Digitale Bildverarbeitung

BILDCODIERUNG

Redundanzreduktion

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

Irrelevanzreduktion

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

Motivation

Vergleich Speicherplatzbedarf für Text-, Bild-, Audio-, Video-Dateien:

1. 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**

2. Farbbild 1:

Wieviel Speicherplatz benötigt ein Farbbild in PAL –Norm (768x576 Pixel) mit 8 Bit / Farbe im RGB-Modus ?

Lösung:

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

3. 1 Sekunde Video (PAL): = 1,26 Mbyte x 25 Vollbilder
= **31,6 Mbyte**

4. 1 Sekunde Audio: CD-Qualität (Abtastung 44 kHz), Stereo, 16 Bit Quantisierung
= 44 kHz x 2 Kanäle x 2 Byte
= 176 kByte

5. 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 mm^2) ; 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**

6. Farbbild 3: Geldschein 10 Euro mit 2400 ppi x 2400 ppi in RGB (8 Bit/Farbe) gescannt:
= **215 Mbyte**

7. DVD-Video

1.) 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 !**

2.) Wie hoch muß die Kompress. 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**

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

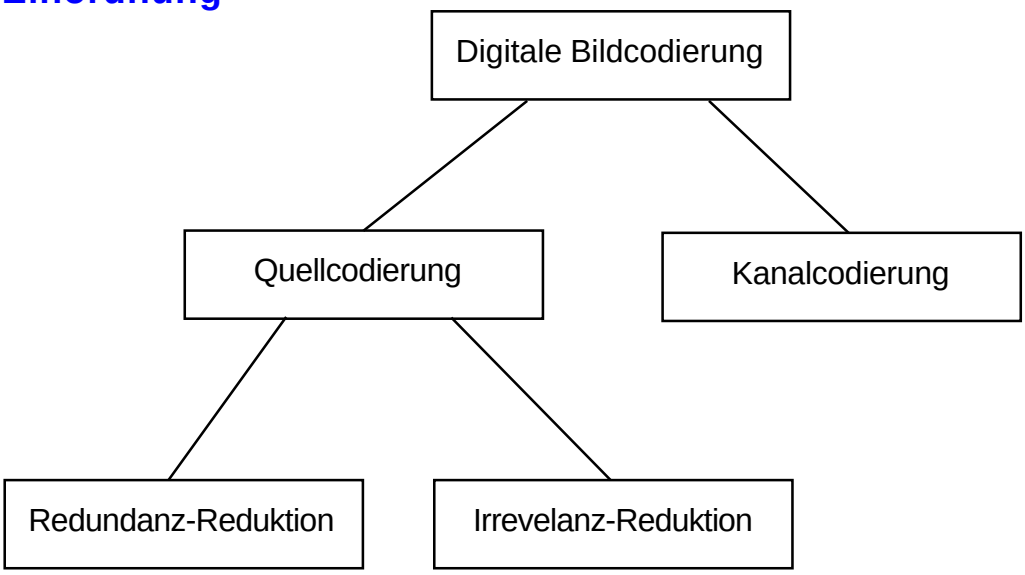
Lösung:

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) platzsparende Speicherung
b) schmalbandige Bildübertragung
c) bei Erhalt des visuellen Eindruckes
d) verlustlose Kompression bei spez. Bildern

Einordnung



Was überflüssig ist --> weg Was unwichtig ist --> weg

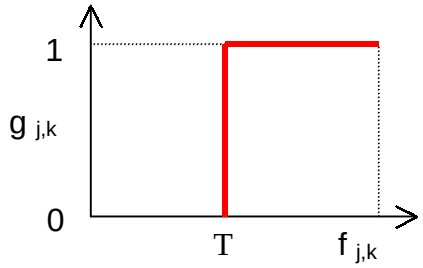
Arten der Codierung

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

a)

Beispiel:
Die 34. Zeile eines Binärbildes trägt folgende Information:

000000111110000011100000000

bessere Darstellung:

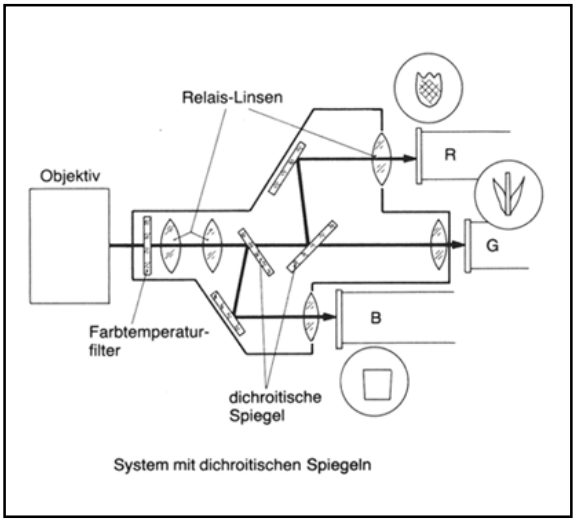
(6x0),(5x1),(5x0),(3x1),(7x0)

RLC:

(6:5),(16:3)

Verlustlos !

