Kapitel 9

Bildkompression

> But life is short and information endless ...
> Abbreviation is a necessary evil and the abbreviator's business is to make the best of a job which, although intrinsically bad, is still better than nothing.
>
> Aldous Huxley

Inhalt:

- Grundlagen
- Modell von Kompressionssystemen
- Kriterien der Wiedergabetreue
- Verschiedene Kodierungsmethoden
- Kompressionsstandards (Fax und JPEG)

Grundlagen (1)

Große Menge digitaler Bilddaten $\Longrightarrow$
hohe Anforderungen an Bildspeicherung und -übertragung
(kompaktere Darstellungen jenseits der herkömmlichen Repräsentation)

$$\text{Daten} \gg \text{Information}$$

Bild- bzw. i.a. Datenkompression behandelt das Problem, *die Menge der Daten zur Repräsentation einer gegebenen Menge an (visueller) Information zu reduzieren*

Grundlagen (2)

Daten:

- Mittel zur Darstellung von Information; Informationsträger
- Existenz verschiedener Datendarstellungen für dieselbe Information
- Teile der Daten enthalten *keine* Information

Diese Eigenschaften der Daten führen zu Datenredundanz

$$\text{Kompression} = \text{Reduktion von redundanten Daten}$$

Grundlagen (2)

Daten:

- Mittel zur Darstellung von Information; Informationsträger
- Existenz verschiedener Datendarstellungen für dieselbe Information
- Teile der Daten enthalten *keine* Information

Diese Eigenschaften der Daten führen zu Datenredundanz

Kompression = Reduktion von redundanten Daten

Entropie:

Maß zur statistischen Bestimmung des Informationsgehaltes einer Informationsquelle. Besteht das Alphabet der Informationsquelle aus n Symbolen mit der Auftrittswahrscheinlichkeit $p(k)$, so ist die Entropie gegeben:

$$H_e = -\sum_{k=0}^{n} p(k) \cdot \log_2 p(k), \quad (0 \cdot \log_2 0 \equiv 0)$$

H_e erlaubt Bestimmung der minimalen Länge einer kodierten Nachricht

Grundlagen (3)

Kompressionsrate: n_1 (n_2) Datenmenge vor (nach) Kompression

$$C_r = \frac{n_1}{n_2}$$

(Eliminierte) Redundanz:

$$R_d = 1 - \frac{1}{C_r} = \frac{n_1 - n_2}{n_1}$$

Beispiel:

- $n_1 = n_2$, $C_r = 1$ und $R_d = 0$: keinerlei Redundanz
- $n_1 \gg n_2$, $C_r \to \infty$ und $R_d \to 1$: hohe Redundanz ermöglicht signifikante Reduktion der Datenmenge
- Kompressionsrate von 10 (oder 10:1): zehnmal kleinere Datenmenge nach Kompression (entsprechende Redundanz von 0.9: 90% der initialen Daten redundant)

Grundlagen (4)

Kodierungsredundanz:
Gleiche Länge von Codewörtern für alle Grauwerte (in einem natürlichen Binärcode) ist eine Verschwendung!

$l(k)$: Länge des Codewortes für Grauwert k
Durchschnittliche Anzahl benötigter Bits pro Pixel:

$$L_{\text{avg}} = \sum_{k=0}^{255} l(k) \cdot p(k)$$

Gesamte Datenmenge für ein Bild der Größe $M \times N$: MNL_{avg}

Grundlagen (4)

Kodierungsredundanz:
Gleiche Länge von Codewörtern für alle Grauwerte (in einem natürlichen Binärcode) ist eine Verschwendung!

$l(k)$: Länge des Codewortes für Grauwert k
Durchschnittliche Anzahl benötigter Bits pro Pixel:

$$L_{\text{avg}} = \sum_{k=0}^{255} l(k) \cdot p(k)$$

Gesamte Datenmenge für ein Bild der Größe $M \times N$: MNL_{avg}

Beispiel: Natürlicher Binärcode mit $l(k) = 8 \Longrightarrow L_{\text{avg}} = 8$

Grundlagen (5)

Beispiel: Nur vier Grauwerte 0–3; natürlicher Binärcode mit $l(k) = 2 \Longrightarrow L_{\text{avg}} = 2$

Alternative: Codewörter variabler Länge

k	$p(k)$	Code 1	$l(k)$	Code 2	$l(k)$
0	0.2	00	2	010	3
1	0.3	01	2	00	2
2	0.1	10	2	011	3
3	0.4	11	2	1	1

Entropie: $H_e = -[0.2 \cdot \log_2 0.2 + 0.3 \cdot \log_2 0.3 + 0.1 \cdot \log_2 0.1 + 0.4 \cdot \log_2 0.4] \approx 1.846$

Durchschnittliche Anzahl benötigter Bits pro Pixel:

$$L_{\text{avg}} = 3 \cdot 0.2 + 2 \cdot 0.3 + 3 \cdot 0.1 + 1 \cdot 0.4 = 1.9$$

Somit gilt:

$$C_r = \frac{2}{1.9} = 1.053, \quad R_d = 0.05$$

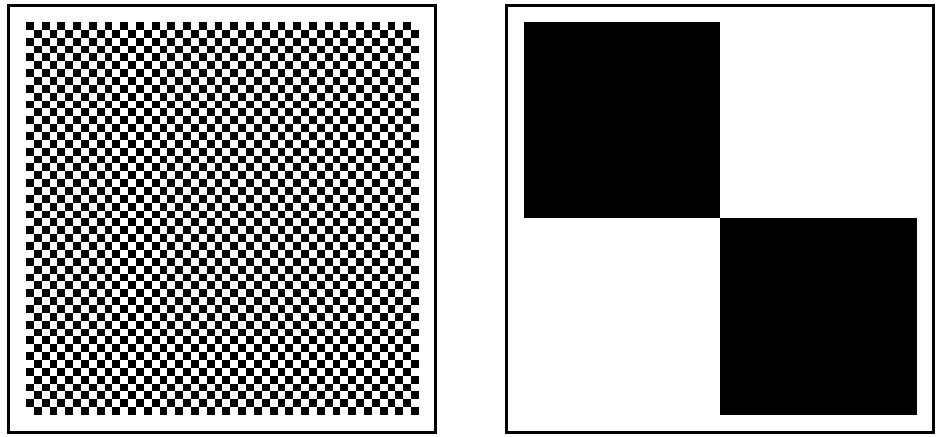
Aus dem natürlichen Binärcode resultiert (eliminierte) Redundanz von 5%

Ein Bild enthält Kodierungsredundanz, wenn dessen Kodierungsschema mehr Datenmenge als absolut nötig erzeugt. Ohne Berücksichtigung der Auftrittswahrscheinlichkeiten der Grauwerte, z.B. im natürlichen Binärcode, ist praktisch immer Kodierungsredundanz vorhanden.

Grundlagen (6)

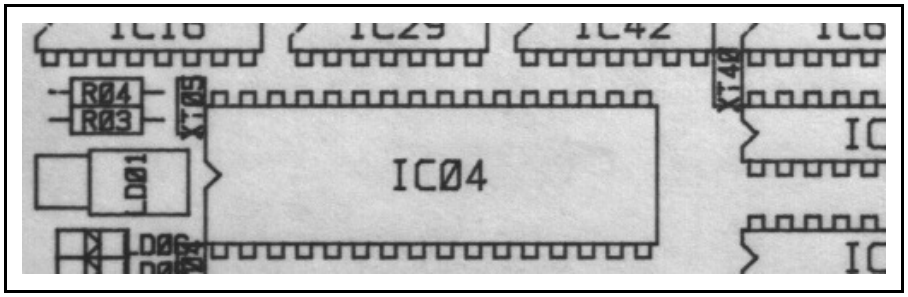
Interpixel-Redundanz:
Pixelwerte innerhalb einer kleinen Bildregion sind tendenziell ähnlich. Das Informationsgehalt einzelner Pixel ist eher gering; aus der Nachbarschaft lässt sich ein Pixelwert recht gut schätzen. Diese Ähnlichkeitseigenschaft wird Interpixel-Redundanz genannt und ermöglicht Datenreduktion.

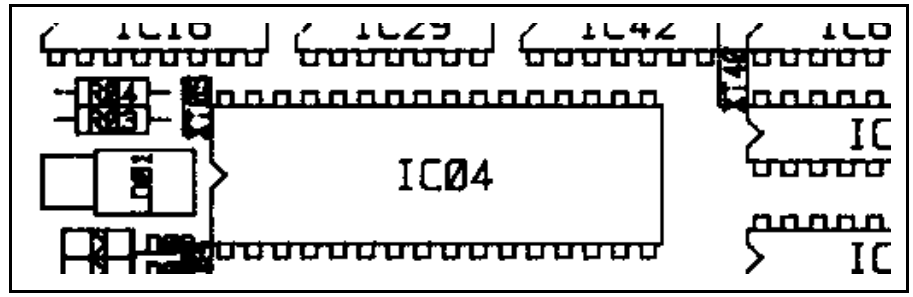
Beispiel: Identisches Histogramm. Das rechte Bild hat wesentlich größere Interpixel-Redundanz als das linke.



Grundlagen (7)

Beispiel: Binärbilder (343×1024 Pixel)





Kodierung einer Zeile (der Länge 1024):
(1,63)(0,87)(1,37)(0,5)(1,4)(0,556)(1,62)(0,210) (1024 Bits $\rightarrow$ 88 Bits)

Für das gesamte Bild:

$$C_r = \frac{1024 \cdot 343 \cdot 1 \text{ Bit}}{12166 \cdot 11 \text{ Bit}} = 2.63, \quad R_d = 1 - \frac{1}{2.63} = 0.62$$

62% Redundanz in der initialen Darstellung!

