Digitale Medien 2

Entropie:
(Minimale Anzahl Bits, mit der eine Quelle codiert werden kann)

$$H(x) = \sum_{i=1}^{m} P(x_i) \cdot \log_2\left(\frac{1}{P(x_i)}\right)$$

H = Entropie [bit/symbol]
P = Wahrscheinlichkeit
x_i = das x-ite symbol

Mittlere Codewortlänge:

$$L = \sum_{i=1}^{m} P(x_i) \cdot n_i$$

L = Durchschnittliche Codewortlänge in Bit
$P(x_i)$ = Wahrscheinlichkeit für Codewort
n_i = Anzahl Bit für Codewort mit Wahrscheinlichkeit $P(x_i)$

Varianz:

Maximale Entropie:

$$H_{max} = \log_2(m)$$

H_{max} = maximal mögliche Entropie
m = Anzahl unterschiedlicher Symbole

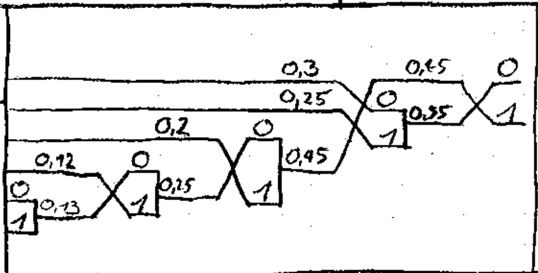
Redundanz:

$$R(x) = H_{max} - H(x)$$
$$R(x) = \log_2(m) - H(x)$$

R(x) = Redundanz
H(x) = Entropie
m = Anzahl unterschiedlicher Symbole

Huffman Codierung:

Der Huffman Codes ist ein prefix code der ein optimalen Code erzeugt.



x_i	Huffman Code
0,3	00
0,25	01
0,2	11
0,12	101
0,08	1000
0,05	1001

optimaler Huffmann Code = maximale Entropie

Logarithmen Basiswechsel:

$$\log_b r = \frac{\log_a r}{\log_a b}$$

(einige gängige Log Werte einfügen)

Shannon-Fano

$P(x_i)$					Code
0,3	>50% = 0	0			00
0,25		1			01
0,2	<50% = 1	0			10
0,12		1	0		110
0,08			1	0	1110
0,05				1	1111

1) Wahrscheinlichkeiten der Grösse nach ordnen
2) Teilen der Wahrscheinlichkeiten, so dass die Summe der beiden Teile möglichst gleich ist.

Minimum Varianz Huffmann:
⇒ Wahrscheinlichkeiten immer wieder neu ordnen nach grösse, bei gleichen Wahrscheinlichkeiten die die bereits mehr Bit hat nach oben!

Sin, Cos Werte

	0	$\pi/6$	$\pi/4$	$\pi/3$	$\pi/2$
sin	0	$1/2$	$\frac{\sqrt{2}}{2}$	$\frac{\sqrt{3}}{2}$	1
cos	1	$\frac{\sqrt{3}}{2}$	$\frac{\sqrt{2}}{2}$	$\frac{1}{2}$	0
tan	0	$\frac{1}{\sqrt{3}}$	1	$\sqrt{3}$	∞

$$\sin(2x) = 2 \cdot \sin(x)\cos(x)$$
$$\cos(2x) = \cos^2(x) - \sin^2(x) = 1 - 2\cdot\sin^2(x) = 2\cdot\cos^2(x) - 1$$
$$\cos^2\left(\frac{x}{2}\right) = \frac{1+\cos(x)}{2}$$
$$\sin^2\left(\frac{x}{2}\right) = \frac{1-\cos(x)}{2}$$

Discrete Cosine Transformation (DCT):

$$Y_{DFT}(k) = \frac{1}{N}\sum_{n=0}^{N-1} y(n)\cdot e^{-j\frac{2\pi}{N}\cdot k\cdot n} \text{ für } 0 \le k < N$$

$$X_{DCT}(0) = \frac{1}{\sqrt{N}}\sum_{n=0}^{N-1} x(n)\cdot\cos\left(\frac{(2n+1)\cdot\pi\cdot 0}{2N}\right); \text{ für } k=0$$

$$X_{DCT}(k) = \frac{\sqrt{2}}{\sqrt{N}}\sum_{n=0}^{N-1} x(n)\cdot\cos\left(\frac{(2n+1)\cdot\pi\cdot k}{2N}\right); \text{ für } 1 \le k < N$$

$$X_{DCT}(0) = \sqrt{N}\cdot e^{-j\cdot\frac{\pi\cdot 0}{2\cdot N}}\cdot Y_{DFT}(0); \text{ für } k=0$$

$$X_{DCT}(k) = \left(\sqrt{2N}\cdot e^{-j\cdot\frac{\pi\cdot k}{2N}}\right)\cdot Y_{DFT}(k); \text{ für } k=1,\dots,N-1$$