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

Beispiele zur Lauflängencodierung von Binärbildern

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

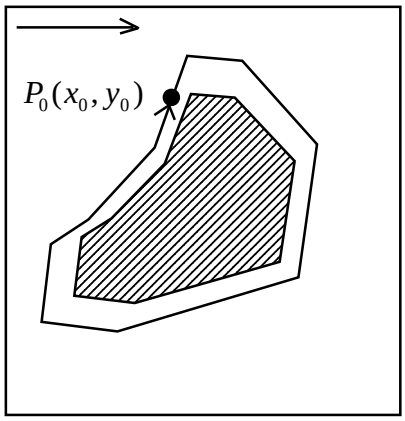
Lauflängencodierung von GW-Bildern: --> (vorlaufende Nullen, GW)

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:

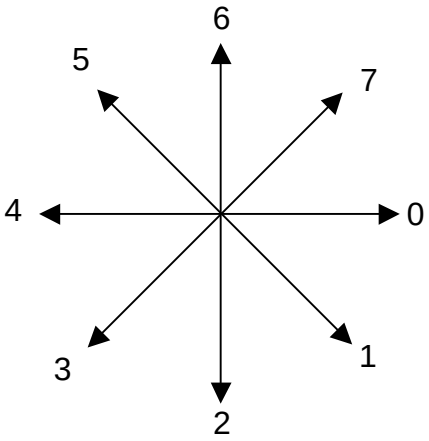
1. Durchsuchen des Bildes nach Konturpunkten
   Abstand Suchzeilen =1 oder >1

2. Erster gefundener Konturpunkt
   $f_{j,k}$ =1 --> Startpunkt
   $P_0(x_0, y_0)$

3. Konturverfolgung durch Abfrage Nachbarn
   effektiv: 1. Abfrage 4-N, wenn Objektpunkt
   dann -->
   2. Abfrage 8-N

4. Abspeicherung FREEMAN-Code

5. Algorithmus liefert Objektkontur,
   gleichzeitig können Primermerkmale generiert
   werden--> siehe Merkmalsextraktion

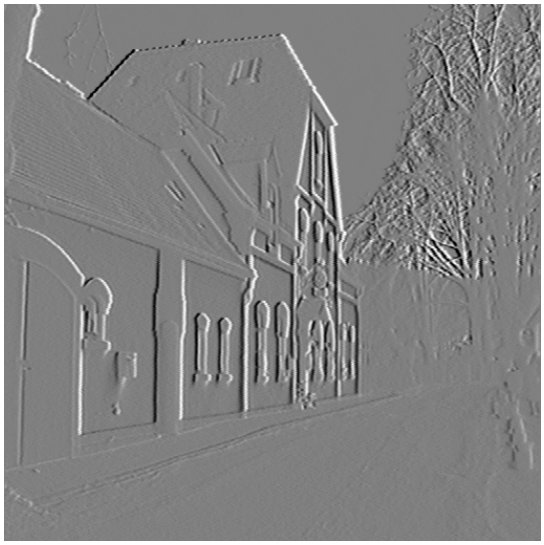


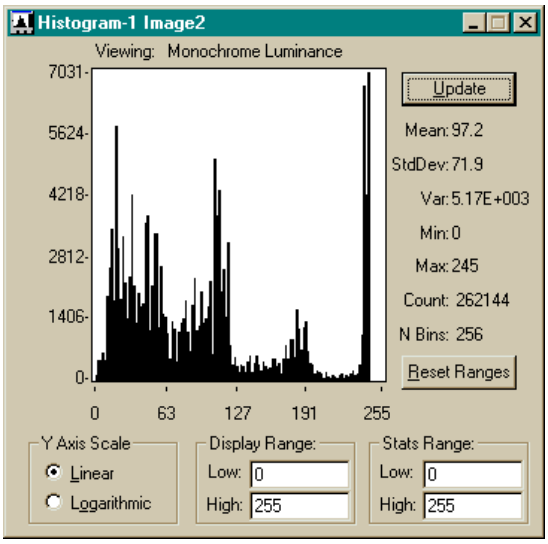
Verlustlos !