Grundlagen (8)

Beispiel: Grauwertbilder

Differenz mit rechtem Nachbarn

Differenz	Anteil
0	6.73%
1	12.71%
2	11.88%
3	10.07%
4	8.51%
5	7.00%
6	5.44%
7	4.39%
8	3.64%
9	2.90%
10	2.48%

Etwa 75% der Grauwertdifferenzen liegen unter 10

Selbst Grauwertbilder weisen Interpixel-Redundanz in einem Maß auf, das zur Datenreduktion genutzt werden kann

Grundlagen (9)

Psychovisuelle Redundanz:
Menschliches Sehsystem zeigt ungleiche Sensibilität bei visuellen Reizen (gewisse Informationen von Augen kaum oder gar nicht wahrgenommen)
$\Longrightarrow$ Wenn nur Visualisierung (aber nicht Analyse von Bildinhalten), sind diese Informationen psychovisuell redundant und können eliminiert werden
$\Longrightarrow$ Elimination/Reduktion psychovisueller Redundanz verursacht immer Informationsverlust!

Beispiel: Quantisierung der Grauwerte von 8 Bits in 4 Bits ($p^* = p \,\&\, 0xF0$)

Bitweise AND-Operation mit 11110000 (Weglassen der letzten vier Bits)
$\Longrightarrow$ Kompressionsrate $C_r = 2$; typischerweise falsche Konturen

Grundlagen (10)

Durch den sog. Machband-Effekt wird das Signal an den falschen Konturen künstlich erhöht wahrgenommen. Umso störender wirken die zahlreichen, nahezu parallel zueinander angeordneten künstlichen Kantenzüge.

Der Eindruck der quantisierten Darstellung kann verbessert werden, wenn zufallsverteiltes Rauschen den regelmäßigen Verlauf der künstlichen Kanten zerstört. Es muss jedoch dafür gesorgt werden, dass reale und Quantisierungskanten unterschiedlich behandelt werden.

$$f^*(r,c) \;=\; \text{round}\left(\frac{f(r,c)}{2^{8-q}} + \eta(r,c)\right)$$

q: Quantisierungsstufen
$\eta(r,c)$: Gleichverteilte Rauschfunktion mit Erwartungswert 0 und Maximum 0.5

Beispiel: Der Machband-Effekt im linken Bild ist störender als das leichte Rauschen im rechten Bild

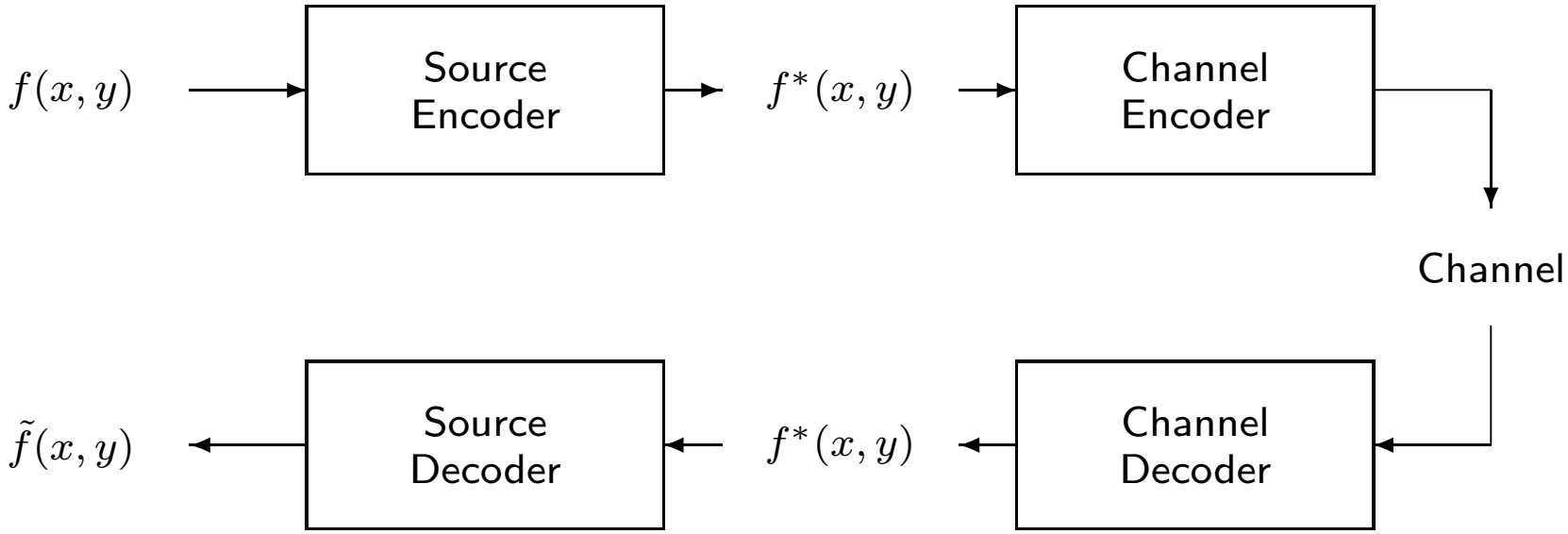
Modell von Kompressionssystemen (1)

Kompressionsverfahren lassen sich in zwei Klassen aufteilen:

- Verlustfreie Kompression (lossless compression):
  - Kostengründe (Satellitenbilder) oder rechtlichen Gründe (medizinische Bilddaten, Business Documents); in der Medizin beeinflusst der Informationsverlust die diagnostische Genauigkeit
  - Typischerweise $2 \leq C_r \leq 10$
  - Originalbild fehlerfrei rekonstruierbar
- Verlustbehaftete Kompression (lossy compression):
  - Vor allem zur Visualisierung hochgradig zulässig
  - Typischerweise $10 \leq C_r \leq 50$ mit kaum wahrnehmbaren Unterschieden zwischen Original und komprimierten Bildern
  - Originalbild nur näherungsweise rekonstruierbar

Modell von Kompressionssystemen (2)

Kompressionsmodell im Kommunikationsnetz:



Falls der Kommunikationskanal störungsfrei oder kaum störungsbehaftet ist, entfallen der Channel Encoder und der Channel Decoder

Modell von Kompressionssystemen (3)

Source Encoder:

Mapper: Transformiert Eingabedaten in neues, normalerweise *nichtvisuelles* Format (Reduktion der Interpixel-Redundanz). Dieser Schritt ist umkehrbar und kann ummittelbar sowohl weniger Daten (z.B. Kodierung von Binärbildern) oder mehr Daten (Transform Coding) produzieren. Im letzteren Fall wird die Interpixel-Redundanz besser zugänglich für nachfolgende Schritte des Kompressionsvorgangs gemacht.

Quantizer: Reduziert die Genauigkeit der Ausgabe des Mappers (Reduktion der psychovisuellen Redundanz). Dieser Schritt ist *nicht* umkehrbar und kann somit kein Bestandteil einer verlustfreien Kompression sein!

Symbol Encoder: Benutzt Codewörter (fast immer) variabler Länge zur Repräsentation der Ausgabe des Quantizers bzw. Mappers (Reduktion der Kodierungsredundanz). Dieser Schritt ist umkehrbar.

Modell von Kompressionssystemen (3)

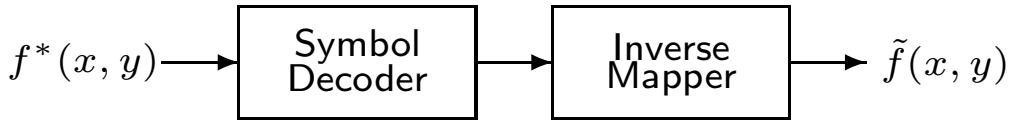
Source Encoder:

Mapper: Transformiert Eingabedaten in neues, normalerweise *nichtvisuelles* Format (Reduktion der Interpixel-Redundanz). Dieser Schritt ist umkehrbar und kann ummittelbar sowohl weniger Daten (z.B. Kodierung von Binärbildern) oder mehr Daten (Transform Coding) produzieren. Im letzteren Fall wird die Interpixel-Redundanz besser zugänglich für nachfolgende Schritte des Kompressionsvorgangs gemacht.

Quantizer: Reduziert die Genauigkeit der Ausgabe des Mappers (Reduktion der psychovisuellen Redundanz). Dieser Schritt ist *nicht* umkehrbar und kann somit kein Bestandteil einer verlustfreien Kompression sein!

Symbol Encoder: Benutzt Codewörter (fast immer) variabler Länge zur Repräsentation der Ausgabe des Quantizers bzw. Mappers (Reduktion der Kodierungsredundanz). Dieser Schritt ist umkehrbar.

Source Decoder:



Bei verlustfreier Kompression gilt $\tilde{f}(x,y) = f(x,y)$

Kriterien der Wiedergabetreue (1)

Objektive/subjektive Kriterien für Wiedergabetreue (fidelity) vei verlustbehafteter Kompression

Objektive Kriterien:

- Root-mean-square error:

$$e_{\text{rms}} = \sqrt{\frac{1}{MN}\sum_{r=1}^{M}\sum_{c=1}^{N}[f(r,c)-\tilde{f}(r,c)]^2}$$

- Mean-square signal-to-noise ratio:

$$SNR_{\text{ms}} = \frac{\sum_{r=1}^{M}\sum_{c=1}^{N}\tilde{f}(r,c)^2}{\sum_{r=1}^{M}\sum_{c=1}^{N}[f(r,c)-\tilde{f}(r,c)]^2}$$

- RMS-Wert von SNR_{ms}:

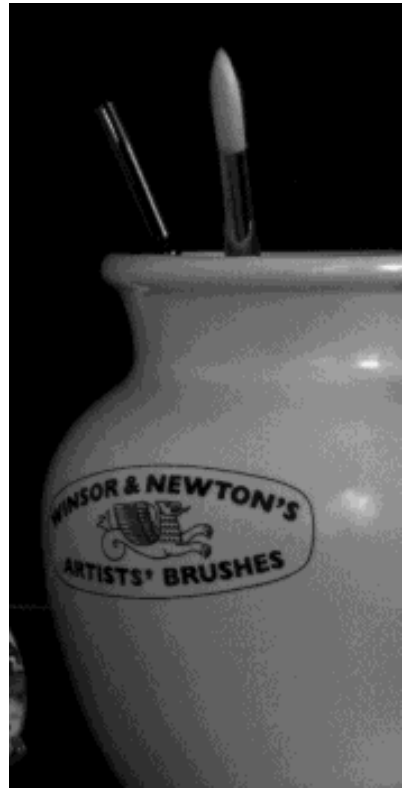
$$SNR_{rms} = \sqrt{SNR_{\text{ms}}}$$

- Peak signal-to-noise ratio: L = Anzahl Grauwerte (256 für 8 Bits)

