‘)‘
4. Untergrund → 0
5. Objektpixel → 1

   Quadtree-Code:
   q(1q( 0 01q (1111)1q(0010))01q(…

Vervollständigen Sie Codebaum und Code!

# 7. Huffman-Codierung

Vergleich der Codewortlängen von ASCII-, Morse- und Huffman-Code für Buchstabenverteilungen aus englischen Texten

| Zeichen | Wahrsch. | ASCII-Code | Morse-Code | Huffman-Code |
|---|---|---|---|---|
| Leerz. | 0.1859 | 00100000 | space | 000 |
| A | 0.0642 | 01000001 | 01 | 0100 |
| B | 0.0127 | 01000010 | 1000 | 0111111 |
| C | 0.0218 | 01000011 | 1010 | 11111 |
| D | 0.0317 | 01000100 | 100 | 01011 |
| E | 0.1031 | 01000101 | 0 | 101 |
| F | 0.0208 | 01000110 | 0010 | 001100 |
| G | 0.0152 | 01000111 | 110 | 011101 |
| H | 0.0467 | 01001000 | 0000 | 1110 |
| I | 0.0575 | 01001001 | 00 | 1000 |
| J | 0.0008 | 01001010 | 0111 | 0111001110 |
| K | 0.0049 | 01001011 | 101 | 01110010 |
| L | 0.0321 | 01001100 | 0100 | 01010 |
| M | 0.0198 | 01001101 | 11 | 001101 |
| N | 0.0574 | 01001110 | 10 | 1001 |
| O | 0.0632 | 01001111 | 111 | 0110 |
| P | 0.0152 | 01010000 | 0110 | 011110 |
| Q | 0.0008 | 01010001 | 1101 | 0111001101 |
| R | 0.0484 | 01010010 | 010 | 1101 |
| S | 0.0514 | 01010011 | 000 | 1100 |
| T | 0.0796 | 01010100 | 1 | 0010 |
| U | 0.0228 | 01010101 | 001 | 11110 |
| V | 0.0083 | 01010110 | 0001 | 0111000 |
| W | 0.0175 | 01010111 | 011 | 001110 |
| X | 0.0013 | 01011000 | 1001 | 01110011000 |
| Y | 0.0164 | 01011001 | 1011 | 001111 |
| Z | 0.0005 | 01011010 | 1100 | 0111001111 |

## Verfahren

Codelänge eines Zeichens (eines Grauwertes) ist umgekehrt proportional zur Wahrscheinlichkeit seines Auftretens

***Beispiel: Für ein 2 Bit -Bild (GW: 0, 1, 2, 3) gelten folgende statistischen Eigenschaften:***

| GW | p(g) | Binärcode | Huffmancode | p(g) x Huffmanlänge |
|---|---|---|---|---|
| 0 | 0,87 | 00 | 1 | 0,87 |
| 1 | 0,06 | 01 | 00 | 0,12 |
| 2 | 0,03 | 10 | 010 | 0,09 |
| 3 | 0,04 | 11 | 011 | 0,12 |
| | | durchschn. 2Bit | | durchschn. **1,2Bit** |

Kompression = Original/Huffman = 2Bit/ 1,2 Bit =**1,67 !**
Verlustlos !

# 8. Arithmetische Codierung

# Arithmetische Codierung

1. Völlig andere Codierungsmethode

2. Effektivste Codierung, aber höchste Komplexität

3. Nachteil von Huffman: Bitlängen entsprechen nicht exakter Wahrscheinlichkeit.
(da keine Bruchteile von Bits „investiert“ werden)

4. Die Arithmetische Codierung löst Problem, indem Zeichensequenz im Intervall
[ ] 1 , 0 dargestellt wird. z.B.: 0,75 --> Null weglassen --> 75

5. Codierung erfolgt in Abhängigkeit von vergangenen Zeichen und Wahrschein-
lichkeit p(a).

6. Optimale (nahezu) Codierung

7. Nachteil: --> kein wahlfreier Zugriff

## Idee:

1. Codiere durch ein Intervall [l,r ] *r l*, mit 0 < = λ < = λ
2. Zerlege Intervall [0,1] in paarweise disjunkte Teilintervalle [la,ra ] für jedes Zeichen a.

ra - la = p(a)

### Codierungs-Algorithmus:

```
l:=0.0; r:=1.0;
        repeat                                  p: Intervall
                read (a);
                p:=r-l;
                l:=l+p*la;
                r:=l+p*ra;
until a==’EOF’;
Ergebnis: l;
```

# Beispiel 1:

Das Alphabet besteht aus den Zeichen A, B, C mit den Wahrscheinlichkeiten p(A)=0.5; p(B)=0.3; p(C)=0.2.