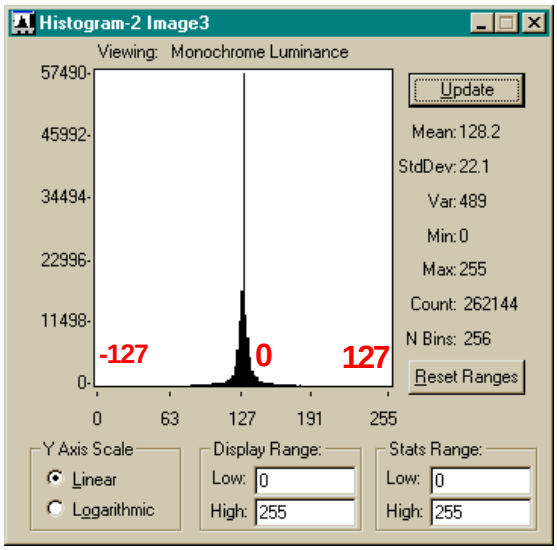
z. Beispiel:

Startpunkt P_0(55,22);
77444111222233334447 77.....

5. Differenzencodierung







Differenzen-Codierung:

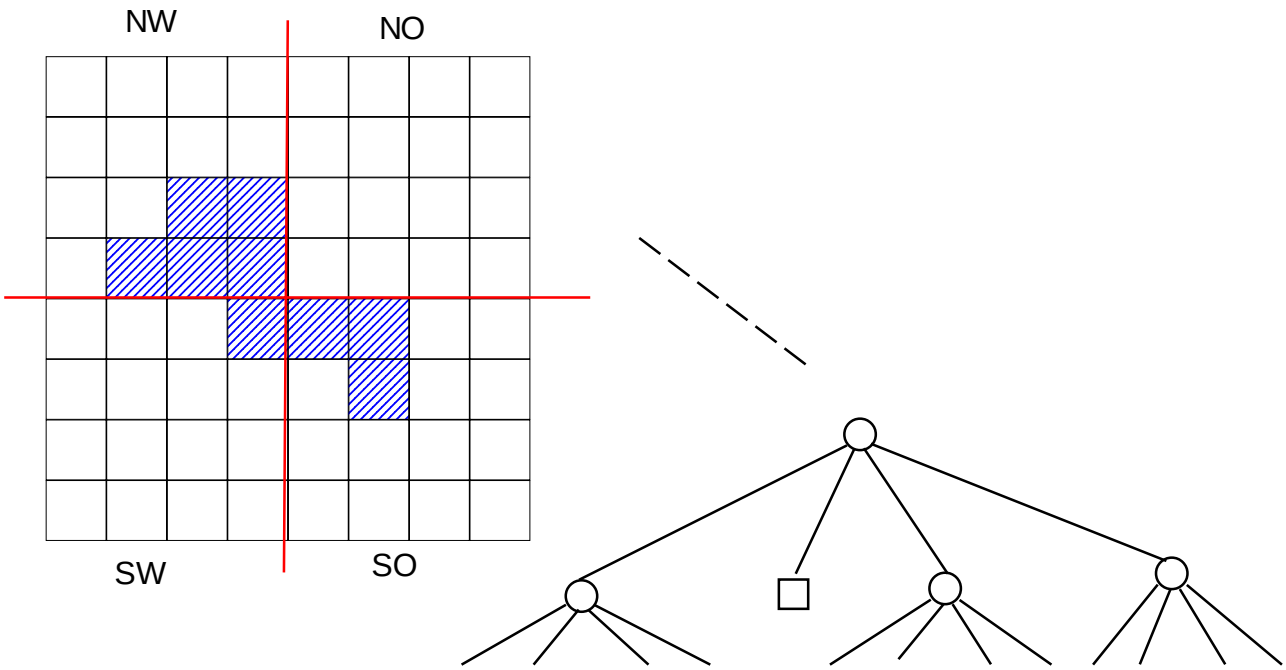
a) Original
b) Differenzbild
c) Histogramm zu a)
Entropie =7,6
d) Histogramm zu b)
Entropie = 5,8

Verlustlos !

6. Quadtree-Codierung

Prinzip:
Untersuch. des Bildes auf homogene Gebiete --> effekt. Abspeicherung dieser Gebiete

Vorschrift:
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
   -->Einteilung in 4 Teilbilder usw.



Vervollständigen Sie Codebaum und Code!

Quadtree-Notierung

1. Durchquerung des Quadtree von links --> rechts
2. Abstieg --> Klammer auf '('
3. Aufstieg --> Klammer zu ')'
4. Untergrund --> 0
5. Objektpixel -->1

Quadtree-Code:

((0 0 (...

Verlustlos !

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

Beispiel: Für ein 2 Bit -Bild (GW: 0, 1, 2, 3) gelten folgende stat. 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

Charakteristika:

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 $[0,1]$ dargestellt wird. z.B.: 0,75 --> Null weglassen --> 75
5. Codierung erfolgt in Abhängigkeit von vergangenen Zeichen und Wahrscheinlichkeit p(a).
6. Optimale (nahezu) Codierung
7. Nachteil: --> kein wahlfreier Zugriff

Idee:

1. Codiere durch ein Intervall $[l,r]$ mit $0 \le l \le r \le 1$
2. Zerlege Intervall $[0,1]$ in paarweise disjunkte Teilintervalle $[l_a, r_a]$ für jedes Zeichen a.