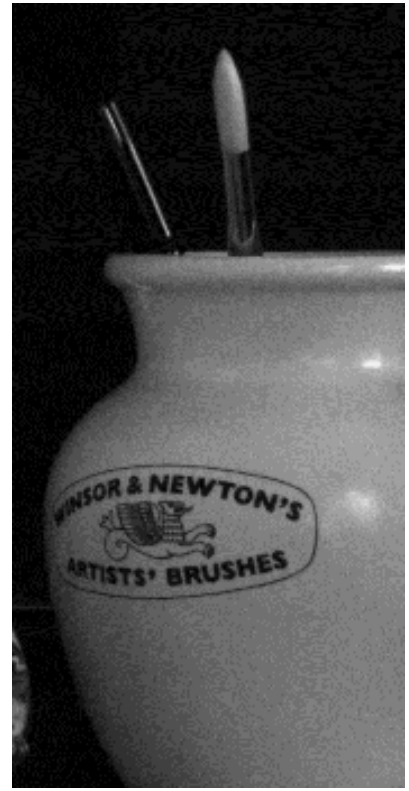
$$SNR_{\text{peak}} = 10 \cdot \log_{10} \frac{(L-1)^2}{\frac{1}{MN}\sum_{r=1}^{M}\sum_{c=1}^{N}[f(r,c)-\tilde{f}(r,c)]^2}$$

Kriterien der Wiedergabetreue (2)

Beispiel:



Quantisierung (4 Bits)
$e_{rms} = 6.93$
$SNR_{rms} = 10.25$



IGS (4 Bits)
$e_{rms} = 6.78$
$SNR_{rms} = 10.39$

IGS (Improved Gray-Scale quantization) ist eine weitere Methode, um den durch Quantisierung verursachten Machband-Effekt zu reduzieren

Kriterien der Wiedergabetreue (3)

Subjektive Kriterien:
Urteil durch Testpersonen an Bildsammlung (statistische Auswertung)

Impairment	Quality	Comparison
5 – imperceptible	A – excellent	+2 much better
4 – perceptible, not annoying	B – good	+1 better
3 – somewhat annoying	C – fair	0 the same
2 – severely annoying	D – poor	-1 worse
1 – unusable	E – bad	-1 much worse

Variable-Length Coding: Huffman Code (1)

Mit dem Symbol Encoder werden die Ergebnisse des Quantizers bzw. Mappers im Fall der verlustfreien Kompression *kodiert* dargestellt (Reduktion der Kodierungsredundanz)

Typischerweise Codewörter variabler Länge mit Eigenschaft:

Zeichen mit größerer Auftrittswahrscheinlichkeit

$\downarrow$

kürzere Codewörter

Huffman Code gehört zu den populärsten Kodierungen dieser Klasse

Variable-Length Coding: Huffman Code (2)

Konstruktion von Huffman Code

1. Ordne Auftrittswahrscheinlichkeiten der Symbole steigend

Wiederhole 2.1 und 2.2 bis Symbolliste nur zwei Elemente enthält

2.1. Bilde aus den beiden Symbolen s_1 und s_2 mit den kleinsten Auftrittswahrscheinlichkeiten $p(s_1)$ und $p(s_2)$ ein Zwischensymbol s_{12} mit der Wahrscheinlichkeit $p(s_1) + p(s_2)$

2.2 Entferne s_1 und s_2 aus der Symbolliste; Nimm das Symbol s_{12} in die Liste auf, so dass sie sortiert bleibt

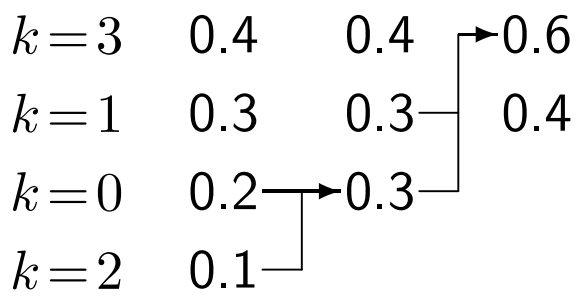
3. Repräsentiere den obigen Prozess mit einem binären Baum und markiere für jeden Knoten die beiden ausgehenden Kanten mit 0 und 1. Der Pfad von der Wurzel bis zu einem Blattknoten liefert das Codewort des entsprechenden initialen Symbols.

Variable-Length Coding: Huffman Code (3)

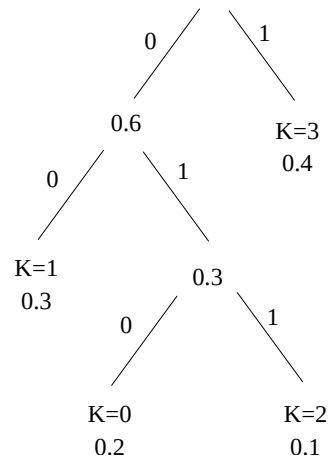
Beispiel: vgl. Folie 6. Vier Grauwerte 0–3 mit Auftrittswahrscheinlichkeiten

Grauwert k	0	1	2	3
$p(k)$	0.2	0.3	0.1	0.4
Huffman Code	010	00	011	1

Konstruktionsprozess



Binärer Baum



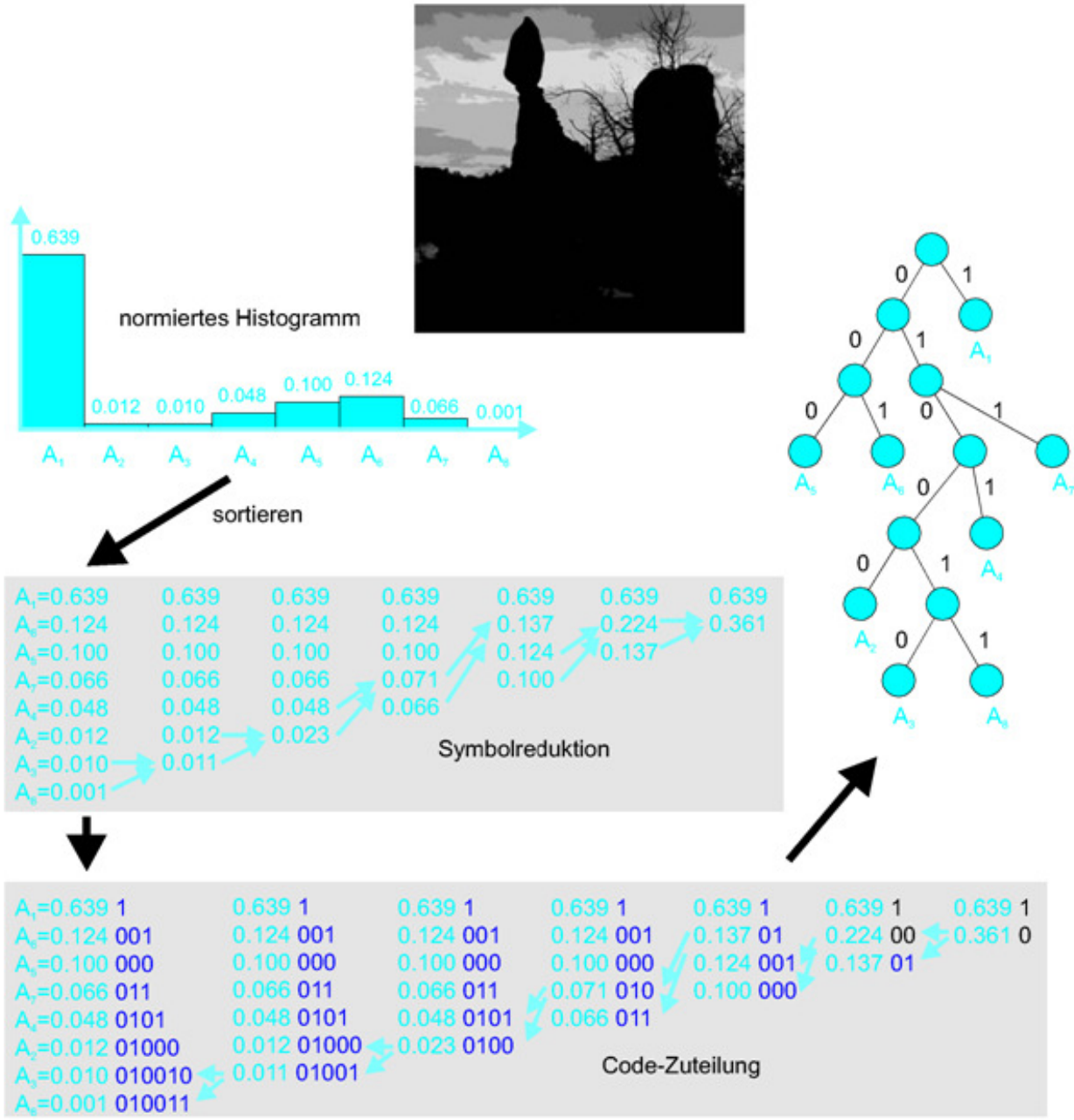
$L_{\text{avg}} = 1.9$ im Gegensatz zu $L_{\text{avg}} = 2.0$ beim natürlichen Binärcode $\Longrightarrow$
$C_r = 1.053,\ R_d = 0.05$

Decodierung (eindeutig):

$$\underbrace{1}_{3}\ \underbrace{011}_{2}\ \underbrace{00}_{1}\ \underbrace{010}_{0}\ \underbrace{011}_{2}\ \underbrace{1}_{3}\ \underbrace{00}_{1}\ \underbrace{010}_{0}$$

Variable-Length Coding: Huffman Code (4)

Beispiel:



Fixed-Length Code: LZW-Coding (1)

Statistische Modellierung der Informationsquelle (Auftrittswahrscheinlichkeiten der Symbole) nicht immer verfügbar. Beim LZW (Lempel-Ziv-Welch) Coding wird sie deshalb nicht vorausgesetzt; statt dessen wird eine Kodierungstabelle (dictionary) während des Kodierungsprozesses aufgebaut.