Eine Sequenz ACB wird dann auf ein Teilintervall in p(A) darin in p(C) und darin in p(B) abgebildet.

| Zeichen | p(a) | Intervall p |
|---|---|---|
| C | 0.2 | [0, 0.2] |
| B | 0.3 | [0.2, 0.5] |
| A | 0.5 | [0.5, 1.0] |

***Codierungs-Algorithmus:***

repeat
read (a);
p:=r-l;
l:=l+p*la;
r:=l+p*ra;
until a==’EOF’;
Ergebnis: l;

# Arithmetische Codierung – Schematischer Ablauf, dargestellt am Beispiel 1

## zu codierende Zeichenfolge: ‘ACB‘

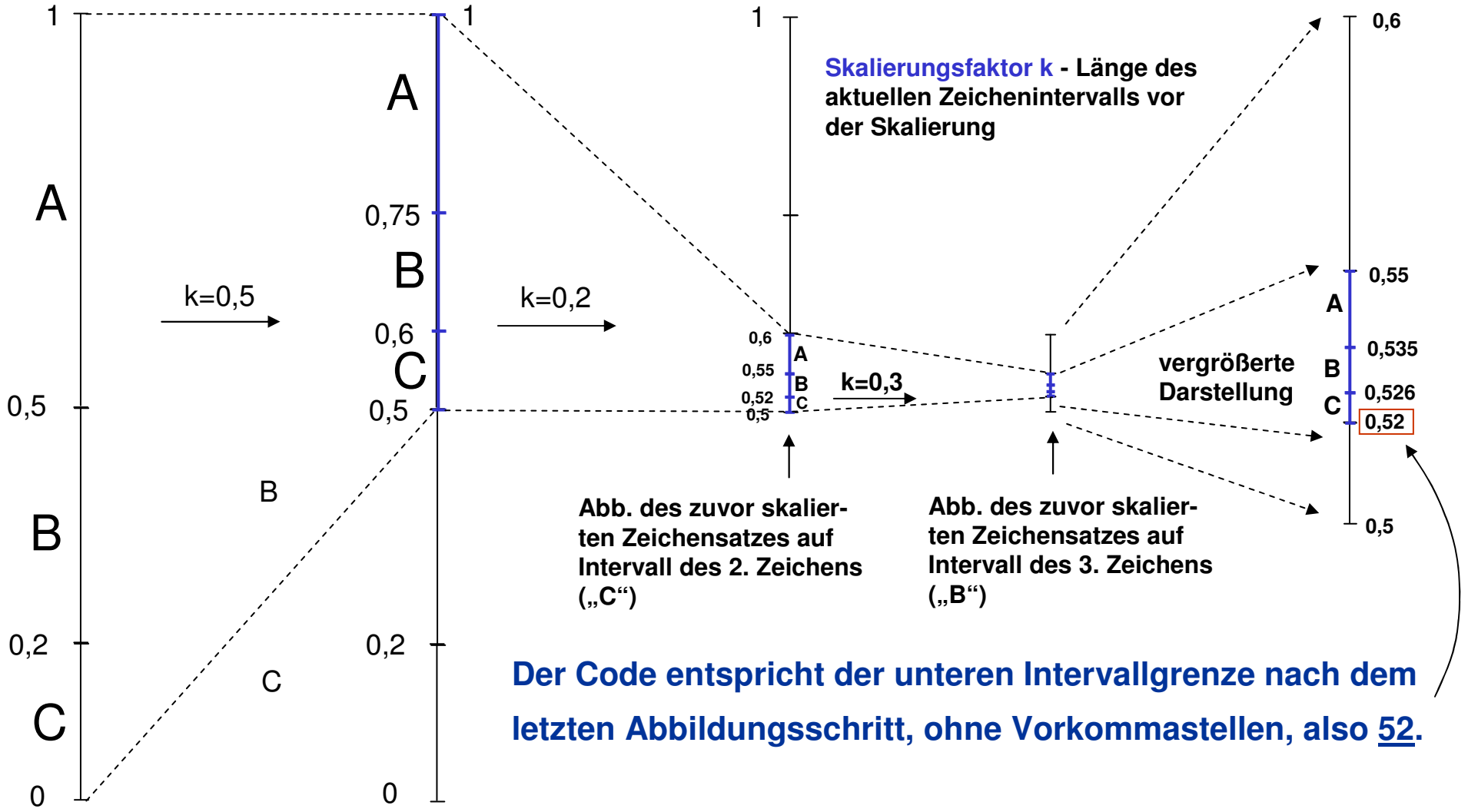



**Abb. des zuvor skalierten Zeichensatzes auf Intervall des 2. Zeichens („C“)**

**Abb. des zuvor skalierten Zeichensatzes auf Intervall des 3. Zeichens („B“)**

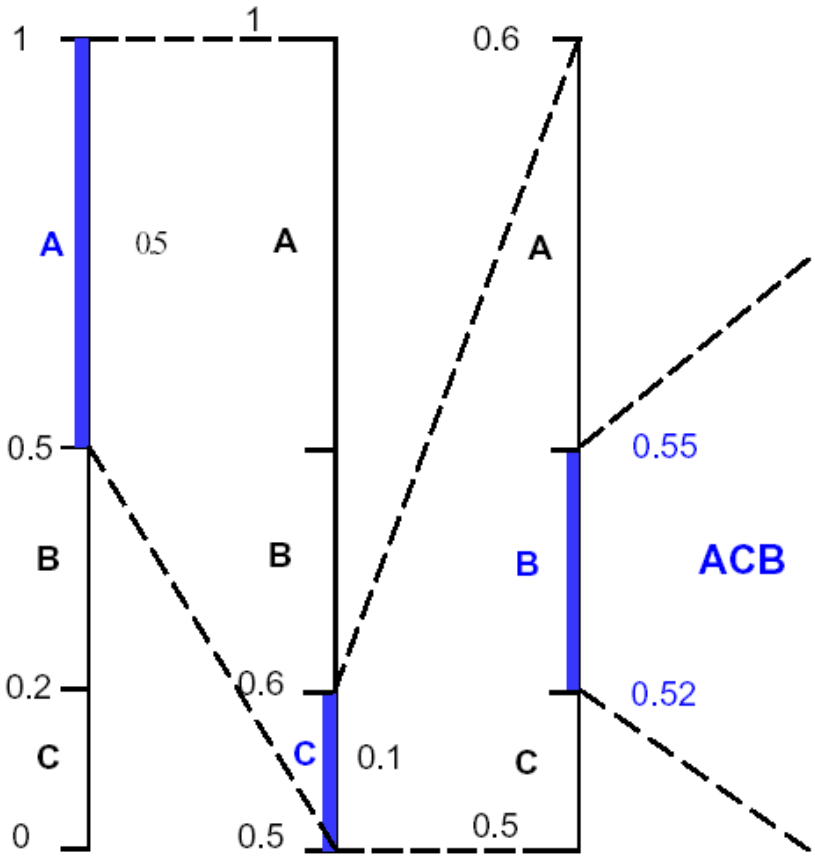
**Der Code entspricht der unteren Intervallgrenze nach dem letzten Abbildungsschritt, ohne Vorkommastellen, also 52.