N = Anzahl Werte im Vektor
k = Laufvariable durch Ergebnisvektor
n = Laufvariable über Summe

bsp. 1D DCT
$(1\ 0) \overset{DCT}{=} (\frac{1}{\sqrt{2}}\ \frac{1}{\sqrt{2}})$
$(0\ 1) \overset{DCT}{=} (\frac{1}{\sqrt{2}}\ -\frac{1}{\sqrt{2}})$

Logarithmen

$\log_2(1) = 0$
$\log_2(2) = 1$
$\log_2(3) = 1,584$
$\log_2(4) = 2$
$\log_2(5) = 2,321$
$\log_2(6) = 2,584$
$\log_2(7) = 2,807$
$\log_2(8) = 3$
$\log_2(9) = 3,1699$

DCT von Vektoren

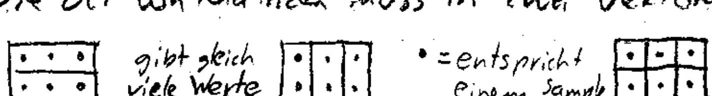
$x[n] = [x_1, x_2, x_3] \Rightarrow X_{DCT}(k) = [\ \ \ ]$ (↑ k=0, ↑ k=1, ↑ k=2)

$$X_{DCT}(0) = \frac{1}{\sqrt{3}}\left[x_1\cdot\cos\left(\frac{(2\cdot0+1)\cdot\pi\cdot0}{2\cdot3}\right) + x_2\cdot\cos\left(\frac{(2\cdot1+1)\cdot\pi\cdot0}{2\cdot3}\right) + x_3\cdot\cos\left(\frac{(2\cdot2+1)\pi\cdot0}{2\cdot3}\right)\right]; \text{ für } k=0$$

$$X_{DCT}(k) = \frac{\sqrt{2}}{\sqrt{3}}\left[x_1\cdot\cos\left(\frac{(2\cdot0+1)\cdot\pi\cdot k}{2\cdot3}\right) + x_2\cdot\cos\left(\frac{(2\cdot1+1)\cdot\pi\cdot k}{2\cdot3}\right) + x_3\cdot\cos\left(\frac{(2\cdot2+1)\cdot\pi\cdot k}{2\cdot3}\right)\right]; \text{ für } k>0$$

DCT von Matrizen

Die DCT von Matrizen muss in zwei Vektoren DCT's zerlegt werden.



Alle Zeilen einzel DCT rechnen ⇒ Alle Spalten einzel DCT rechnen mit den Werten der Zeilen DCT ⇒ Matrix ist nun DCT transformiert.

$$\begin{bmatrix}1&0\\0&0\end{bmatrix} \overset{DCT}{=} \begin{bmatrix}1/2&1/2\\1/2&1/2\end{bmatrix}$$
$$\begin{bmatrix}0&1\\0&0\end{bmatrix} \overset{DCT}{=} \begin{bmatrix}1/2&-1/2\\1/2&-1/2\end{bmatrix}$$
$$\begin{bmatrix}0&0\\1&0\end{bmatrix} \overset{DCT}{=} \begin{bmatrix}1/2&1/2\\-1/2&-1/2\end{bmatrix}$$
$$\begin{bmatrix}0&0\\0&1\end{bmatrix} \overset{DCT}{=} \begin{bmatrix}1/2&-1/2\\-1/2&1/2\end{bmatrix}$$

bsp: 2D DCT mit Linearitätsgesetz

$$\begin{bmatrix}7&2\\1&-3\end{bmatrix} = 7\begin{bmatrix}1&0\\0&0\end{bmatrix} + 2\begin{bmatrix}0&1\\0&0\end{bmatrix} + 1\begin{bmatrix}0&0\\1&0\end{bmatrix} - 7\begin{bmatrix}0&0\\0&1\end{bmatrix} \overset{DCT}{=} 7\begin{bmatrix}1/2&1/2\\1/2&1/2\end{bmatrix} + 2\begin{bmatrix}1/2&-1/2\\1/2&-1/2\end{bmatrix} + 1\begin{bmatrix}1/2&1/2\\-1/2&-1/2\end{bmatrix} - 7\begin{bmatrix}1/2&-1/2\\-1/2&1/2\end{bmatrix} = \begin{bmatrix}3/2&9/2\\11/2&1/2\end{bmatrix}$$