Konstruktion von LZW Code

Annahme: Alphabet der Informationsquelle besteht aus n Symbolen s_k $(1 \leq k \leq n)$

1. Initialisiere die ersten n Einträge der Kodierungstabelle mit s_k

Index	1	2	$\cdots$	n
Eintrag	s_1	s_2	$\cdots$	s_n

2. Finde die längste Sequenz w von Eingabesymbolen, für die es einen entsprechenden Eintrag in der Tabelle gibt

3. Kodiere w mit dem Index des Eintrags.

4. Füge am Ende der Tabelle einen neuen Eintrag ws hinzu, wobei s das Eingabesymbol unmittelbar hinter w ist

5. Gehe zurück zu Schritt 2 zur Kodierung der restlichen Eingabesymbole mit s als Start bis die gesamte Eingabesequenz kodiert ist

Fixed-Length Code: LZW-Coding (2)

Beispiel: Alphabet $\{s_1, s_2, s_3, s_4\}$; Kodierungsprozess für $s_1 s_2 s_1 s_2 s_3 s_2 s_1 s_2$ (Code: 125362)

w	Code	ws	Index	Eintrag
			1	s_1
			2	s_2
			3	s_3
			4	s_4
s_1	1	$s_1 s_2$	5	$s_1 s_2$
s_2	2	$s_2 s_1$	6	$s_2 s_1$
$s_1 s_2$	5	$s_1 s_2 s_3$	7	$s_1 s_2 s_3$
s_3	3	$s_3 s_2$	8	$s_3 s_2$
$s_2 s_1$	6	$s_2 s_1 s_2$	9	$s_2 s_1 s_2$
s_2	2			

Die Kodierungstabelle wird *nicht* abgespeichert. Der Dekodierungsprozess ist dem Kodierungsprozess ähnlich und baut exakt dieselbe Tabelle auf!

LZW Coding nur für lange Sequenzen sinnvoll (bei kurzen Sequenzen vergrößert sich häufig sogar die Datenmenge, z.B. $s_1 s_2 s_3 s_4 s_1 s_3 s_1 s_4 s_2 s_4 s_3$)

LZW Coding stellt ein allgemeines Verfahren zur Reduktion der Kodierungsredundanz zur Verfügung. Neben zahlreichen Anwendungen in Bildkompression (z.B. GIF) auch die Grundlage zur Komprimierung anderer Datentypen, z.B. für Unix Befehle wie `compress/gzip`.

Run-Length Coding (1)

Run-Length Coding ist effektiv für Reduktion der Interpixel-Redundanz in Binärbildern. Codierung einer Bildzeile:

$$(w, l_1)(b, l_2)(w, l_3)\ldots$$
$$(b, l_1)(w, l_2)(b, l_3)\ldots$$

oder

$$l_1 l_2 l_3 \ldots$$
$$[\equiv (w, l_1)(b, l_2)(w, l_3)\ldots]$$

(Annahme: Bildzeile beginnt immer mit l_1 mal Weiß, ggf. $l_1 = 0$)
Sondersymbol EOL zur Markierung des Zeilenendes

Typischerweise wird diese Repräsentation anschließend mit einem Variable-Length Code, z.B. Huffman Code, kodiert. Dieses Schema bildet die Grundlage für die Faxübertragung.

Es existiert auch ein zwei-dimensionales Run-Length Coding, bei dem die Redundanz benachbarter Bildzeilen untersucht werden

Run-Length Coding (2)

Bit-Plane Coding: Anwendung an Grauwertbildern; Zerlegung in einzelne (binäre) Bildebenen

Problem: viele Übergänge $0 \longrightarrow 1$ bzw. $1 \longrightarrow 0$ auch bei kleinen Veränderungen der Grauwerte

$$\text{Binärcode:} \quad 127\ (01111111) \longrightarrow 128\ (10000000)$$

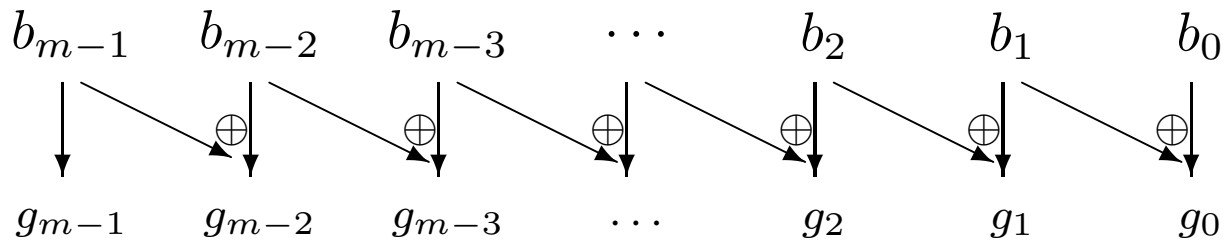
Gray-Code:
Zur Kodierung einzelner Bildebenen ist der Gray-Code (nach Frank Gray) besser geeignet als der natürliche Binärcode

Umwandlung:
Binärcode $b_{m-1}b_{m-2}\ldots b_1 b_0 \longrightarrow$ Gray-Code $g_{m-1}g_{m-2}\ldots g_1 g_0$

$$\begin{aligned} g_{m-1} &= b_{m-1} \\ g_k &= b_k \oplus b_{k+1}, \quad 0 \leq k \leq m-2 \end{aligned}$$

($\oplus$: exclusive OR Operation)

Run-Length Coding (3)



4-Bit Gray-Code

Grauwert	Binärcode	Gray-Code	Grauwert	Binärcode	Gray-Code
0	0000	0000	8	1000	1100
1	0001	0001	9	1001	1101
2	0010	0011	10	1010	1111
3	0011	0010	11	1011	1110
4	0100	0110	12	1100	1010
5	0101	0111	13	1101	1011
6	0110	0101	14	1110	1001
7	0111	0100	15	1111	1000

Eigenschaft: Zwei Grauwerte G und $G+1$ unterscheiden sich lediglich in einem einzigen Bit, z.B.

$$\text{Gray Code:} \quad 127\ (01000000) \implies 128\ (11000000)$$

Run-Length Coding (4)

Beispiel: Bitebenen Binärcode vs. Gray Code

b_7: Bildebene 7

b_7: Bildebene 6

b_7: Bildebene 5

b_7: Bildebene 4

g_7: Bildebene 7 ($\equiv b_7$)

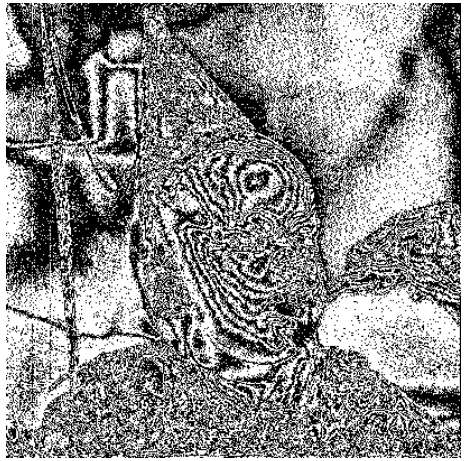
g_7: Bildebene 6

g_7: Bildebene 5

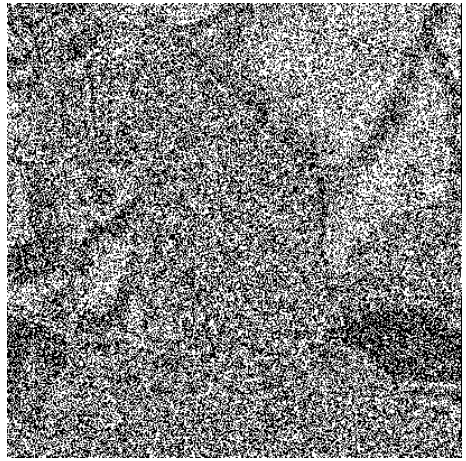
g_7: Bildebene 4

Run-Length Coding (5)

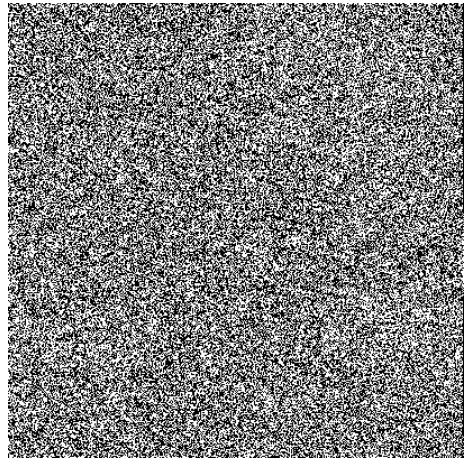
Beispiel: Bitebenen Binärcode vs. Gray Code



