- Digitale Bilder und Dateiformate -

Seminar: Simulation und Bildanalyse mit Java

07.07.2003

Jörg Zimmermann, Daniel Meschenmoser

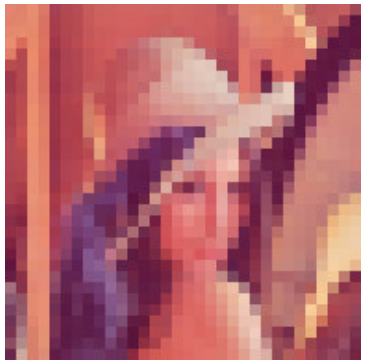
Übersicht

1. Digitalisierung
1.1 Einführung

räumliche und farbliche Diskretisierung des realen Bildes

Qualität des Bildes abhängig von der Anzahl der Bildpunkte / Auflösung

8 x 8 Pixel



32 x 32 Pixel

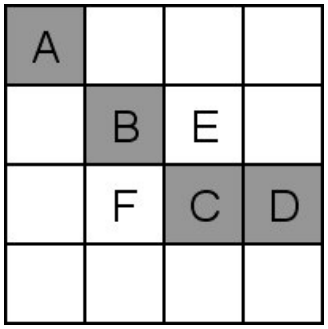
256 x 256 Pixel

... und der Anzahl der speicherbaren Farben

1.2 räumliche Diskretisierung

verschiedene Ansätze der räumlichen Diskretisierung:

(a) quadratisches Gitter



(b) hexagonales Gitter

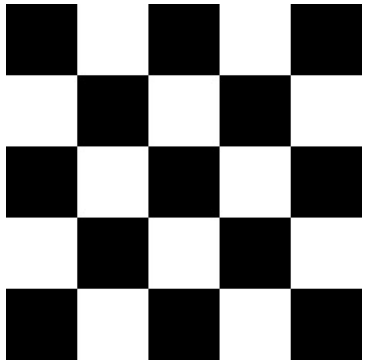
	(a) quadratisches Gitter	(b) hexagonales Gitter
Vorteile:	- gängige Darstellung - waagrechte / senkrechte Linien können gut dargestellt werden	- eindeutige Nachbarschaft - Abstände zwischen benachbarten Pixeln sind gleich
Nachteile:	- sind diagonal angrenzende Pixel benachbart? - Abstände zwischen benachbarten Pixeln sind unterschiedlich	- wird kaum von Hardware unterstützt - Darstellung von waagrechten / senkrechten Linien schlecht möglich

1.3 farbliche Diskretisierung

	Binärbild	Graustufenbild	Farbbild
			
Anzahl der Farben:	2 Farben 0 = weiß 1 = schwarz	2^n Graustufen oft 256	ca. 16,7 Mio. (= 256^3) verschiedene Farben RGB-Modell, CMY-Modell, HSI-Modell
Speicherbedarf für 1 Pixel:	1 Bit	n Bit	24 Bit
Speicherbedarf für das Bild:	256^2 / (8*1024) ≈ 8 kB	256^2 / 1024 ≈ 64 kB	256^2 * 3 / 1024 ≈ 192 kB

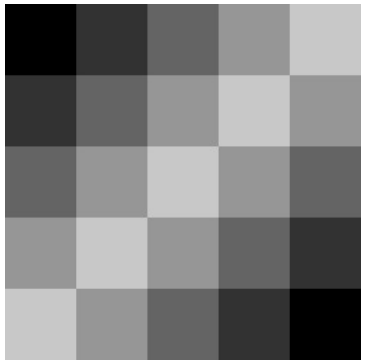
2. Bilddateiformate
2.1 PBM, PGM, PPM

PBM (Portable Bitmap)