Arithmetic Coding

Beim Arithmetic Coding werden nicht mehr einzelne Symbole codiert, sondern ganze Sequenzen.
z.B. Sequenz: acbaab
$P(a) = 1/2 \Rightarrow P(acbaab) = (\frac{1}{2})^3\cdot(\frac{1}{4})^2\cdot(\frac{1}{4})$
$P(b) = 1/4 \quad P(acbaab) = \frac{1}{512}$
$P(c) = 1/4 \Rightarrow$ Eigeninformation L_{min} der Sequenz:

L_{min} = minimal benötigte Bits zur Codierung

$$L_{min} = \log_2\left(\frac{1}{P(acbaab)}\right)$$

z.B. Wenn die Wahrscheinlichkeit der Symbole laufend angepasst werden

		a	c	b	a	a	b
P(a)	1/3	2/4	2/5	2/6	3/7	4/8	4/9
P(b)	1/3	1/4	1/5	2/6	2/7	2/8	3/9
P(c)	1/3	1/4	2/5	2/6	2/7	2/8	2/9

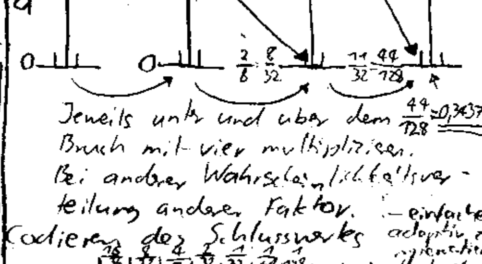
↑ Anfangswahrscheinlichkeiten

$P(ac\ldots b) = \frac{1}{3}\cdot\frac{1}{4}\cdot\frac{1}{5}\cdot\frac{2}{6}\cdot\frac{3}{7}\cdot\frac{3}{8}$

$L_{min} = \log_2\left(\frac{1}{P(ac\ldots b)}\right)$

$P(a) = 1/2$
$P(b) = 1/4$
$P(c) = 1/4$

Sequenz abc Codieren!



Jeweils unter und über dem Bruch mit vier multiplizieren. $\frac{44}{128} = 0,34375$
Bei anderer Wahrscheinlichkeitsverteilung anderer Faktor.

Codieren des Schlusswertes

	2^0	2^{-1}	2^{-2}	2^{-3}	2^{-4}	2^{-5}	2^{-6}	2^{-7}	2^{-8}
11/32	0	0	1	0	1	1	0	0	0
<12/32	0	0	1	0	1	1	1	1	1

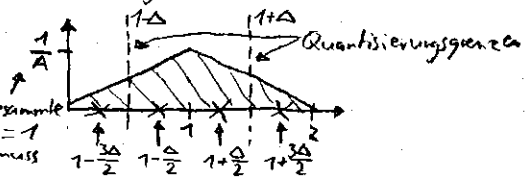
– einfacher adaptiv anpassbar als Huffman
– gleich wie Entropie oder zumindest näher als Huffman bei Entropie

Skalare Quantisierung

$$\sigma_q^2 = \sum_{i=1}^{M} \int_{b_{i-1}}^{b_i} (x - y_i)^2 \cdot f_x(x) \cdot dx$$

σ_q^2 = mittlere quadratische Quantisierungsfehler
M = Anzahl Quantisierungsschritte
b_i = Quantisierungsgrenzen [----] M+1 (Entscheidungsgrenzen)
y_i = Rekonstruktionspunkt [x] M Rekonstruktionswerte
$f_x(x)$ = Verteilungsfunktion

bsp:



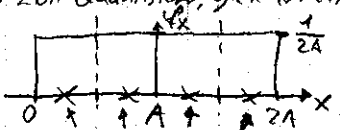
$$\sigma_q^2 = 2\left[\left(\int_0^{1-\Delta} (x-(1-\tfrac{3\Delta}{2}))^2 \cdot x\,dx\right) + \left(\int_{1-\Delta}^{1} (x-(1-\tfrac{\Delta}{2}))^2 \cdot x\,dx\right)\right]$$

für minimales Δ (kleinste Fehler) am schärfst nach langer Rechnung, ableiten

$$\frac{d\sigma_q^2}{d\Delta} = 0 \Rightarrow \Delta \text{ mit dieser Bedingung sind optimal (kleinster Fehler garantiert)}$$