**

**Skala für gesamten Zeichensatz**

**Abb. des Zeichensatzes auf Intervall des 1. Zeichens („A“) mittels Verschieben und Skalierung**

# Arithmetische Codierung Beispiel 1: nochmal schematischer Ablauf, etwas anders dargestellt

## zu codierende Zeichenfolge: ‘ACB‘




Zeichenfolge ACB wird auf Teil-Intervall 0.52 - 0.55 abgebildet.

**Ergebnis:0.52 -->Code 52**

Durch Angabe eines beliebigen Wertes, der Element des Codeintervalls ist (beliebige Nachricht), läßt sich bei bekannter Wahrscheinlichkeit die Symbolfolge (Nachricht) wieder vollständig rekonstruieren.

# Beispiel 2: Codierung von: RITTER?

| Zeichen | Wahrscheinlichkeit | Intervall p |
|---|---|---|
| a | p(a) | [la,ra] |
| E | 0.5 | [0.0, 0.5] |
| I | 0.1 | [0.5, 0.6] |
| R | 0.1 | [0.6, 0.7] |
| T | 0.2 | [0.7, 0.9] |
| ? | 0.1 | [0.9, 1.0] |

l – linke (untere) Intervallgrenze

r – rechte (obere) Intervallgrenze

p - Intervallbreite (entspricht der Wahrscheinlichkeit)