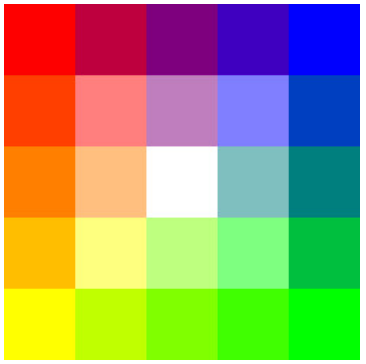
```
P1
5 5
 1 0 1 0 1
 0 1 0 1 0
 1 0 1 0 1
 0 1 0 1 0
 1 0 1 0 1
```

PGM (Portable Greymap)



```
P2
5 5
255
   0  50 100 150 200
  50 100 150 200 150
 100 150 200 150 100
 150 200 150 100  50
 200 150 100  50   0
```

PPM (Portable Pixmap)



```
P3
5 5
255
 255   0   0 191   0  63 127   0 127 ...
 255  63   0 255 127 127 191 127 191 ...
 255 127   0 255 191 127 255 255 255 ...
 255 191   0 255 255 127 191 255 127 ...
 255 255   0 191 255   0 127 255   0 ...
```

2.2 GIF (Graphics Interchange Format)

- in jeder GIF – Datei wird eine Farbtabelle mit max. 256 Einträgen gespeichert
  ➔ Es werden nur die Farben gespeichert, die am häufigsten im Bild vorkommen.
- verlustfreie Komprimierung mit Hilfe des LZW – Algorithmus
- Verwendung bei Zeichnungen, Logos, ... mit wenig Farben oder bei Bildern mit großen, einfarbigen Flächen
- typische Kompressionsrate 1/2 – 1/3

2.3 JPEG (Joint Photographic Expert Group)

- verlustbehaftete Bildkompression
- verschiedene Kompressionsgrade / Bildqualitäten einstellbar
- Verwendung bei Photos oder Bildern mit fließenden Farbübergängen
- bei starker Kompressionsrate Bildung von Artefakten, vor allem bei harten Farbübergängen
- typische Kompressionsrate 1/10 – 1/50

2.4 Vergleich GIF und JPEG

(1) photorealistisches Bild:

flagge.jpg
Größe: 11,9 kB

flagge.gif
Größe: 14,8 kB

(2) Comiczeichnung

comic.jpg
Größe: 2,1 kB
→ starke Artefakte

comic.gif
Größe: 1,1 kB

3. verlustfreie Datenkompression

3.1 Lauflängenkodierung (RLE – Run Length Encoding)

Aufeinanderfolgende identische Zeichen werden zu einem Paar (a,b) zusammengefasst, wobei a die Anzahl der aufeinanderfolgenden identischen Zeichen b ist.

Beispiel:

1) ineffizient (doppelt so viele Zeichen müssen gespeichert werden):
   ABABABABAB → (1,A),(1,B), (1,A),(1,B), (1,A),(1,B), (1,A),(1,B), (1,A),(1,B)

2) effizient:
   AAABBBBBAA → (3,A),(5,B),(2,A)

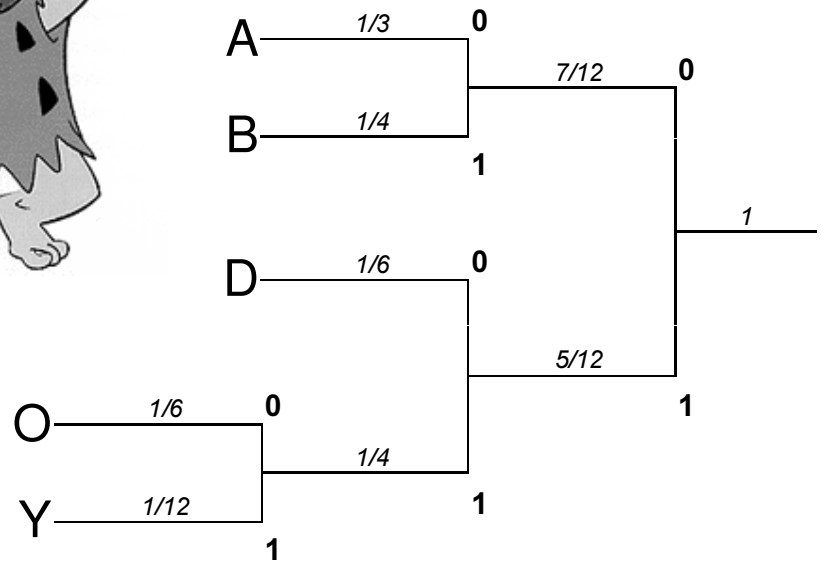
3) noch effizienter, falls nur zwei Zeichen vorkommen:
   AAABBBBBAA → 0, 3, 5, 2
   (o.B.d.A. fängt die Zeichenfolge immer mit ‚B' an)