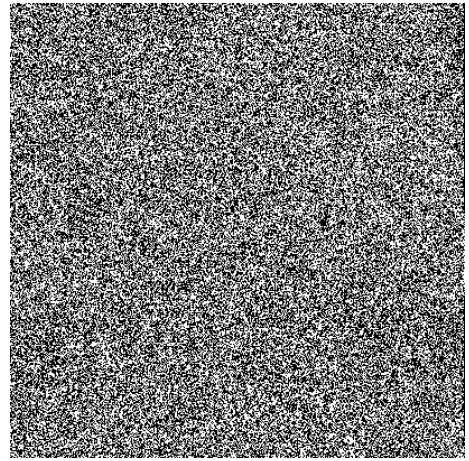
b_3: Bildebene 3



b_2: Bildebene 2

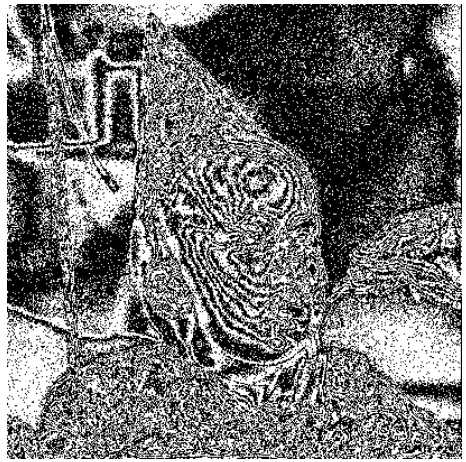


b_1: Bildebene 1

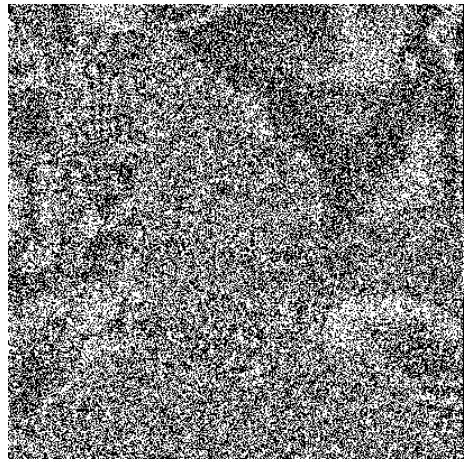


b_0: Bildebene 0

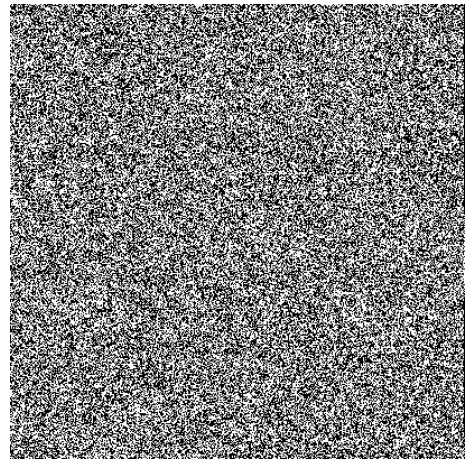
g_3: Bildebene 3



g_2: Bildebene 2



g_1: Bildebene 1



g_0: Bildebene 0

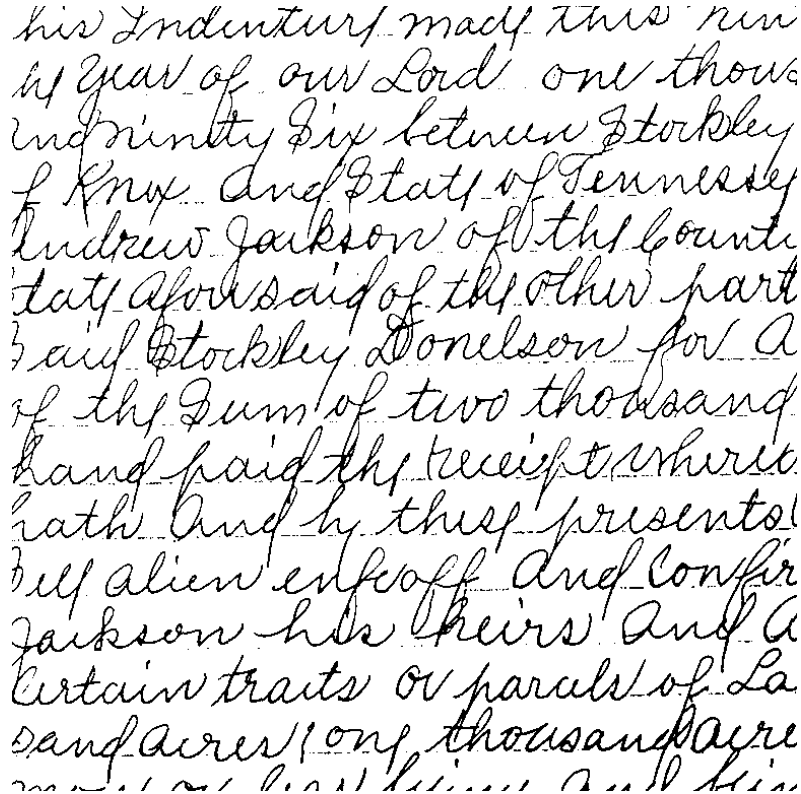
Run-Length Coding (6)

Generell weisen die Bitebenen mit Gray Code weit weniger Übergänge auf, was bei Run-Length Coding sehr vorteilhaft ist

Beispiel: Kompression mit Run-Length Coding

Binärcode: $C_r = 1.5$

Gray Code: $C_r = 1.9$



$C_r = 3.1$

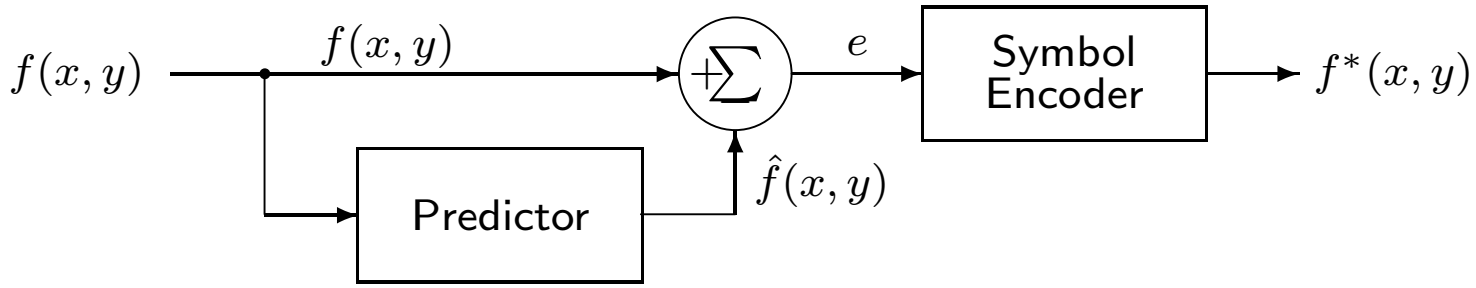
Predictive Coding (1)

Weitere Technik zur Reduktion der Interpixel-Redundanz (nur neue Information in jedem Bildpunkt kodiert). Neue Information als die Differenz:

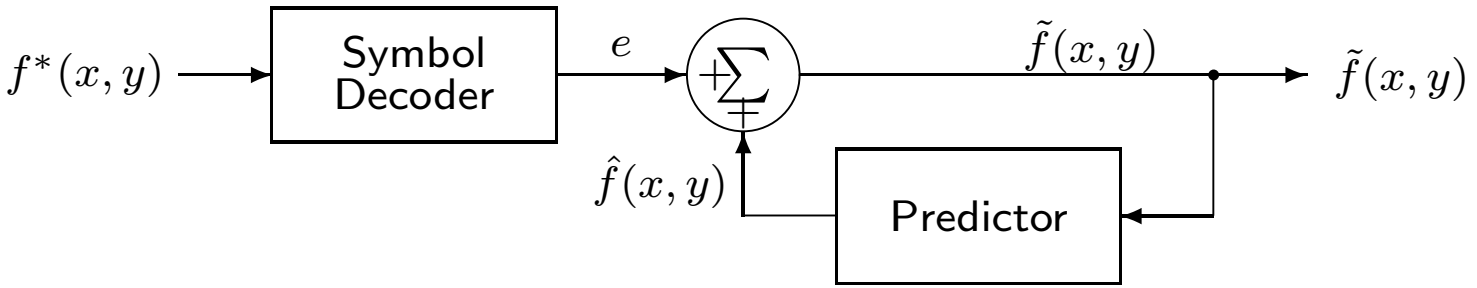
$$e \;=\; f(x,y) - \hat{f}(x,y)$$

zwischen dem betrachteten Grauwert $f(x,y)$ und einer Schätzung $\hat{f}(x,y)$ aus seinen Nachbarn (Größe e typischerweise effektiver zu kodieren als $f(x,y)$)

Source Encoder:



Source Decoder:



Predictive Coding (2)

Schätzer $\hat{f}(x, y)$:

C	B
A	$f(x, y)$

Schätzer
A
B
C
A+B-C
A+(B-C)/2
B+(A-C)/2
(A+B)/2

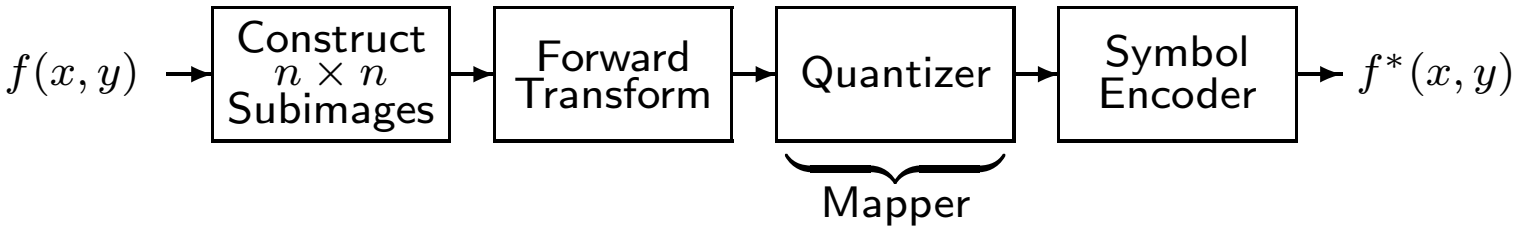
Das obige Schema stellt eine (bis auf Rundungsfehler) verlustfreie Kompression dar. Mit einem Quantizer (auf Größe e) ergibt sich bei Bedarf auch eine verlustbehaftete Kompression. Ebenso kann ein Mapper eingesetzt werden.

