

1 Zahlensysteme

Umrechnen von Dezimalzahlen in andere Zahlensysteme

Die Dezimalzahl 338 wird ins 5er-System umgewandelt:

- 338 : 5 = 67 Rest 3
- 67 : 5 = 13 Rest 2
- 13 : 5 = 2 Rest 3
- 2 : 5 = 0 Rest 2
- Rückwärts gelesen: 2323

Umrechnen von anderen Zahlensystemen in Dezimalzahlen

Die Zahl 20022 (3er-System) wird ins Dezimalsystem umgewandelt:

- $2 \cdot 3^0 = 2$
- $2 \cdot 3^1 = 6$
- $0 \cdot 3^2 = 0$
- $0 \cdot 3^3 = 0$
- $2 \cdot 3^4 = 162$
- $2 + 6 + 0 + 0 + 162 = 170$

1.1 Negative Zahlen

1.1.1 Einerkomplement

- 1. Die Zahl -6 wird ins Dualsystem umgewandelt: 6 = 0110
- 2. Das Einerkomplement wird gebildet, indem alle Bits invertiert werden: 1001
- 3. Das Ergebnis ist -6 im Einerkomplement: 1001

1.1.2 Zweierkomplement

Wertebereich z.B. 8 Bit: +127 bis -128, Asymetrie aufgrund der 0.

- 1. Subtraktion ist auch eine Addition mit einer negativen Zahl $2 - 6 = 2 + (-6) = -4$
- 2. Die Addition $2 + (-6)$ aufschreiben
- 3. Zahlen aus dem Dezimal- ins Dualsystem umschreiben. $2 = 0010; 6 = 0110$
- 4. Da wir mit einer negativen Zahl rechnen -6, müssen wir das Komplement (1001) bilden und mit 1 (0001) addieren, damit wir das sogenannte Zweierkomplement erhalten.
- 5. Addition vom Komplement und 1:

1001

+0001

1010

- 6. Addition mit der 2 und -6 $2 + (-6)$:

0010

1010

























1100

= -4

Kurzgesagt: Um ein Zweierkomplement zu bilden muss man invertieren und mit1 (0001) addieren.

2 Digitaltechnik

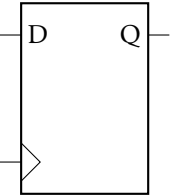
2.1 Operatoren

Function	Boolean Algebra ⁽¹⁾	IEC 60617-12 since 1997	US ANSI 91 1984	DIN 40700 until 1976
AND	A & B			
OR	A # B			
Buffer	A			
XOR	A \$ B			
NOT	!A			
NAND	!(A & B)			
NOR	!(A # B)			
XNOR	!(A \$ B)			

2.2 Flip-Flops

Flip-Flops sind Speicherelemente, die den Zustand speichern und bei einem Taktimpuls den Zustand ändern. Ein normaler "D-Flip-Flop"kann genau ein Bit Information speichern.

- Clock *C*
- Data *D*
- Ausgang *Q*



3 Informationstheorie

3.1 Typen von Datenquellen

3.1.1 Discrete Memoryless Source (DMS)

- Discrete heisst, dass die Quelle (zeitlich) einzelne Ereignisse liefert.
- Memoryless bedeutet, die Quelle erinnert sich beim Produzieren eines Ereignisses nicht an die Vorgeschichte. → Die Ereignisse sind (statistisch) unabhängig voneinander

3.1.2 Binary Memoryless Source (BMS)

- Bei dieser Quelle handelt es sich um eine DMS, die aber nur zwei verschiedene Ereignisse erzeugt.
- Ausgabe ist eine Folge von 0 und 1

3.2 Zweier-Logarithmus

$$x = \log_2(K) = \frac{\log_{10}(K)}{\log_{10}(2)}$$

3.3 Gleiche Wahrscheinlichkeit $P(x_n)$

- Je mehr Fälle es gibt, desto seltener tritt ein bestimmtes Ereignis ein.
- Je seltener ein Ereignis ist, desto höher ist sein Informationsgehalt.
- *N* sei wieder die Anzahl der möglichen Ereignisse. Wenn alle Ereigniswerte *x_n* die Gleiche Auftretungswahrscheinlichkeit *P(x_n)* haben, gilt:

$$P(x_n) = \frac{1}{N} \rightarrow N = \frac{1}{P(x_n)}$$

3.4 Informationsgehalt von Ereignissen $I(x_n)$

- Je seltener ein Ereignis eintritt, desto grösser ist der Informationsgehalt (Überraschungseffekt)
- Die folgende Formel gilt allgemein:

$$I(x_n) = \log_2(\frac{1}{P(x_n)})$$

3.5 Entropie $H(X)$

Den mittleren Informationsgehalt von Quellen nennt man Entropie:

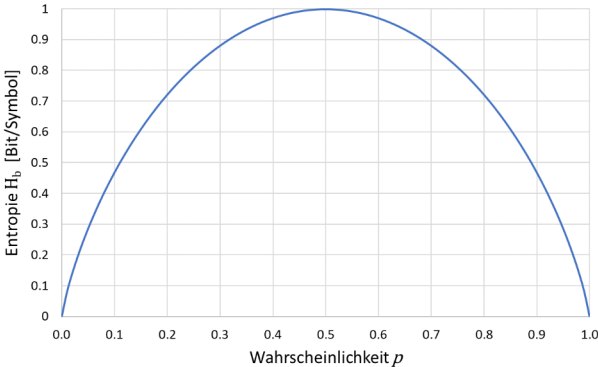
$$H(X) = \sum_{n=0}^{N-1} P(x_n) \cdot \log_2(\frac{1}{P(x_n)}) = \sum_{n=0}^{N-1} P(x_n) \cdot I(x_n)$$

Die Masseinheit der Entropie ist *Bit/Symbol*.