$r_a - l_a = p(a)$

Codierungs-Algorithmus:

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

p: Intervall

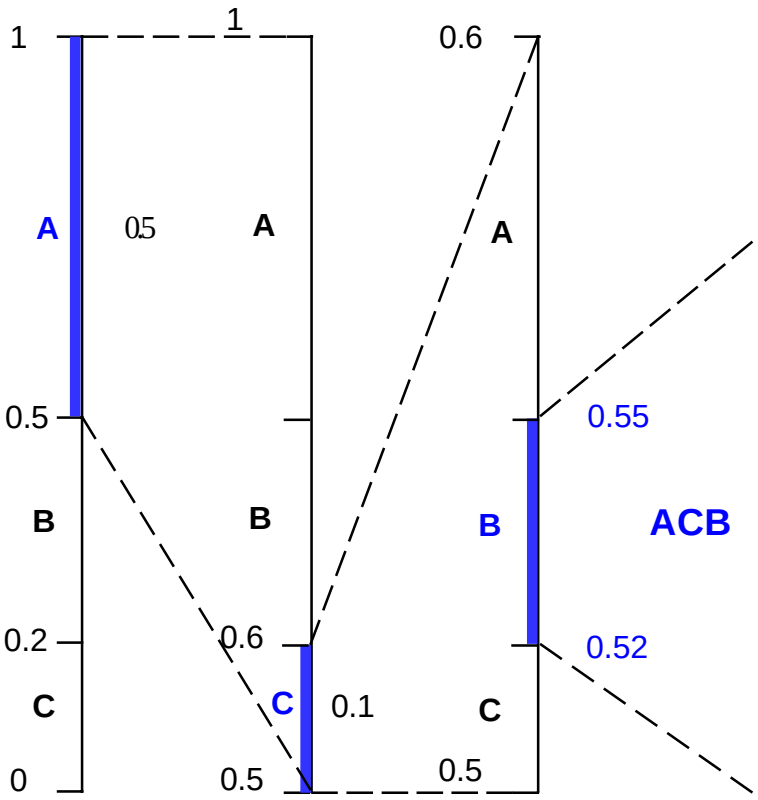
Beispiele:

Beispiel(1):
Das Alphabet besteht aus den Zeiichrn 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:

```
l:=0.0; r:=1.0;
repeat
    read (a);
    p:=r-l;
    l:=l+p*l_a;
    r:=l+p*r_a;
until a=='EOF';
Ergebnis: l;
```



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)	$[l_a, r_a]$
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]

nach Anwendung Codierungalgorithmus:

gelesen	l	r	$[l_a, r_a]$ 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	

(0.65858, 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

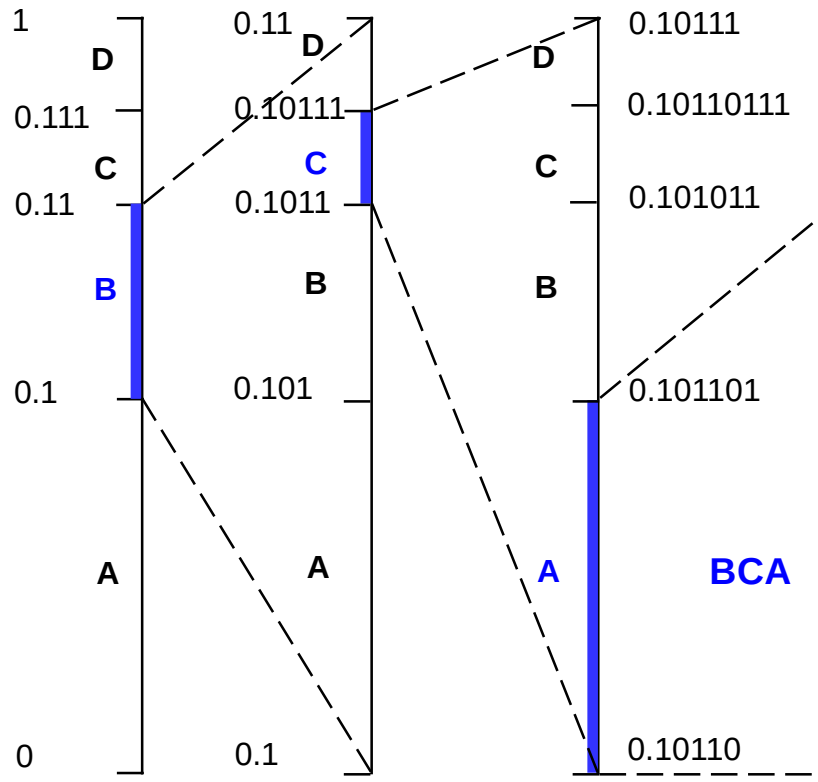
In der Praxis erfolgt die Codierung der Symbole natürlich binär:

Beispiel(4):

Gegeben ist das Alphabet {A, B, C, D} 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

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

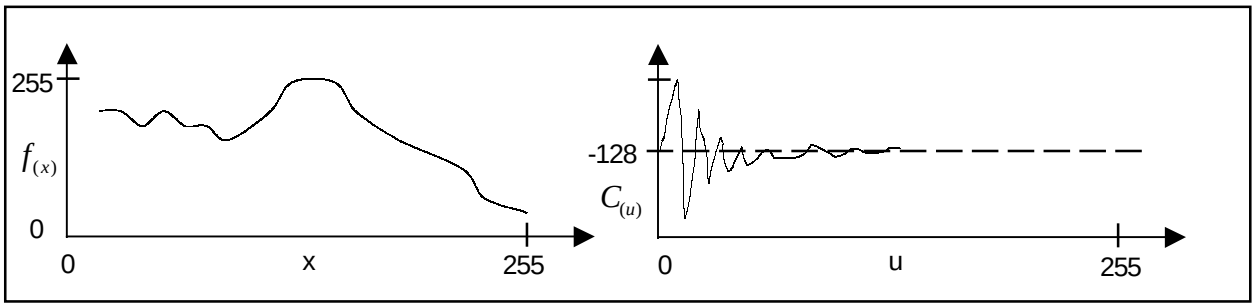
Für die inverse DCT gilt:

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

$C_U = 1/\sqrt{2}$ 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.

JPEG - Kompression

Vergrößerung JPEG - komprimierter Bilder

oben: Original
mitte:
unten: komprimiert auf 1,6% (60:1)
(deutlich sind die
8 x 8 Pixelblöcke zu erkennen)

10. JPEG / MPEG

JPEG - Joint **P**hotographics **E**xpert **G**roup

MPEG - Movie **P**icture **E**xpert **G**roup

JPEG ⟶ Standbildkompression

MPEG ⟶ Bewegtbildkompression

JPEG - Algorithmus:

1. 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

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

1a) 1. Reduktion
statt Y : U : V = 4 : 4 : 4 ⟶ 4 : 2 : 2 (bzw. 4 : 1 : 1)
⟶ Subsampling

2. Diskrete Cosinus - Transformation
DCT für Blöcke von 8 x 8

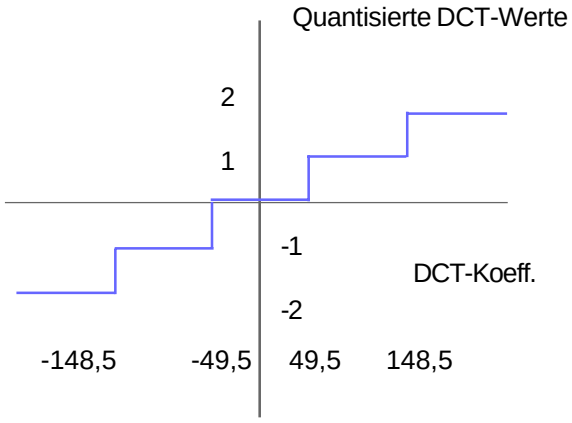
2 D - DCT $$C_{u,v} = \frac{1}{4} C_u C_v \sum_{u=0}^{7} \sum_{v=0}^{7} f_{(x,y)} \cdot \cos \frac{(2y+1)u\pi}{16} \cdot \frac{(2y+1)v\pi}{16}$$

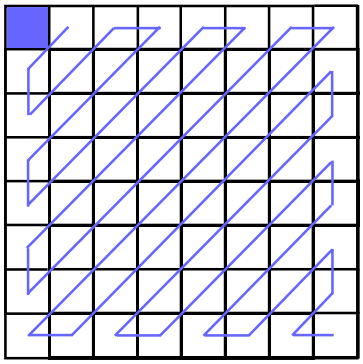
2D - IDCT $$f_{x,y} = \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} C_{(u,v)} \cdot \cos \frac{(2y+1)u\pi}{16} \cdot \frac{(2y+1)v\pi}{16}$$