3.2 Huffmann – Kodierung

- Kodierung der Zeichen anhand ihrer Häufigkeit
  → oft vorkommende Zeichen bekommen einen kurzen Code
  → selten auftauchende Zeichen einen längeren Code
- 2-maliger Durchlauf der Daten notwendig
  (1) relative Häufigkeiten der Zeichen ermitteln
  (2) Kodierung der Zeichen
- keine Trennzeichen im fortlaufenden Code notwendig
- nicht sehr effizient, wenn alle Zeichen etwa die gleiche Häufigkeit besitzen
- Tabelle zur Zuweisung der Codes muss mit abgespeichert werden

- Verwendung: z.B. bei der JPEG – Komprimierung

Beispiel: YABADABBADOO



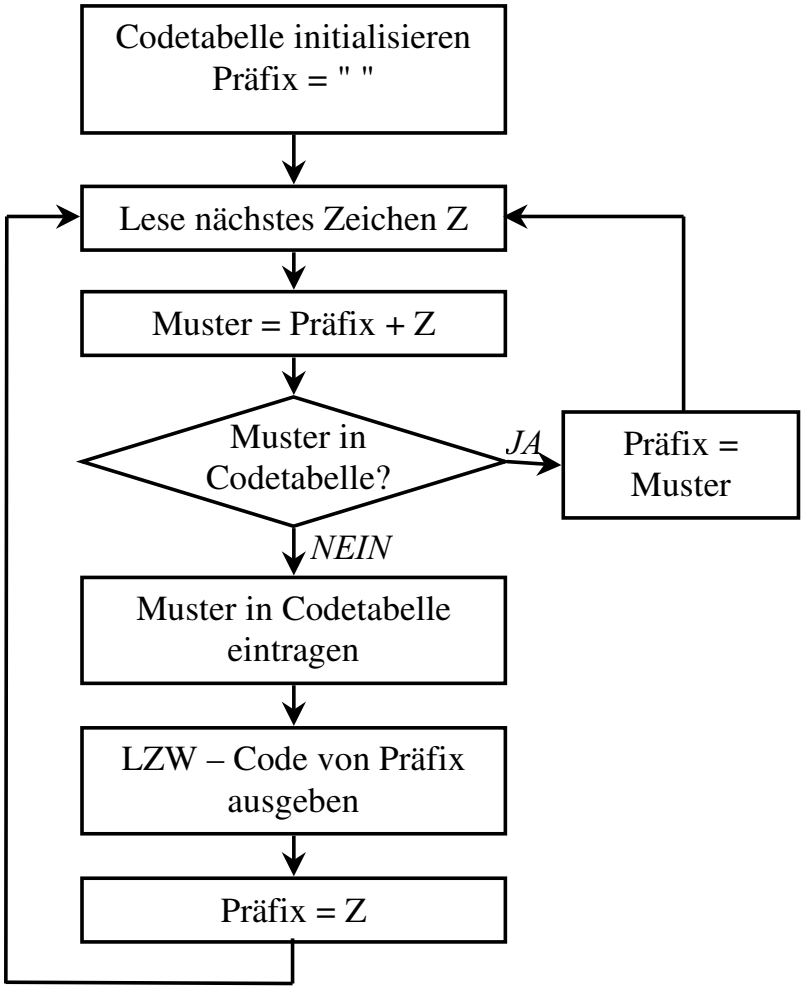
Zeichen	Häufigkeit	Code
A	1/3	00
B	1/4	01
D	1/6	10
O	1/6	110
Y	1/12	111