3.5.1 Entropie Binary Memoryless Source

Eine BMS kennt nur zwei Symbole. Ist p die Auftretungswahrscheinlichkeit des eines Symbols, folgt dass $(1-p)$ jene des anderen Symbols ist.

$$H_b = p \cdot \log_2(\frac{1}{p}) + (1 - p) \cdot \log_2(\frac{1}{1-p})$$



4 Quellencodierung

4.1 Redundanz

4.1.1 Codewortlänge ℓ_n

Symbol	Code	Codewortlänge
x_0	$c_0 = (10)$	$\ell_0 = 2Bit$
x_1	$c_1 = (110)$	$\ell_1 = 3Bit$
x_2	$c_2 = (1110)$	$\ell_2 = 4Bit$

4.1.2 Mittlere Länge der Codierung L

$$L = \sum_{n=0}^{N-1} P(x_n) \cdot \ell_n$$

4.1.3 Redundanz R

In Bit/Symbol:

$$R = L - H(X)$$

4.1.4 Theorem zu Quellencodierung

- Falls $R > 0$, dann kann verlustfrei komprimiert werden.
- Falls $R \leq 0$, dann kann nur verlustbehaftet komprimiert werden.

4.2 Kompressionsrate

$$R = \frac{\text{Grösse Originaldaten}}{\text{Grösse komprimierte Daten}}$$

Kompressionsrate $R \neq$ Redundanz R

4.3 Lauflängencodierung

Lauflängencodierung oder Run-Length Encoding (RLE) ist eine einfache Methode zur verlustfreien Datenkompression.

- Marker bestimmen, z.B. selten genutztes Zeichen.
- Marker und Anzahl der Wiederholungen speichern.

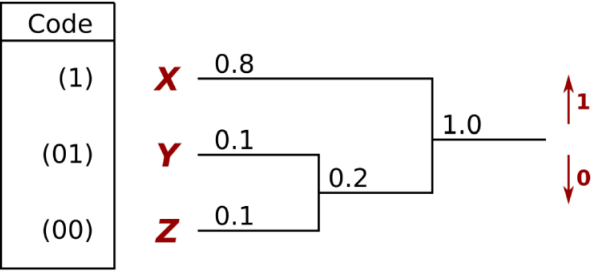
Hier verwenden wir als Marker z.B. Z:
Original: ASKEEEEEEEFEIIIIIPPPP ...
Codiert: ASKZ10EZ05IZ04P ...

4.4 Huffman-Codierung

Um Huffman-Codierung anzuwenden, muss die Wahrscheinlichkeit $P(x_n)$ der Symbole bekannt sein.

- Ordne alle Symbole nach aufsteigenden Auftretenswahrscheinlichkeiten auf einer Zeile. Dies sind die Blätter des Huffman-Baums.
- Notiere unter jedes Blatt seine Wahrscheinlichkeit.
- Schliesse die beiden Blätter mit der kleinsten Wahrscheinlichkeit an einer gemeinsamen Astgabel an und ordne dem Ast die Summe der Wahrscheinlichkeiten der beiden Blätter zu.
- Wiederhole den vorherigen Schritt mit Blättern und Ästen so lange, bis nur noch der Stamm des Baums übrig bleibt.
- Nun wird bei jeder Astgabel dem einen Zweig eine 0 und dem anderen eine 1 zugeordnet. (Die Zuordnung ist frei wählbar, muss aber über den ganzen Baum einheitlich sein).
- Nun werden auf dem Pfad vom Stamm zu jedem Blatt die Nullen und Einsen ausgelesen und von links nach rechts nebeneinander geschrieben. Dies sind die Huffman-Codeworte.

$P(X) = 0.80 \qquad P(Y) = 0.10 \qquad P(Z) = 0.10$



4.5 LZ77

Für LZ77 ein Suchbuffer n_s und Vorschabuffer n_v definiert. Der Token hat das Format (Offset, Länge, Zeichen).

Suchbuffer (n_s)	Vorschabuffer (n_v)	Restdaten	Token
...
...
...
...
...