mit $$C_u, C_v = \frac{1}{\sqrt{2}}$$ für u,v = 0

$$C_u, C_v = 1$$ für u,v = 1 ... 7

3. Quantisierung der DCT - Koeffizienten





DC - Koeffizienten:
Codierung nur der Differenz zum vorhergehenden Gleichanteil

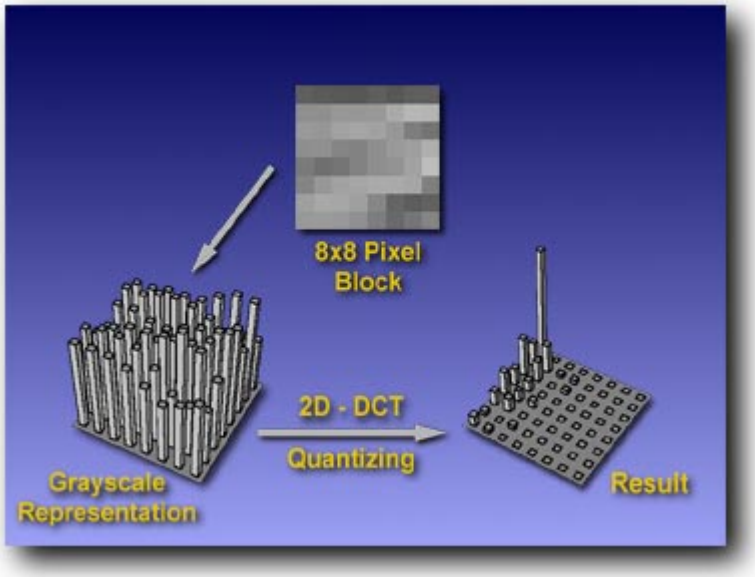
AC - Koeffizienten:
Zick - Zack - Aufreihung
(Kette aus 63 Elementen)

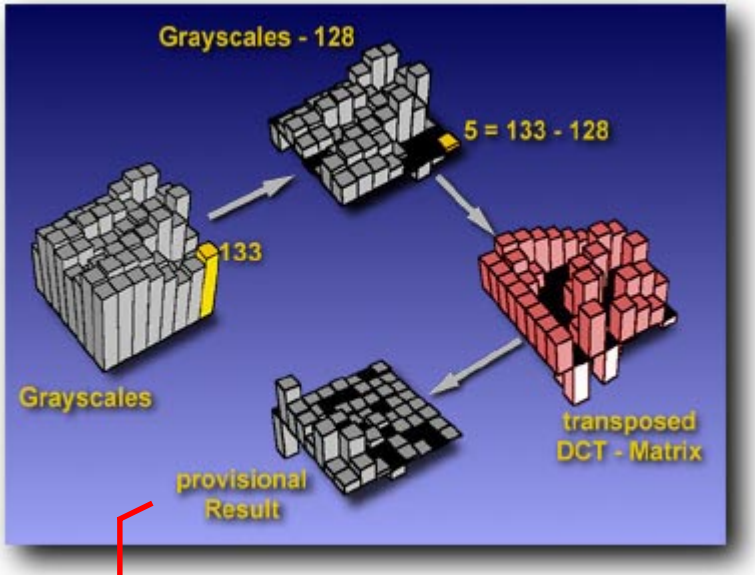
4. RLC - Codierung

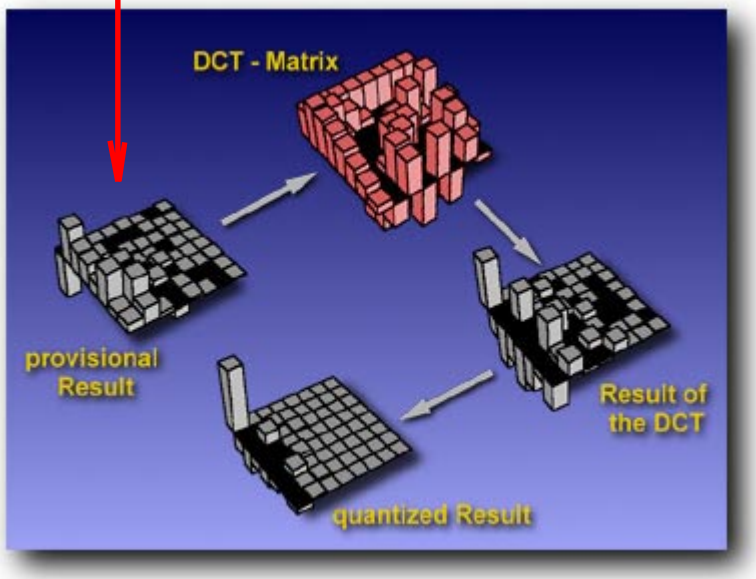
Codierung	**RRRR**	**SSSS**	
	Run-Length	Size	mehr als 16 Nullen werden mit 0 0 0 0 codiert
	RLC - nur Werte ungleich 0		Quantisierte Werte