***nach Anwendung Codierungalgorithmus:***

| gelesen | l | r | [la,ra] p |
|---|---|---|---|
| | 0.0 | 1.0 | 1.0 |
| R | 0.6 | 0.7 | 0.1 |
| I | 0.65 | 0.66 | 0.01 |
| T | 0.657 | 0.659 | 0.002 |
| T | 0.6584 | 0.6588 | 0.0004 |
| E | 0.6584 | 0.6586 | 0.0002 |
| R | 0.65852 | 0.65854 | 0.00002 |
| ? | 0.658538 | 0.65854 | |

***Ergebnis: l --> 0.658538***

Code: 658538

# **Beispiel 3:** Codierung von JENA

| Zeichen | Wahrscheinlichkeit | Intervall p |
|---|---|---|
| a | p(a) | $[l_a, r_a]$ |
| A | 0.5 | [0.0, 0.5] |
| E | 0.3 | [0.5, 0.8] |
| J | 0.1 | [0.8, 0.9] |
| N | 0.1 | [0.9, 1.0] |

***nach Anwendung Codierungalgorithmus:***

| gelesen | l | r | $[l_a, r_a]$ p |
|---|---|---|---|
| | 0.0 | 1.0 | 1.0 |
| J | 0.8 | 0.9 | 0.1 |
| E | 0.85 | 0.88 | 0.03 |
| N | 0,877 | 0,88 | 0,003 |
| A | 0,877 | 0,8785 | |

(0.877, 0.8785)

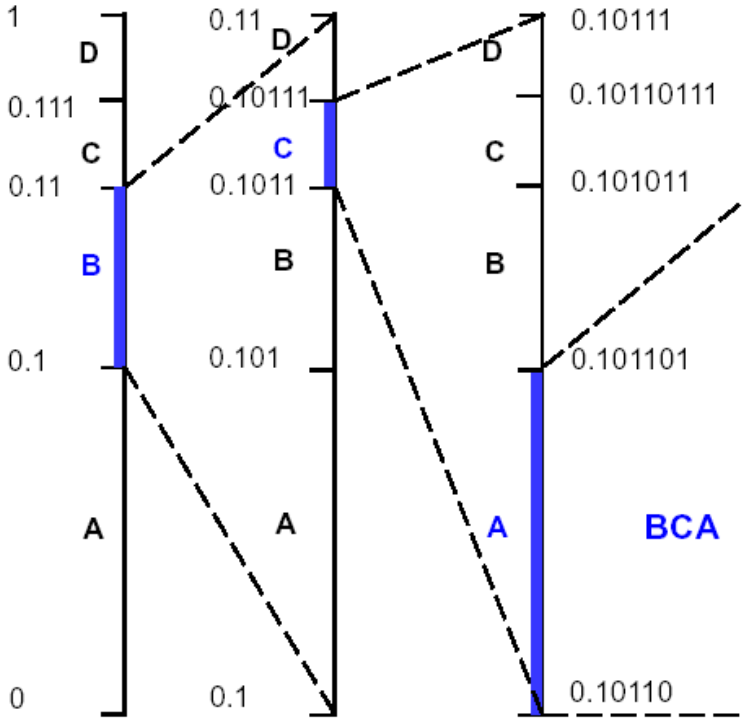
Ergebnis: l --> 0.877 Code: 877

## Beispiel(4):

Gegeben ist das Alphabet mit den Wahrscheinlichkeiten
p(A)=0.5, p(B)=0.25, p(C)=0.125, p(D)=0.125.

Gesucht: Binäre Darstellung der Sequenz BCA

| Zeichen | p(a) | p(a) Stellen/Basis2 | p(a) binär | Intervall p [la, ra] | Subintervalle von B | Subintervalle von BC |
|---|---|---|---|---|---|---|
| D | 0,125 | 2-3 | 0.001 | [0.111, 1] | [0.10111, 0.11] | [0.10110111, 0.10111] |
| C | 0,125 | 2-3 | 0.001 | [0.11, 0.111] | [0.1011, 0.10111] | [0.1011011, 0.10110111] |
| B | 0,25 | 2-2 | 0.01 | [0.1, 0.11] | [0.101, 0.1011] | [0.101101, 0.1011011] |
| A | 0,5 | 2-1 | 0.1 | [0, 0.1] | [0.1, 0.101] | [0.10110, 0.101101] |




Zeichenfolge BCA wird auf Teil-Intervall 0.10110 - 0.101101 abgebildet.
Ergebnis: 0.10110 -->

Code 10110

Vergleich: Huffmancode

| | p(g) | Huffman |
|---|---|---|
| A | 0,5 | 0 |
| B | 0,25 | 10 |
| C | 0,125 | 110 |
| D | 0,125 | 11 |