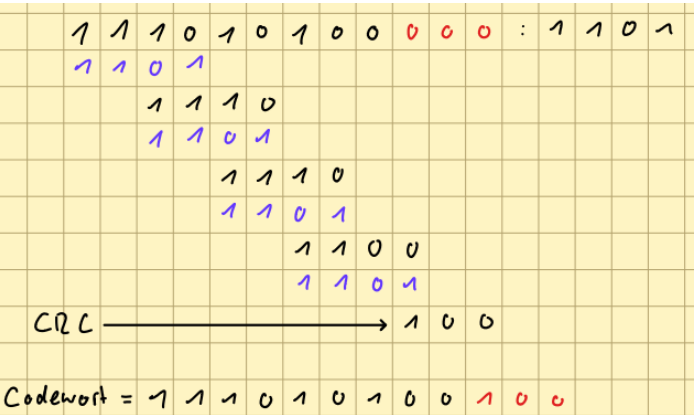
4.6 LZW

5 Kanalcodierung

5.1 CRC

5.1.1 CRC-Polynomdivision

- Nutzdatenwort: 111010100
- Generatorpolynom: $x^3 + x^2 + 1 \Rightarrow 1101$
- Nutzdatenwort mit Nullen (Grad des Generatorpolynoms [3]) erweitern: 111010100000
- Rest bestimmen durch Division: 100
- Rest an Nutzdatenwort anhängen: 111010100100



6 JPEG

- Y Cr Cb Conversion
- 8x8 Pixel Blocks
- Discrete Cosine Transformation (DCT)
- Quantization
- Zig-Zag Scanning
- DC and AC Separation

7. Run-Length Encoding
8. Huffman Encoding
9. JFIF File Creation

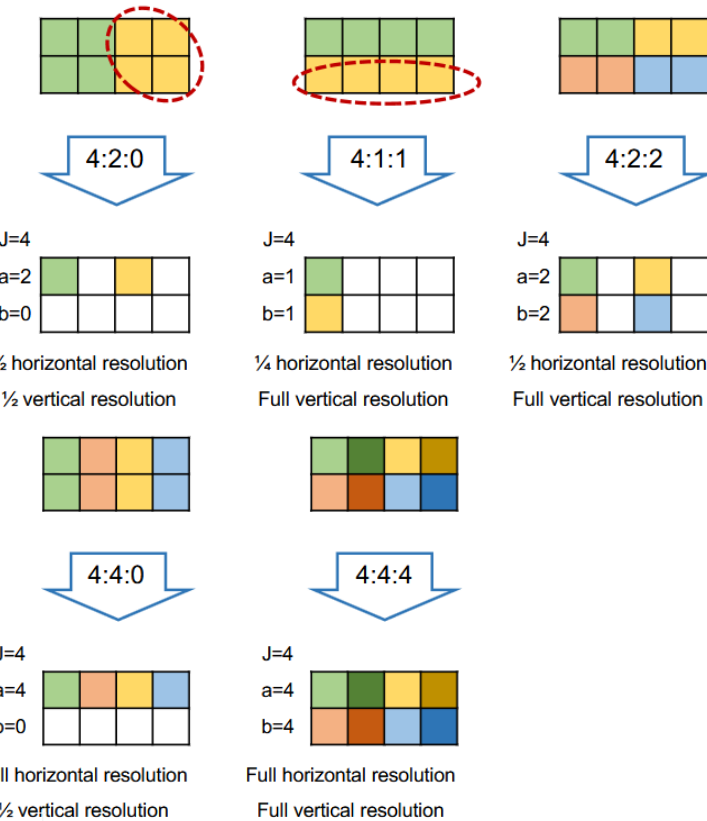
6.1 y C_r C_b Conversion

Das Bild wird in Lumineszenz (Y) und Chrominanz (C_r, C_b) aufgeteilt. C_b ist der Blauanteil und C_r der Rotanteil.

6.1.1 Subsampling

$$R = \frac{\text{Resultierende Pixel}}{\text{Ursprüngliche Pixel}}$$

- Subsampling meint, dass in beiden Chrominanz-Ebenen in der Horizontalen oder Vertikalen mehrere Pixel zusammengefasst werden.
- Der Schema-Indikator gibt die Art des Subsamplings an und hat die Form J:a:b (z.B. 4:2:0)
- Diese Notation basiert auf einem Referenzbildblock, der J Pixel breit und 2 Pixel hoch ist. Üblich ist J = 4.



6.2 8x8 Pixel Blocks

Das Bild wird in 8x8 Pixel Blöcke aufgeteilt. Diese Blöcke werden dann einzeln bearbeitet.

6.3 Discrete Cosine Transformation (DCT)

Jeder wert der 8x8 Matrix wird durch die DCT in einen Frequenzraum transformiert.

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

Vor DCT:

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

Nach DCT (Q_{vu}):

1260	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	-0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

6.4 Quantization

Die Frequenzanteile von der DCT werden nun durch eine Quantizationstabelle (Q_{vu}) geteilt. Die Quantizationstabelle wird je nach JPEG Verfahren anders gewählt. Quantizationstabelle für Luminanz:

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				

Nun nimmt man die 8x8 Tabelle von nach der DCT und teilt sie durch die Quantizationstabelle bzw. Folgende Funktion:

$$F_{vu} = \text{round}(F_{vu}/Q_{vu})$$

79	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