3.3 LZW – Algorithmus

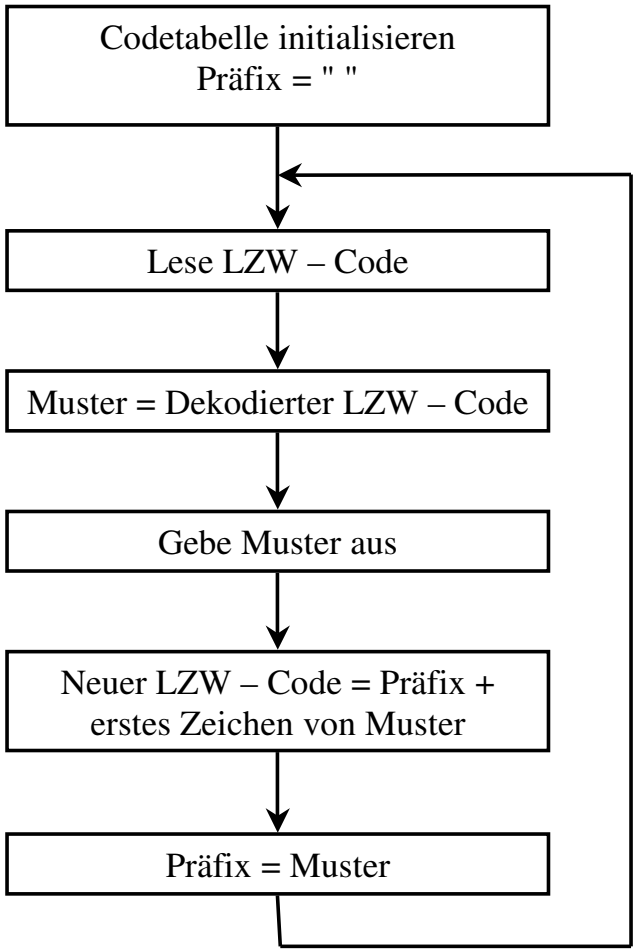
- basiert auf der Erkennung von Mustern,
  d.h. es wird nach beliebigen Zeichenfolgen gesucht, die sich wiederholen und in eine Codetabelle eingetragen werden
- neben dem codierten Datenstrom muss lediglich eine Codetabelle der Elementarzeichen abgespeichert werden
- diese wird bei der Kompression bzw. Dekompression automatisch um die vorkommenden Muster ergänzt

- Verwendung: z.B. bei der GIF – Komprimierung, aber auch bei verschiedenen anderen Komprimierungsverfahren (z.B. Zip)

Kodieren



Dekodieren



Beispiel: ABCABCABCABCD → Kodieren

Präfix	eingelesenes Zeichen	Muster	Ausgabe
-	A	A	-
A	B	AB	0
B	C	BC	1
C	A	CA	2
A	B	AB	-
AB	C	ABC	4
C	A	CA	-
CA	B	CAB	6
B	C	BC	-
BC	A	BCA	5
A	B	AB	-
AB	C	ABC	-
ABC	D	ABCD	7
D	-	-	3

LZW – Code	Zeichen
0	***A***
1	***B***
2	***C***
3	***D***
4	AB
5	BC
6	CA
7	ABC
8	CAB
9	BCA
10	ABCD

ausgegebener Code: 0 1 2 4 6 5 7 3

Beispiel: 0 1 2 4 6 5 7 3 → Dekodieren

Präfix	eingelesener LZW -Code	dekodiertes Muster	Ausgabe
-	0	A	A
A	1	B	B
B	2	C	C
C	4	AB	AB
AB	6	CA	CA
CA	5	BC	BC
BC	7	ABC	ABC
ABC	3	D	D
D	-	-	-