**BCA** --> Code: 101100

# 9. Transformationscodierung

# 9.1 Art und Algorithmus der Transformation

**Ansatz: Transformation des Bildes vom Ortsbereich → Ortsfrequenzbereich**

z. B. Fouriertransformation: Zeitbeich → Frequenzbereich

Bildverarbeitung: Ortbereich → Ortsfrequenzbereich → gleicher Informationsgehalt!
Effektiver, mit geringer Blockbildung als DFT ist die **DCT (Diskrete Cosinus Transformation)**

Für die 1D - DCT von 8 Signalwerten einer Zeile gilt:

$$C_{(u)} = \frac{1}{2} C_u \sum_{x=0}^{7} f(x) * \cos \frac{(2x+1)u\pi}{16}$$

*u: DCT-Koeffizienten*
*f(x): Signalwert*

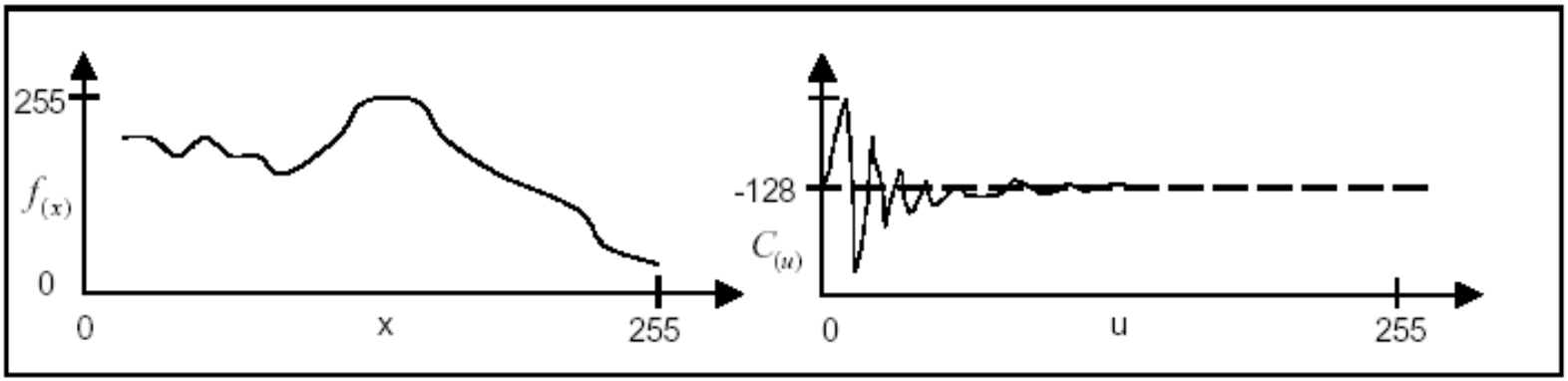
***Für die inverse DCT gilt:***

$$f_{(x)} = \frac{1}{2}\sum_{u=0}^{7} C_U * C_{(u)} * \cos\frac{(2x+1)u\pi}{16}$$

$C_U = 1/\sqrt{2}$ für u = 0

$C_U = 1$ für u = 1...7

Effekt der 1D – DCT:



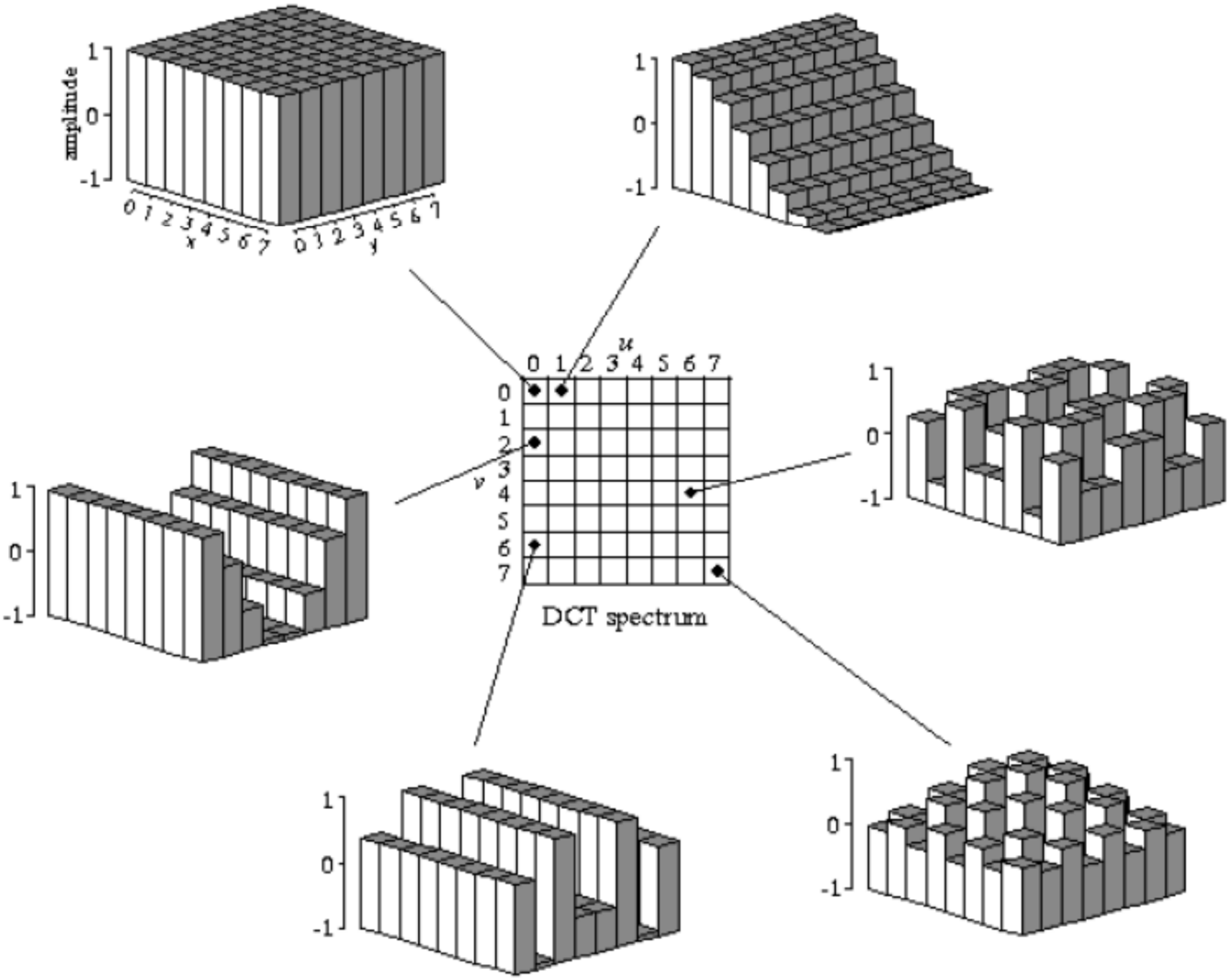

Transformation verlustlos!
Zunächst aber keine Datenreduktion!
Erst durch Quantisierung der DCT-Koeffizienten unter Ausnutzung des psychovisuellen Modell des Menschen + Anwendung der RLC und der VLC (Huffman-Codierung).
**Praxis**: Anwendung der 2D-DCT für 8 x 8 – Blöcke des Bildes.

# Beispiel für die Anwendung der DCT: JPEG – Kompression

Vergrößerung JPEG – komprimierter Bilder

| | |
|---|---|
| oben: | Original |
| Mitte, unten: | komprimiert auf 1,6% (60:1)<br>(deutlich sind die 8 x 8 Pixelblöcke<br>zu erkennen) |

# Beispiele für cosinustransformierte Pixelblöcke



# 9.2 JPEG / MPEG

**JPEG - Joint Photographics Expert Group:** Standbildkompression
**MPEG - Movie Picture Expert Group:** Bewegtbildkompression

**Joint Photographic Experts Group: Verband der Fotografiefachleute**

**JPEG ist außerdem eine gebräuchliche Bezeichnung für die Normen ISO/IEC und CCITT Recommendation T.81, welche von JPEG entwickelt wurde und verschiedene Normen der Bildkompression beschreibt.**

## Typische Vorgehensweise bei der JPEG-Kompression (kann innerhalb der durch die Normen vorgegeben Freiräume zwische Hard und Softwareherstellern differieren)

1. **Änderung des Farbmodells RGB → YUV und erste Stufe der Datenreduktion**
2. **Zerlegung des Bildes in 8x8-Blöcke, ggf. Differenzbildcodierung**
3. **DCT-Transformation der Blöcke**
4. **Gewichtung und Quantisierung der DCT-Koeffizienten (Irrelevanzreduktion → höherfrequente Koeffizienten werden geringer gewichtet)**
5. **Eindimensionale Aneinanderreihung der Koeffizienten mit zunehmender Frequenz**
6. **Lauflängencodierung (RLC)**
7. **Huffmann-Codierung oder Arithmetische Codierung des RLC-Codes (letztere ist in der Norm vorgesehen, wird aber praktisch so gut wie nicht angewendet)**