5. Huffmann - Codierung

Ausschnitt	**RRRR**	**SSSS**	**Huffman - Code - Worte**
	0000	**0000**	**1010**
	.	**0001**	**00**
	.	**0010**	**01**
	.	**0011**	**100**
	.	**0100**	**1011**
	.	**0101**	**11010**
	.	**0110**	**1111000**







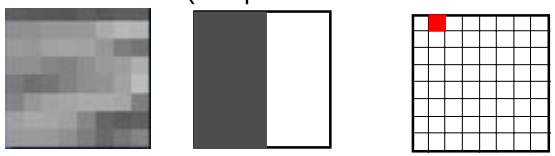
DCT- basierte Kompressionsverfahren

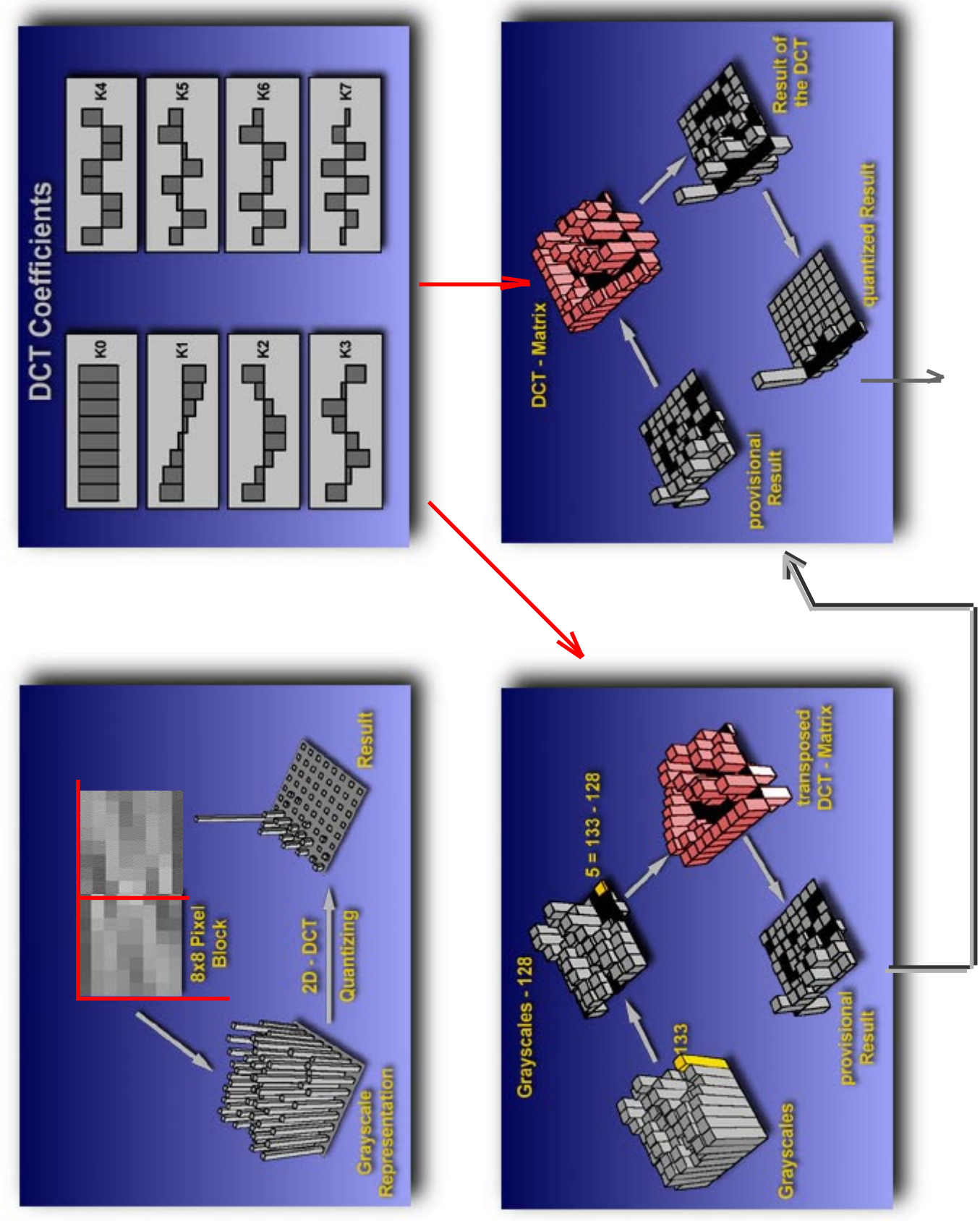
Zusammenhang zwischen Ortsbereich und Ortsfrequenzbereich eines 8x8 Teilbildes