LZW – Code	Zeichen
0	***A***
1	***B***
2	***C***
3	***D***
4	AB
5	BC
6	CA
7	ABC
8	CAB
9	BCA
10	ABCD

ausgegebener Text: ABCABCABCABCD

4. JPEG – Kompression

4.1 Konvertierung des Farbraumes

- menschliches Auge kann feine Helligkeitsunterschiede besser wahrnehmen als leichte Farbunterschiede
  → Konvertierung des Bildes z.B. in den HSI – Farbraum
  → Farbwerte stärker komprimieren als Helligkeitswerte

Alle folgenden Schritte werden für die Helligkeits- und die beiden Farbschichten getrennt durchgeführt.

4.2 Farb – Subsampling

- erste (optionale) Komprimierung
- 2 x 2 – Farbblöcke werden zu einem gemittelten Wert zusammengefasst
- v.a. bei Photos / Bildern mit weichen Farbübergängen sinnvoll

4.3 Einteilung in 8 x 8 – Blöcke

- Einteilung des Bildes in 8 x 8 – Blöcke
- Verschiebung der Pixelwerte von [0, 255] nach [-128, 127]
- Anwendung der folgenden DCT und der Quantisierung getrennt auf jeden Block

- Beispiel:

(1) ursprünglicher 8 x 8 – Block mit Grauwerten

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	155
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	161	160	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

(2) nach der Verschiebung

11	16	21	25	27	27	27	27
16	23	25	28	31	28	28	27
22	27	32	35	30	28	28	28
31	33	34	32	32	31	31	31
31	32	33	33	32	27	27	27
33	33	33	33	32	29	29	29
34	34	33	35	34	29	29	29
34	34	33	33	35	30	30	30

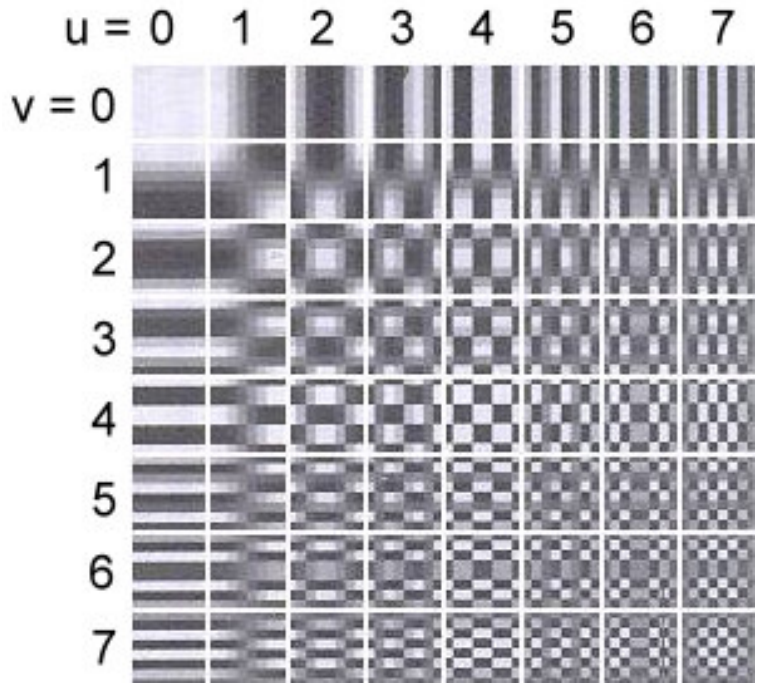
4.4 Diskrete Cosinus Transformation (DCT)

- Transformation vom Raumbereich in den Frequenzbereich zur Vorbereitung der Kompression