# 9.3 JPEG / MPEG: Ablauf JPEG-Codierung

## 1. Änderung des Farbmodells und Datenreduktion

### *Änderung des Farbmodells*

RGB – Modell → YUV-Modell
Y – Luminanz; U, V Chrominanz (Farbdifferenzsignale)

a) menschliches Auge kann Helligkeitsunterschiede besser auflösen als Farbunterschiede

b) Helligkeits- und Farbinformationen voneinander unabhängig verarbeiten

$$\begin{vmatrix} Y \\ U \\ V \end{vmatrix} = \begin{vmatrix} 0{,}3 & 0{,}59 & 0{,}11 \\ -0{,}17 & -0{,}33 & 0{,}5 \\ 0{,}5 & -0{,}42 & -0{,}08 \end{vmatrix} * \begin{vmatrix} R \\ G \\ B \end{vmatrix}$$

### *Reduktion*

statt Y : U : V = 4 : 4 : 4  4 : 2 : 2 (bzw. 4 : 1 : 1)

# JPEG / MPEG: Ablauf JPEG-Codierung / Forts.

## 2. Zerlegung des Bildes bzw. der YUV-Teilbilder in 8 x 8 – Blöcke (Subsampling)

**Hinweis: Bei JPEG erfolgt häufig zunächst noch eine Differenzbildcodierung jeweils benachbarter Blöcke (→ Interframe-DCT), was mit einer weiteren deutlichen Datenreduktion verbunden ist; in diesem Falle beziehen sich die folgenden Schritte jeweils auf diese "Differenz-Blöcke"**

## 3. Diskrete Cosinus - Transformation

DCT für Blöcke der Größe N x N mit den Pixelkoordinaten x,y = 0 .. N-1 und diskreten Frequenzindizes u,v = 0 .. N-1

**2D-DCT** $$F(u,v) = \frac{2 \cdot C(u) \cdot C(v)}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y) \cdot \cos \frac{(2x+1)u\pi}{2 \cdot N} \cdot \cos \frac{(2y+1)v\pi}{2 \cdot N}$$

**2D-IDCT** $$f(x,y) = \frac{2}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u) \cdot C(v) \cdot F(u,v) \cdot \cos \frac{(2x+1)u\pi}{2 \cdot N} \cdot \cos \frac{(2y+1)v\pi}{2 \cdot N}$$

**mit** $$C_u, C_v = \frac{1}{\sqrt{2}} \quad für \quad u,v = 0 \quad \textbf{und} \quad C_u, C_v = 1 \quad für \quad u,v = 1..N-1$$

# JPEG / MPEG: Ablauf JPEG-Codierung / Forts.

## 3. Diskrete Cosinus – Transformation / Forts.

**speziell für die Blockgröße 8 x 8 (übliche Größe)**

**2D-DCT**

$$F(u,v) = \frac{1}{4} C(u) \cdot C(v) \sum_{x=0}^{7} \sum_{y=0}^{7} f(x,y) \cdot \cos \frac{(2x+1)u\pi}{16} \cdot \cos \frac{(2y+1)v\pi}{16}$$

**2D-IDCT**

$$f(x,y) = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C(u) \cdot C(v) \cdot F(u,v) \cdot \cos \frac{(2x+1)u\pi}{16} \cdot \cos \frac{(2y+1)v\pi}{16}$$

**und für die Größe 4 x 4 (z. B. für Testzwecke)**

**2D-DCT**

$$F(u,v) = \frac{1}{2} C(u) \cdot C(v) \sum_{x=0}^{3} \sum_{y=0}^{3} f(x,y) \cdot \cos \frac{(2x+1)u\pi}{8} \cdot \cos \frac{(2y+1)v\pi}{8}$$

**2D-IDCT**

$$f(x,y) = \frac{1}{2} \sum_{u=0}^{3} \sum_{v=0}^{3} C(u) \cdot C(v) \cdot F(u,v) \cdot \cos \frac{(2x+1)u\pi}{8} \cdot \cos \frac{(2y+1)v\pi}{8}$$

# JPEG / MPEG: Ablauf JPEG-Codierung / Forts.

**3. DCT**

**4. Gewichtung und Quantisierung der DCT-Koeffizienten (dargestellt an einem 4 x 4-Block)**

| | | | |
|---|---|---|---|
| 110 | 112 | 109 | 111 |
| 106 | 105 | 107 | 107 |
| 98 | 97 | 97 | 95 |
| 92 | 90 | 94 | 91 |