Transform Coding (1)

Bild $f(x, y)$ mit Hilfe einer umkehrbaren Transformation in $g(x, y)$ bringen.
Geignete Quantisierung/Kodierung von $g(x, y) \Longrightarrow$ Kompression von $f(x, y)$

Kernpunkt: Wahl der Transformation, so dass möglichst viel Information in $f(x, y)$ durch eine kleine Anzahl von Elementen in $g(x, y)$ ausgedrückt wird (Informationsverdichtung)

Source Encoder:



Source Decoder:

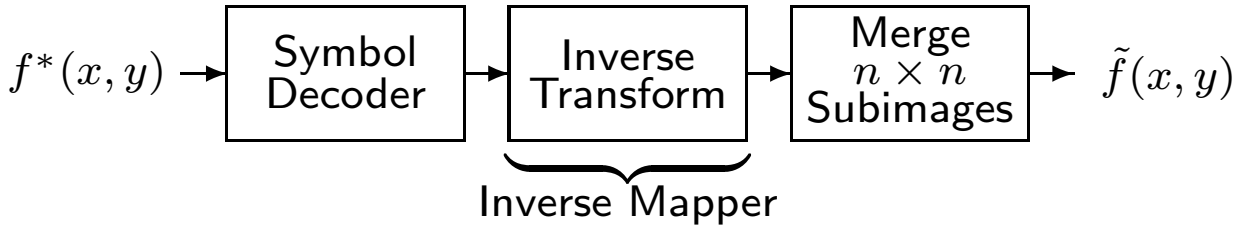


Bild in Regionen der Größe $n \times n$ aufteilen; Transformation für jede Region separat. Dekomprimierung ebenfalls zunächst auf Basis der Regionen; diese werden anschließend zu einem Gesamtbild zusammengesetzt.

Transform Coding (2)

2D diskrete Kosinus-Transformation (DCT):

Populäre Transformation im Zusammenhang mit Bildkompression (rein reelle Transformation im Gegensatz zur Fourier-Transformation)

Kosinus-Transformation: $u = 0, 1, \ldots, M-1;\ v = 0, 1, \ldots, N-1$

$$C(u,v) = k_1(u)k_2(v) \sum_{r=0}^{M-1} \sum_{c=0}^{N-1} f(r,c) \cdot \cos \frac{(2r+1)\pi u}{2M} \cos \frac{(2c+1)\pi v}{2N}$$

$$k_1(u) = \begin{cases} \sqrt{\frac{1}{M}}, & \text{für } u = 0 \\ \sqrt{\frac{2}{M}}, & \text{sonst} \end{cases} \qquad k_2(v) = \begin{cases} \sqrt{\frac{1}{N}}, & \text{für } v = 0 \\ \sqrt{\frac{2}{N}}, & \text{sonst} \end{cases}$$

Inverse Kosinus-Transformation: $r = 0, 1, \ldots, M-1;\ c = 0, 1, \ldots, N-1$

$$f(r,c) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} C(u,v)k_1(u)k_2(v) \cos \frac{(2r+1)\pi u}{2M} \cos \frac{(2c+1)\pi v}{2N}$$

Analog zur Fourier-Transformation ist: $C(0,0) = \frac{1}{\sqrt{MN}} \sum_{r=0}^{M-1} \sum_{c=0}^{N-1} f(r,c)$ proportional zum Durchschnittsgrauwert von $f(r,c)$. Koeffizienten $C(u,v)$ zeigen mit zunehmendem u und v stark abfallende Tendenz.

Transform Coding (3)

Beispiel: DCT

$f(x,y)$:

00	10	20	30	30	20	10	00
10	20	30	40	40	30	20	10
20	30	40	50	50	40	30	20
30	40	50	60	60	50	40	30
30	40	50	60	60	50	40	30
20	30	40	50	50	40	30	20
10	20	30	40	40	30	20	10
00	10	20	30	30	20	10	00

$C(u,v)$:

239	1.19	-89.76	-0.28	1.00	-1.39	-5.03	-0.79
1.18	-1.39	0.64	0.32	-1.18	1.63	-1.54	0.92
-89.76	0.64	-0.29	-0.15	0.54	-0.75	0.71	-0.43
-0.28	0.32	-0.15	-0.08	0.28	-0.38	0.36	-0.22
1.00	-1.18	0.54	0.28	-1.00	1.39	-1.31	0.79
-1.39	1.63	-0.75	-0.38	1.39	-1.92	1.81	-1.09
-5.03	-1.54	0.71	0.36	-1.31	1.81	-1.71	1.03
-0.79	0.92	-0.43	-0.22	0.79	-1.09	1.03	-0.62

Transform Coding (4)

Beispiel: DCT (Fort.)

$C(u,v)$ nach Quantisierung:

239	0	-90	0	0	0	0	0
0	0	0	0	0	0	0	0
-90	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
0	0	0	0	0	0	0	0

$\tilde{f}(x,y)$ (nach IDCT):

0.65	9.23	21.36	29.91	29.84	21.17	8.94	0.30
9.26	17.85	29.97	38.52	38.45	29.78	17.55	8.91
21.44	30.02	42.15	50.70	50.63	41.95	29.73	21.09
30.05	38.63	50.76	59.31	59.24	50.56	38.34	29.70
30.05	38.63	50.76	59.31	59.24	50.56	38.34	29.70
21.44	30.02	42.15	50.70	50.63	41.95	29.73	21.09
9.26	17.85	29.97	38.52	38.45	29.78	17.55	8.91
0.65	9.23	21.36	29.91	29.84	21.17	8.94	0.30

Transform Coding (5)

2D diskrete Sinus-Transformation (DST):

Rein reelle Transformation (analog Kosinus-Transformation)

Sinus-Transformation: $u = 0, 1, \ldots, M-1;\ v = 0, 1, \ldots, N-1$

$$S(u,v) = \frac{2}{\sqrt{(M+1)(N+1)}} \sum_{r=0}^{M-1} \sum_{c=0}^{N-1} f(r,c) \cdot \sin \frac{(r+1)(u+1)\pi}{M+1} \sin \frac{(c+1)(v+1)\pi}{N+1}$$

Inverse Sinus-Transformation: $r = 0, 1, \ldots, M-1;\ c = 0, 1, \ldots, N-1$

$$f(r,c) = \frac{2}{\sqrt{(M+1)(N+1)}} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} S(u,v) \cdot \sin \frac{(r+1)(u+1)\pi}{M+1} \sin \frac{(c+1)(v+1)\pi}{N+1}$$

Transform Coding (6)

Beispiel: DCT vs. DST

f:	11, 22, 33, 44, 55, 66, 77, 88
DCT:	140, -71, 0, -7, 0, -2, 0, 0
DST:	0, 126.9, -57.5, 44.5, -31.1, 29.8, -23.8, 25.2

Bezüglich der Informationsverdichtung ist DCT weit besser als DST

Transform Coding (6)

Beispiel: DCT vs. DST

f:	11, 22, 33, 44, 55, 66, 77, 88
DCT:	140, -71, 0, -7, 0, -2, 0, 0
DST:	0, 126.9, -57.5, 44.5, -31.1, 29.8, -23.8, 25.2

Bezüglich der Informationsverdichtung ist DCT weit besser als DST

Beispiel: DCT vs. FT

f:	20, 25, 40, 55, 70, 85, 100, 115
Rekonstruktion von DCT mit 4 Koeffizienten:	11, 23, 41, 56, 69, 84, 102, 114
Rekonstruktion von FT mit 4 Koeffizienten:	49, 41, 56, 57, 71, 70, 85, 90

Bezüglich der Informationsverdichtung ist DCT weit besser als FT

Umfangreiche Untersuchungen zeigen die Überlegenheit von DCT zur Verdichtung von Bildinformationen gegenüber DST/FT und anderen Bildtransformationen

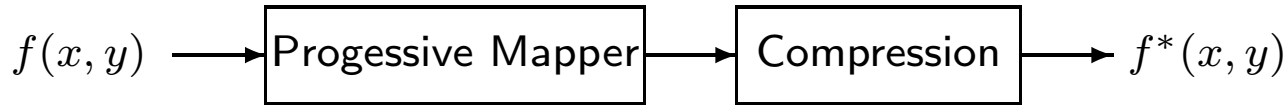
Progressive Kompression: Pyramid Coding (1)

Viele Kompressionsverfahren sind sequenzieller Natur, d.h. nur eine vollständige Dekomprimierung erlaubt die Anzeige des gesamten Bildes. Diese Eigenschaft ist vor allem in Web-Anwendungen unerwünscht. Im Gegensatz dazu erhält man bei einer progressiven Komprimierung eine Reihe von Gesamtbildern zunehmender Qualität (letztes Bild der Reihe = Eingabebild).

Progressive Kompression: Pyramid Coding (2)

Grundprinzip:

Sender:



Empfänger:

Sender (Mapper): Das ursprüngliche Bild wird in eine Repräsentation mit der progressiven Eigenschaft gebracht, die dann komprimiert wird

Empfängerseite (inverse Mapper): Der entsprechende inverse Mapper erlaubt ein progressives Anzeigen des Bildes. Zu dieser Klasse von Kompressionsverfahren gehört das Pyramid Coding.

Progressive Kompression: Pyramid Coding (3)

Bildpyramide: als progressive Repräsentation
Annahme: $N = M = 2^n$. $f(x, y) \equiv f_n \Longrightarrow$ eine Reihe von Bildern mit Auflösung $2^{n-1} \times 2^{n-1}$ (f_{n-1}), $2^{n-2} \times 2^{n-2}$ (f_{n-2}), ..., 4×4 (f_2), 2×2 (f_1), 1×1 (f_0). f_{k-1} aus f_k durch Reduktion eines 2×2 Blocks auf ein einziges Pixel

Blockreduktion durch Averaging:

54.125

81.25	32.5
61.25	41.5

90	72	58	33
140	23	21	18
100	70	72	18
16	59	44	32

Progressive Darstellung:
54.125, 32.5, 41.5, 61.25, 72, 23, 140, 33, 18, 21, 18, 32, 44, 70, 59, 16

Die Pyramide besitzt eine Beschreibung exakt derselben Größe 4^n wie das usprüngliche Bild!

Progressive Kompression: Pyramid Coding (4)

Blockreduktion durch Median:

32

72	21
59	32

90	72	58	33
140	23	21	18
100	70	72	18
16	59	44	32

Progressive Darstellung:
32, *2*, 72, 21, 59, *1*, 90, 23, 140, *3*, 58, 33, 18, *2*, 72, 18, 44, *2*, 100, 70, 16

Angabe der Position des Medians benötigt jeweils 2 Bits

Kompressionsstandards: Facsimile Compression (1)

The word facsimile comes from the Latin *facere* (make) and *similis* (like)

Dokumente werden als binäre Bitmaps gefaxt. Verlustfreie Kompressionsstandards von ITU-T (Unterorganisation der International Telecommunications Union; bis März 1993 unter dem Namen CCITT (the Consultatative Committee for International Telephone and Telegraph) bekannt.