- Darstellung eines 8 x 8 – Blocks durch eine gewichtete Überlagerung der Basisbildern

$$b(u,v) = \cos\left(\frac{\pi \cdot u}{16}(2i+1)\right)\cos\left(\frac{\pi \cdot v}{16}(2j+1)\right)$$

mit i, j = 0,...,7 $\quad\forall$ u,v = 0,...,7



Basisbilder der DCT

mathematisch:

- 8 x 8 – Block wird als 64-dim. Vektor aufgefasst
- Basiswechsel auf eine Basis von 64 orthogonalen Vektoren, die von der JPEG – Group festgelegt sind
- Berechnung der 64 zugehörigen Koeffizienten F(u,v) durch folgende Formel:

$$F(u,v) = \frac{1}{4}\gamma_u\gamma_v \sum_{i=0}^{7}\sum_{j=0}^{7} f(i,j)\cdot\cos\left[\frac{\pi\cdot u}{16}(2i+1)\right]\cdot\cos\left[\frac{\pi\cdot v}{16}(2j+1)\right] \quad \forall u,v = 0,...,7$$

mit $\gamma_l = \begin{cases} 1/\sqrt{2}, l = 0 \\ 1, sonst \end{cases}$

und $f(i,j)$ = geshifteter Pixelwert

- Beispiel:

(2) 8 x 8 – Block nach der Verschiebung

11	16	21	25	27	27	27	27
16	23	25	28	31	28	28	27
22	27	32	35	30	28	28	28
31	33	34	32	32	31	31	31
31	32	33	33	32	27	27	27
33	33	33	33	32	29	29	29
34	34	33	35	34	29	29	29
34	34	33	33	35	30	30	30

(3) Koeffizientenmatrix nach der DCT

235,1	-0,8	-11,8	-5,1	1,6	-1,4	-2,6	1,2
-22,6	-17,3	-6,6	-3,0	-2,9	0,0	0,3	-1,1
-10,5	-9,2	-2,3	1,7	0,6	-1,1	-0,9	0,2
-7,3	-1,9	0,7	1,3	0,6	0,0	0,1	0,2
-0,9	-1,0	2,1	1,3	-0,4	-0,6	0,9	1,1
2,4	-0,5	1,3	-0,4	-0,2	1,2	0,9	-0,8
-0,9	-0,6	-0,4	-1,6	-0,1	1,6	1,1	-0,7
-3,0	1,5	-3,0	-2,1	1,5	1,3	-0,2	-0,7

- F(0,0) (= DC – Koeffizient) ist das arithmetische Mittel der ursprünglichen Bildwerte x 8 und der mit Abstand größte der 64 Werte

- alle anderen Werte (AC – Koeffizienten) werden nach rechts unten (d.h. mit höherer Frequenz) tendenziell immer kleiner und sind nahe bei 0

4.5 Quantisierung

- Dividieren der DCT – Koeffizienten durch den entsprechenden Wert aus einer Quantisierungsmatrix und ganzzahlig runden
- Verwendung verschiedener Quantisierungsmatrizen für die Farb- bzw. Helligkeitsschichten
- feinere Quantisierung der Koeffizienten der niedrigen Frequenzen (diese bestimmen die Grundstruktur des Bildes)
  → nach der Quantisierung sind die Werte der höheren Frequenzen = 0
  → stärkere Komprimierung möglich
- Standard – Quantisierungsmatrizen sind vorgegeben, es können aber auch eigene verwendet werden

- eingreifendster Schritt des Verfahrens: Bildqualität bzw. Informationsverlust durch Wahl der Quantisierungsmatrix steuerbar

- Beispiel:

Quantisierungsmatrix

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

(3) Koeffizientenmatrix nach der DCT

235,1	-0,8	-11,8	-5,1	1,6	-1,4	-2,6	1,2
-22,6	-17,3	-6,6	-3,0	-2,9	0,0	0,3	-1,1
-10,5	-9,2	-2,3	1,7	0,6	-1,1	-0,9	0,2
-7,3	-1,9	0,7	1,3	0,6	0,0	0,1	0,2
-0,9	-1,0	2,1	1,3	-0,4	-0,6	0,9	1,1
2,4	-0,5	1,3	-0,4	-0,2	1,2	0,9	-0,8
-0,9	-0,6	-0,4	-1,6	-0,1	1,6	1,1	-0,7
-3,0	1,5	-3,0	-2,1	1,5	1,3	-0,2	-0,7

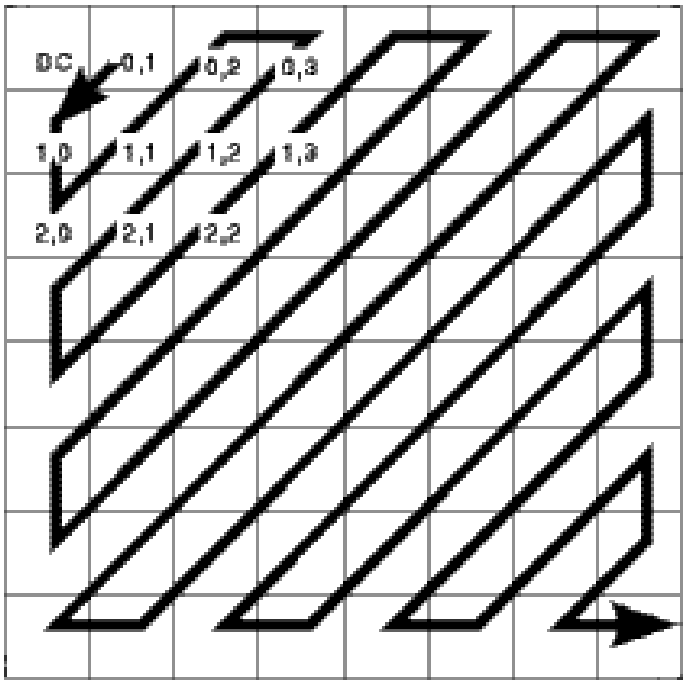
(4) Koeffizientenmatrix nach der Quantisierung

15	0	-1	0	0	0	0	0
-2	-1	0	0	0	0	0	0
-1	-1	0	0	0	0	0	0
-1	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

4.6 Kodierung

- Umwandeln der Matrix in einen Bitstrom nach dem Zick-Zack-Muster

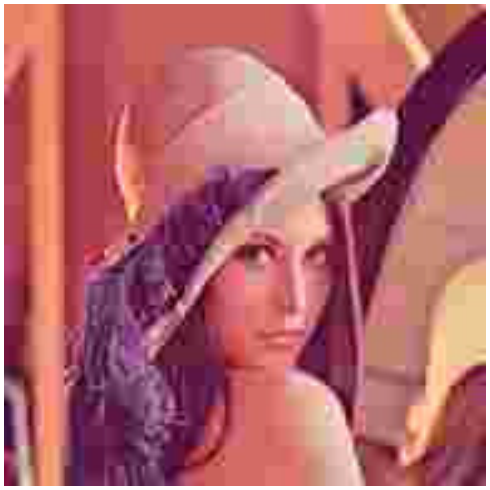
- anstelle des DC-Wertes wird die Differenz zum vorherigen DC-Wert gespeichert



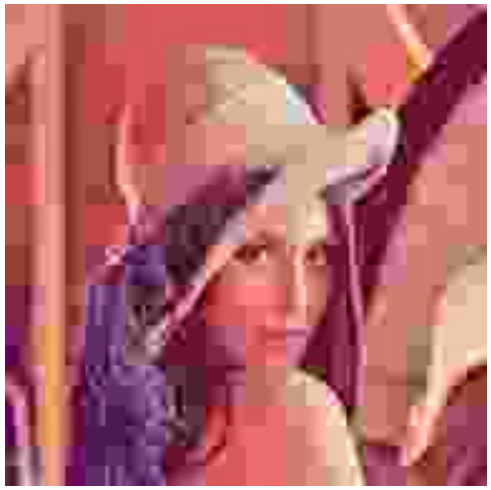
4.7 Kompression

(1) Run Length Encoding (RLE), da oft lange 0 – Folgen auftreten
(2) Huffmann – Kodierung