Transformation →

| | | | |
|---|---|---|---|
| 405,3 | 0,2 | -0,2 | 1,3 |
| 29,6 | -0,5 | 0,6 | -3,3 |
| -0,7 | -0,5 | -0,3 | -0,6 |
| -2,3 | 2,2 | -0,5 | -1,5 |

DCT-Koeffizienten

↓ Quantisierung (Aufrunden auf ganzz. Werte)

| | | | |
|---|---|---|---|
| 1 | 1 | 0,9 | 0,8 |
| 1 | 0,9 | 0,8 | 0,7 |
| 0,9 | 0,8 | 0,7 | 0,6 |
| 0,8 | 0,7 | 0,6 | 0,5 |

Gewichtungsmatrix

→

| | | | |
|---|---|---|---|
| 405 | 0 | 0 | 1 |
| 30 | 0 | 0 | -2 |
| 0 | 0 | 0 | 0 |
| -2 | 2 | 0 | 0 |

Gewichtete DCT-Koeffizienten

→ RLC-Codierung

# JPEG / MPEG: Ablauf JPEG-Codierung / Forts.

## Ergänzung zur DCT

**Wegen der Separierbarket der DCT, kann diese, wie die DFT auch (siehe Vorlesungskomplex globale Operatoren) ebenfalls zeilen- und anschließend spaltenweise (oder umgekehrt) durchgeführt werden, nachfolgend dargestellt am Beispiel von Seite 37:**

x / y

| | | | |
|---|---|---|---|
| 110 | 112 | 109 | 111 |
| 106 | 105 | 107 | 107 |
| 98 | 97 | 97 | 95 |
| 92 | 90 | 94 | 91 |

Transformation der Zeilen:

$$F(u,y) = \sum_{x=0}^{3} f(x,y) \cdot \cos\frac{(2x+1)u\pi}{8}$$

u / y

| | | | |
|---|---|---|---|
| **442,0** | **0,2** | **0,0** | **-3,2** |
| **425,0** | **-1,7** | **0,7** | **1,5** |
| **387,0** | **2,8** | **-0,7** | **1,1** |
| **367,0** | **-0,6** | **-0,7** | **4,1** |

Transformation der Spalten:

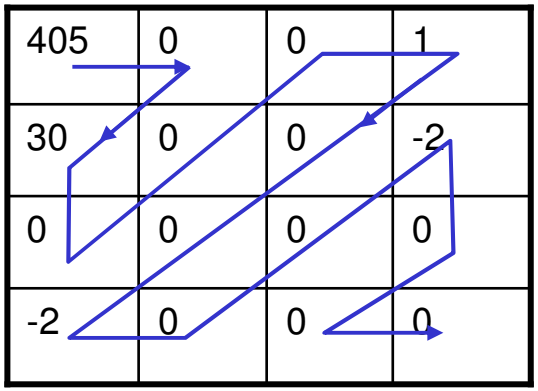
$$F(u,v) = \frac{1}{2} C(u) \cdot C(v) \cdot \sum_{y=0}^{3} F(u,y) \cdot \cos\frac{(2x+1)v\pi}{8}$$

u / v

| | | | |
|---|---|---|---|
| **405,3** | **0,2** | **-0,2** | **1,3** |
| **29,6** | **-0,5** | **0,6** | **-3,3** |
| **-0,7** | **-0,5** | **-0,3** | **-0,6** |
| **-2,3** | **2,2** | **-0,5** | **-1,5** |

# JPEG / MPEG: Ablauf JPEG-Codierung / Forts.

## 5. Eindimensionale Aneinanderreihung der gewichteten DCT-Koeffizienten
## 6. RLC-Codierung der Koeffizienten




**Eindimensionale Aneinanderreihung der gewichteten DTC-Koeffizienten durch Beschreiben eines „Zick-Zack“-Kurses, beginnend bei den kleinsten Frequenzen, aufsteigend zu höheren Frequenzen hin**

405 0 30 0 0 0 1 0 0 -2 2 0 -2 0 0 0

RLC-Codierung, z. B.: Gleichanteil, (Anzahl Nullen, nächste von 0 verschiedene Zahl, mehrere am Ende stehende Nullen werden mit dem Code für EOB abgekürzt)

405 (1,30) (3,1) (2,-2) (0,2) 1,-2 EOB

## 7. Huffman-Codierung der RLC-Koeffizienten