- T4 (Group 3): Faxgeräte im Public Switched Telephone Network (PSTN); Run-Length Coding.
- T6 (Group 4): Faxgeräte in digitalen Netzen, z.B. ISDN; zwei-dimendionales Run-Length Coding.

Beide Verfahren erlauben Kompressionsrate von $\geq$ 10:0

Group 3 Code:

Statistik über Längen von weißen und schwarzen Run-Lengths wurde anhand von acht Trainingsdokumenten erstellt (Grundlage für Entwicklung eines Huffman Codes)

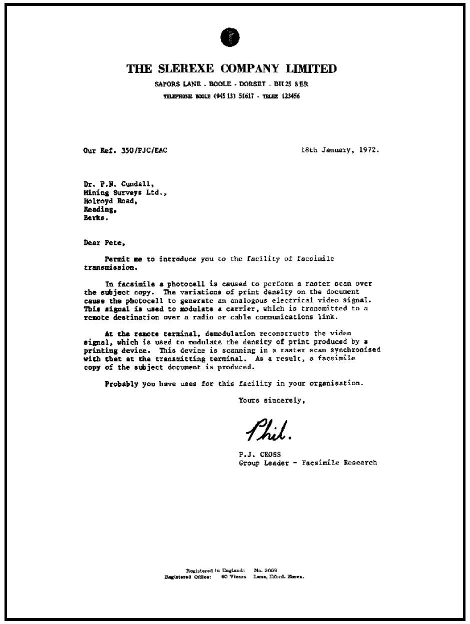
- Am häufigsten schwarze Run-Lengths von 2/3/4 und dann weiße Run-Lengths von 2–7)
- Die meisten Run-Lengths haben sehr niedrige Auftrittshäufigkeit

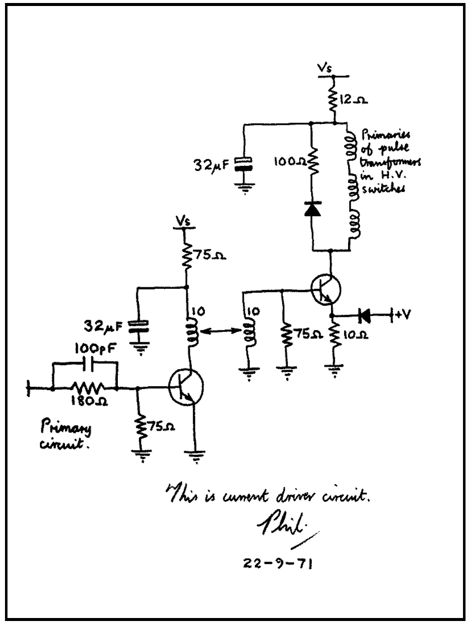
Kompressionsstandards: Facsimile Compression (2)

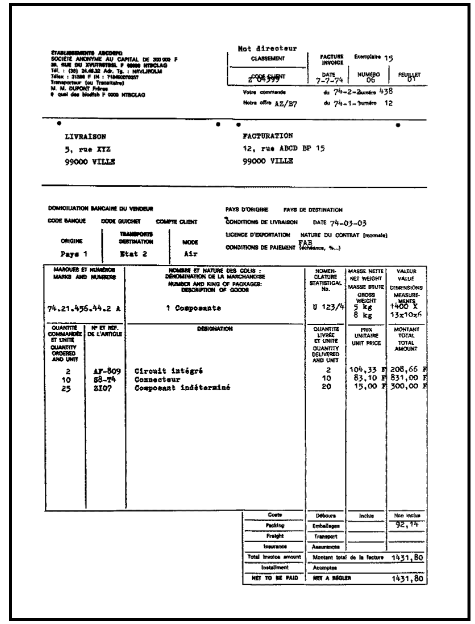
Trainingsdokumente:

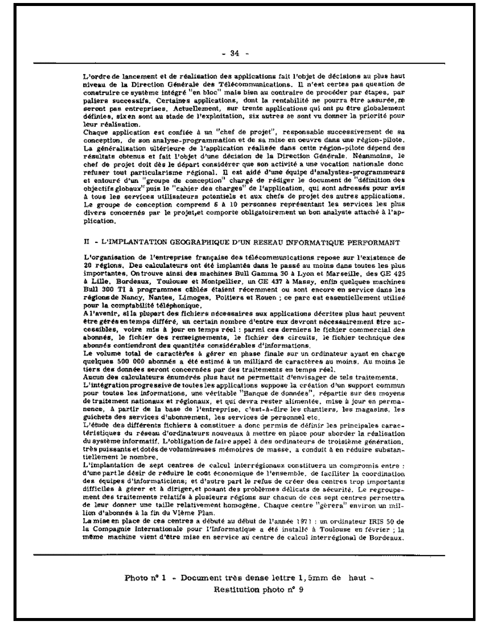
- Typed business letter (English)
- Circuit diagram (hand drawn)
- Printed and typed invoice (French)
- Densely typed report (French)
- Printed technical article including figures and equations (French)
- Graph with printed captions (French)
- Dense document (Kanji)
- Handwritten memo with very large white-on-black letters (English)

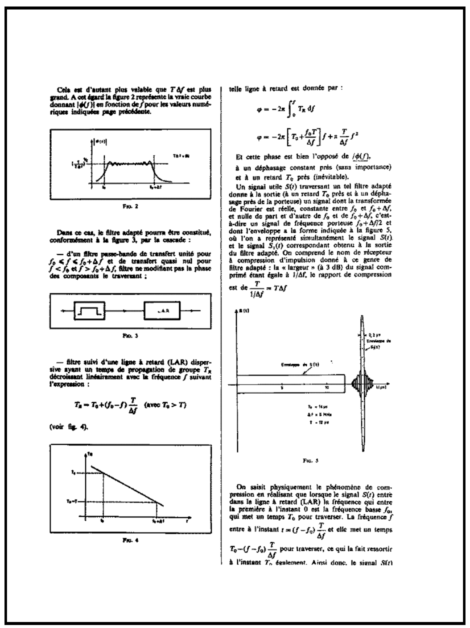
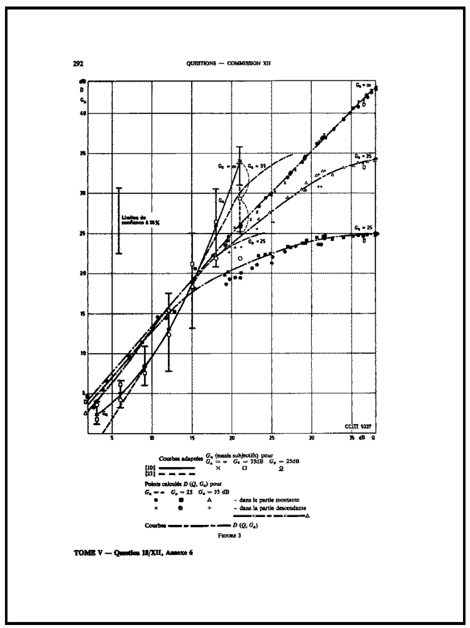
Kompressionsstandards: Facsimile Compression (3)

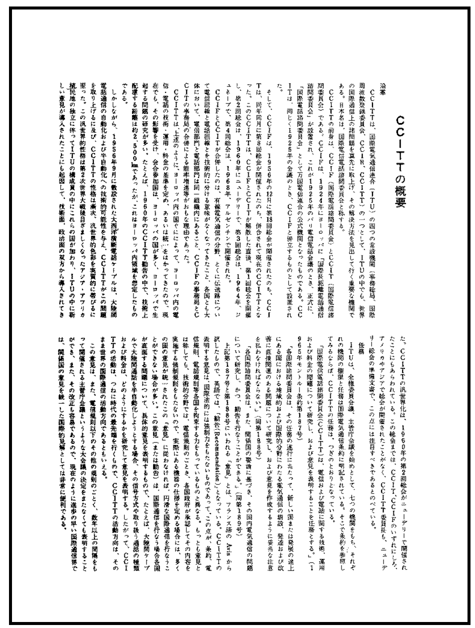


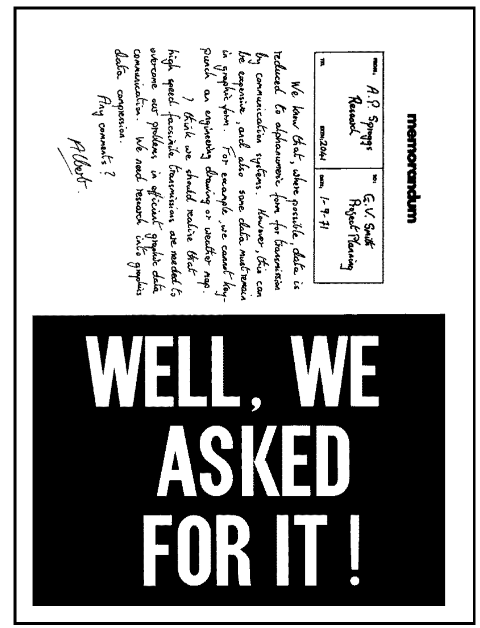










Kompressionsstandards: Facsimile Compression (4)