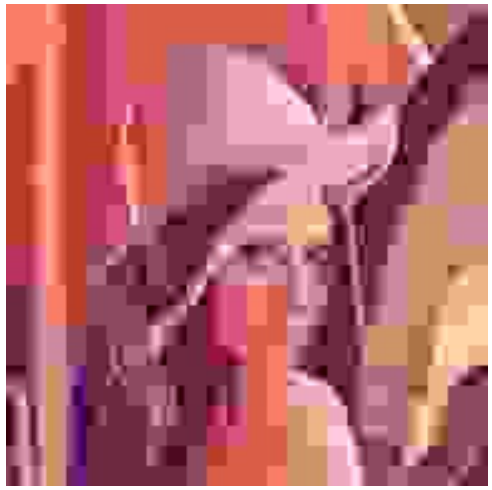
4.8 Beispielbilder



Qualitätsfaktor: 10%
Dateigröße: 1,8 kB



5%
1,2 kB



1%
0,8 kB

Fazit: bei sehr starker Kompression ist die Blockstruktur deutlich zu erkennen

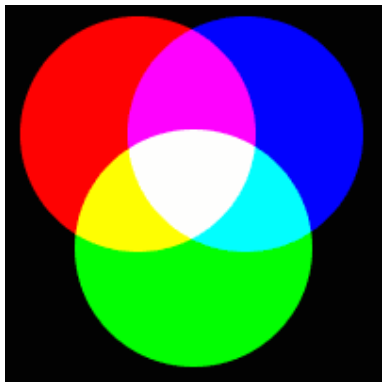
zum Vergleich: Größe des Bildes mit Qualitätsfaktor 100% : 31,9 kB

5. Literatur

- Efford: Digital Image Processing, Addison-Wesley, 2000
- Lyon: Image Processing in Java, Prentice Hall, 1999
- Specht, Kalb (Hrsg.): Dokumentenformate im Web, DBIS-32 Interne Ulmer Informatik-Berichte, Juli 2002
- http://www.faqs.org/faqs/jpeg-faq/
- http://www.wikipedia.org
- http://193.23.168.186/pc_pool/lernmodule/ multimediadateien/

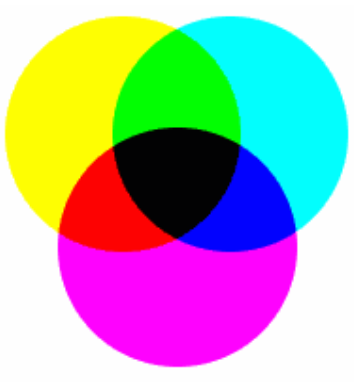
6. Anhang: Farbmodelle

RGB



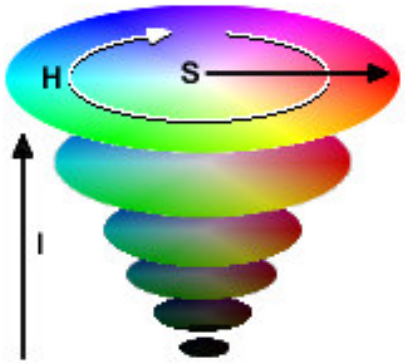
additives Farbmodell, Grundfarben: Red, Green, Blue

CMYK



subtraktives Farbmodell, Grundfarben: Cyan, Magenta, Yellow, Black

HSI



Hue (= Farbton), Saturation (= Sättigung), Intensity(= Intensität)