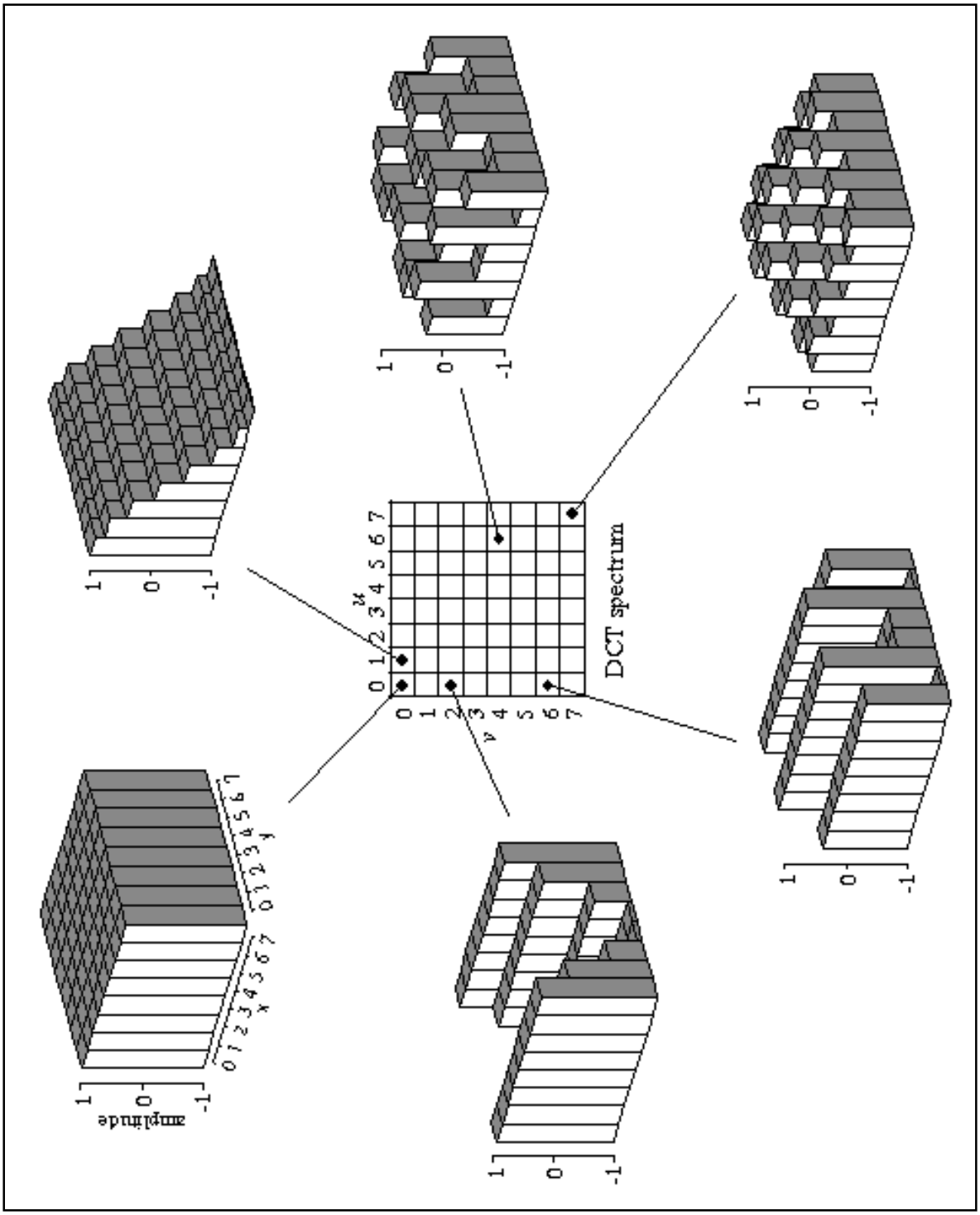
a) Redundanzreduktion (verlustfrei) mittels DCT, RLC, Huffman
b) Irrelevanzreduktion (verlustbehaftet) mittels Quantisierung
Anwendung des Psychovisuellen Modells:
- hochfrequente Bilddetails sind vom menschl. Auge schwerer aufzulösen
- optimale Linienauflösung 10 Cyclen / Grad Blickwinkel
bei 2,9 m --> T= 0,5 cm

ITU-R 601: 720 Pixel
49 cm Bildschirmdiagonale
-->39 cm / 720 --> 0,06 Pixelabstand
8 x 0,06 = 0,48 cm
(entspr. 1. Koeffizienten der DCT horizontal)

8 x 8 GW




DCT Coefficients
K0
K1
K2
K3
K4
K5
K6
K7
DCT - Matrix
Result of the DCT
quantized Result
provisional Result
8x8 Pixel Block
2D - DCT
Quantizing
Result
Grayscale Representation
Grayscales - 128
5 = 133 - 128
transposed DCT - Matrix
133
Grayscales
provisional Result



DCT-Koeffizienten