Group 3 Fax Code: Termination Codes

Run length	White code word	Black code word	Run length	White code word	Black code word
0	00110101	0000110111	32	00011011	000001101010
1	000111	010	33	00010010	000001101011
2	0111	11	34	00010011	000011010010
3	1000	10	35	00010100	000011010011
4	1011	011	36	00010101	000011010100
5	1100	0011	37	00010110	000011010101
6	1110	0010	38	00010111	000011010110
7	1111	00011	39	00101000	000011010111
8	10011	000101	40	00101001	000001101100
9	10100	000100	41	00101010	000001101101
10	00111	0000100	42	00101011	000011011010
11	01000	0000101	43	00101100	000011011011
12	001000	0000111	44	00101101	000001010100
13	000011	00000100	45	00000100	000001010101
14	110100	00000111	46	00000101	000001010110
15	110101	000011000	47	00001010	000001010111
16	101010	0000010111	48	00001011	000001100100
17	101011	0000011000	49	01010010	000001100101
18	0100111	0000001000	50	01010011	000001010010
19	0001100	00001100111	51	01010100	000001010011
20	0001000	00001101000	52	01010101	000000100100
21	0010111	00001101100	53	00100100	000000110111
22	0000011	00000110111	54	00100101	000000111000
23	0000100	00000101000	55	01011000	000000100111
24	0101000	00000010111	56	01011001	000000101000
25	0101011	00000011000	57	01011010	000001011000
26	0010011	000011001010	58	01011011	000001011001
27	0100100	000011001011	59	01001010	000000101011
28	0011000	000011001100	60	01001011	000000101100
29	00000010	000011001101	61	00110010	000001011010
30	00000011	000001101000	62	00110011	000001100110
31	00011010	000001101001	63	00110100	000001100111

Kompressionsstandards: Facsimile Compression (5)

Group 3 Fax Code: Makeup Codes

Run length	White code word	Black code word	Run length	White code word	Black code word
64	11011	0000001111	1344	011011010	0000001010011
128	10010	000011001000	1408	011011011	0000001010100
192	010111	000011001001	1472	010011000	0000001010101
256	0110111	000001011011	1536	010011001	0000001011010
320	00110110	000000110011	1600	010011010	0000001011011
384	00110111	000000110100	1664	011000	0000001100100
448	01100100	000000110101	1728	010011011	0000001100101
512	01100101	0000001101100	1792	00000001000	same as
576	01101000	0000001101101	1856	00000001100	white
640	01100111	0000001001010	1920	00000001101	from this
704	011001100	0000001001011	1984	000000010010	point
768	011001101	0000001001100	2048	000000010011	
832	011010010	0000001001101	2112	000000010100	
896	011010011	0000001110010	2176	000000010101	
960	011010100	0000001110011	2240	000000010110	
1024	011010101	0000001110100	2304	000000010111	
1088	011010110	0000001110101	2368	000000011100	
1152	011010111	0000001110110	2432	000000011101	
1216	011011000	0000001110111	2496	000000011110	
1280	011011001	0000001010010	2560	000000011111	

Kodierung von Run-Lengths:

- ≤ 63: Termination Code
- ≥ 64: Makeup Code + Termination Code

Kompressionsstandards: Facsimile Compression (6)

Beispiel:

12 weiß $\rightarrow$ 001000

76 weiß ($\equiv$ 64+12) $\rightarrow$ 11011 001000

140 weiß ($\equiv$ 128+12) $\rightarrow$ 10010 001000

64 weiß ($\equiv$ 64+0) $\rightarrow$ 11011 00110101

64 schwarz ($\equiv$ 64+0) $\rightarrow$ 0000001111 000110111

Kompressionsstandards: Facsimile Compression (6)

Beispiel:

12 weiß $\rightarrow$ 001000

76 weiß ($\equiv$ 64+12) $\rightarrow$ 11011 001000

140 weiß ($\equiv$ 128+12) $\rightarrow$ 10010 001000

64 weiß ($\equiv$ 64+0) $\rightarrow$ 11011 00110101

64 schwarz ($\equiv$ 64+0) $\rightarrow$ 0000001111 000110111

Beispiel: Kodierung von Zeilen

- bbbwwbbwwwwwww
  1w 3b 2w 2b 7w EOL
  000111 10 0111 11 1111 000000000001
- wwbbbbbwwwwwbb
  3w 5b 5w 2b EOL
  1000 0011 1100 11 000000000001

Kompressionsstandards: Facsimile Compression (6)

Beispiel:

12 weiß $\rightarrow$ 001000

76 weiß ($\equiv$ 64+12) $\rightarrow$ 11011 001000

140 weiß ($\equiv$ 128+12) $\rightarrow$ 10010 001000

64 weiß ($\equiv$ 64+0) $\rightarrow$ 11011 00110101

64 schwarz ($\equiv$ 64+0) $\rightarrow$ 0000001111 000110111

Beispiel: Kodierung von Zeilen

- bbbwwbbwwwwwwww
  1w 3b 2w 2b 7w EOL
  000111 10 0111 11 1111 000000000001
- wwbbbbbwwwwwbb
  3w 5b 5w 2b EOL
  1000 0011 1100 11 000000000001

Interessant: 1664 weiß $\rightarrow$ 011000 (kurz)
Typische Papierbreite von 8.2 Zoll $\Longrightarrow$ 1664 Pixeln; 1664 weiß = leeren Zeilen!

Group 3 Fax Code kennt keine Möglichkeit zur Fehlerkorrektur (einziger Übertragungsfehler $\Longrightarrow$ nicht interpretierbare Zeile)
Ein Fehler erkannt, wenn maximal 13 Bits nicht dekodiert werden können
EOL gefunden $\Longrightarrow$ Vorgang mit der nächsten Zeile fortsetzen

Kompressionsstandards: JPEG (1)

Joint Photographic Experts Group (gemeinsam von CCITT und ISO)
G.K. Wallace, The JPEG still picture compression standard, Communications of the ACM, vol. 34, no. 4, 30-44, 1991.

Varianten:

- **Verlustfreies JPEG:** Predictive Coding
  Nicht sehr verbreitet. Neu: JPEG-LS von ISO.
- **Verlustbehaftetes JPEG:** Transform Coding (DCT)
- **MJPEG:** Video-Codec, bei dem jedes Frame separat als JPEG-Bild komprimiert wird
- **JPEG2000:** basiert auf Waverlet Transformation (siehe Kapitel "Wavelets")

Kompressionsstandards: JPEG (2)

JPEG Baseline compression scheme:

- Zerlegung eines Bildes in 8×8 Blöcke
- Verschiebung des Wertebereichs ins Intervall $[-128, 127]$
- Durchführung von DCT an jedem Block
- Quantizer: Die DCT-Koeffizienten werden durch Division mit entsprechenden Werten einer 8×8 Quantisierungsmatrix quantisiert. Dieser Schritt führt zu zahlreichen Nullen, welche die Grundlage der Kompression bilden.
- Die Blöcke werden durch eine zigzag Traversierung in Vektoren überführt
- Die Koeffizienten C(0,0) werden mit Run Length kodiert; die anderen Werte mit Huffman Code

Kompressionsstandards: JPEG (3)

Default 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

Z.B. $C(0,0) = 415 \;\rightarrow\; \text{round}\left[\frac{415}{16}\right] = 26$

$C(0,1) = 29 \;\rightarrow\; \text{round}\left[\frac{29}{11}\right] = 3$

Kompressionsstandards: JPEG (4)

Beispiel:

- Block

$$b = \begin{bmatrix} 87 & 95 & 92 & 73 & 59 & 57 & 57 & 55 \\ 74 & 71 & 68 & 59 & 54 & 54 & 51 & 57 \\ 64 & 58 & 57 & 55 & 58 & 65 & 66 & 65 \\ 57 & 63 & 68 & 66 & 74 & 89 & 98 & 104 \\ 95 & 105 & 117 & 114 & 119 & 134 & 145 & 140 \\ 128 & 139 & 146 & 139 & 140 & 148 & 151 & 143 \\ 137 & 135 & 125 & 118 & 137 & 156 & 154 & 132 \\ 122 & 119 & 113 & 110 & 128 & 144 & 140 & 142 \end{bmatrix}$$

- Verschiebung des Wertebereichs ins Intervall $[-128, 127]$

$$b* = b - 128$$

Kompressionsstandards: JPEG (5)

Beispiel: (Fort.)

- DCT von b^*:

$$\begin{bmatrix} -225.4 & -30.8 & 17.4 & 5.7 & -22.4 & -1.9 & 3.8 & 1.7 \\ -241.5 & 52.1 & 0.9 & -21.2 & 8.1 & 1.9 & 0.9 & -1.3 \\ -2.5 & 50.9 & 5.1 & 9.2 & 1.6 & -3.8 & 1.6 & -0.6 \\ 102.6 & 23.4 & -11.6 & -12.8 & -10.7 & 2.8 & -3.7 & 1.2 \\ -2.4 & -20.7 & 3.5 & -10.3 & -1.4 & -2.5 & 0.3 & -0.7 \\ -12.8 & 1.6 & 2.8 & 8.1 & -5.3 & 1.1 & -1.7 & 1.1 \\ 6.6 & 7.9 & -4.9 & -7.0 & 2.2 & 0.9 & 0.7 & -0.2 \\ 10.7 & 0.4 & -0.1 & 8.0 & -4.0 & 2.4 & -2.4 & 0.6 \end{bmatrix}$$

- Quantisierung:

$$\begin{bmatrix} -14 & -3 & 20 & -1 & 0 & 0 & 0 & 0 \\ -20 & 4 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 4 & 0 & 0 & 0 & 0 & 0 & 0 \\ 7 & 1 & -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

Kompressionsstandards: JPEG (6)

Beispiel: (Fort.)

- Überführung in Vektor:

$$-14, -3, -20, 0, 4, 2, 0, 0, 4, 7, 0, 1, 0, -1, -1, 0, 0, 0, -1, 0, -1, \text{EOB}$$

wobei EOB das Blockende symbolisert. Der urspüngliche Bloock mit 64 Werten wird auf einem Vektor mit 21 Werten reduziert.

Kompressionsstandards: JPEG (7)

Beispiel:

1894 Bytes

4549 Bytes

10371 Bytes

Die Kompression nach Blockzerlegung ist effizient und ermöglicht hohe Kompressionsraten. Allerdings sind die Blockgrenzen willkürlich gezogen und koinzidieren durchaus nicht immer mit Objektgrenzen. Außerdem werden Frequenzen, die größer als die Blockgröße sind, nicht erkannt. Bei hohen Kompressionsraten treten so Blockartefakte auf, weil benachbarte Blöcke unabhängig voneinander komprimiert werden.