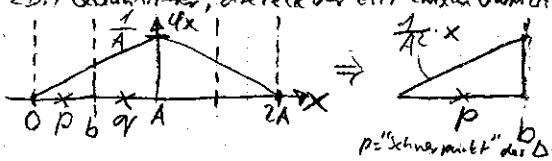
bsp: suchen von optimalen skalaren Quantisierern

① 2 Bit Quantisierer, gleichverteilt zwischen 0 und 2A



Rekonstruktionspunkte

② 2 Bit Quantisierer, dreieckverteilt zwischen 0 und 2A



b = Quantisierungsgrenze die noch nicht ganz fix ist! abhängig von p und q

$$\gamma_{\frac{M}{2}} = \frac{\int_{b_{\frac{M}{2}-1}}^{b_{\frac{M}{2}}} x \cdot f_x(x)\,dx}{\underbrace{\int_{b_{\frac{M}{2}-1}}^{b_{\frac{M}{2}}} f_x(x)\,dx}_{\text{Fläche}}} \Rightarrow p = \frac{\int_0^b \left(\frac{x}{A^2}\right) \cdot x \cdot dx}{\left[\int_0^b \left(\frac{x}{A^2}\right) \cdot dx\right]}$$

$$q = \frac{\int_b^A \left(\frac{x}{A^2}\right) \cdot x \cdot dx}{\left[\int_b^A \left(\frac{x}{A^2}\right) \cdot dx\right]} = \frac{2}{3}\cdot\frac{A^3 - b^3}{A^2 - b^2} \qquad p = \frac{\frac{b^3}{3A^2}}{\frac{b^2}{2A^2}} = \frac{2}{3}b$$

$$b_j = \frac{\gamma_{j+1} + \gamma_j}{2}$$

$\Rightarrow b = (p+q)/2 \Rightarrow 0 = b^3 - 2A^2 b + A^2$

$\Rightarrow b = 0{,}618A \;;\; p = 0{,}4170 \cdot A \;;\; q = 0{,}829 \cdot A$

Vektorquantisierung

Anwendung in: CELP, G.729, Ogg Vorbis
Vorteile: - es kann statistische Abhängigkeiten von Daten berücksichtigen
- die Rekonstruktionswerte können auf die Messwerte angepasst werden.

Bei der Vektorquantisierung spielen die Initialwerte für die Rekonstruktionspunkte (sogenanntes Codebook) eine sehr wichtige Rolle. Je nach dem wie das Codebook gewählt wird, entstehen nach N Iterationen eines Algorithmus (bis dieser konvergiert, kleinster Fehler) entstehen verschiedene Lösungen. Der Initialwert bestimmt demnach die schlussendliche Lösung.

LBG (Linde - Buzo - Gray) - Algorithmus

Der LBG-Algorithmus erzeugt ein gutes Codebook anhand von Trainingsdaten in einem iterativen Verfahren.

① Starten mit Initialisierungswerten $\{y_i^{(0)}\}_{i=1}^{M}$; $k = 0$; $D^{(0)} = 0$; ε definieren (Konvergenzgenauigkeit)

② Quantisierungsregionen finden:
$V_i^{(k)} = \{X : d(X, Y_i) < d(X, Y_j)\ \forall j \neq i\}$ $j = 1, 2, \ldots, M$

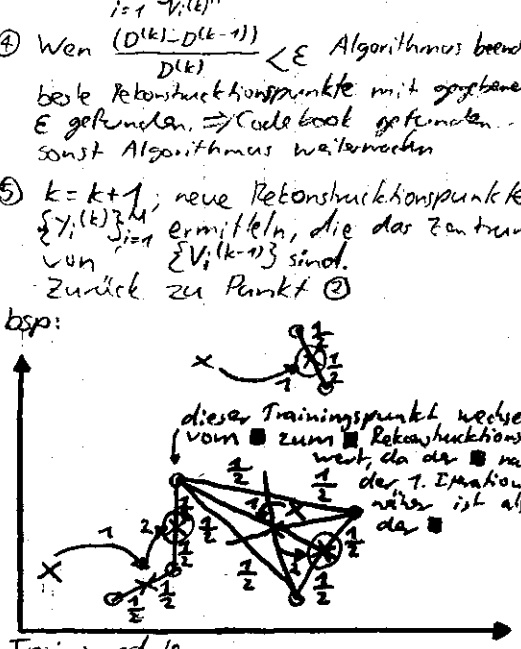
③ Rechnen der ... (distortion)

$$D^{(k)} = \sum_{i=1}^{M} \int_{V_i^{(k)}} \|X - Y_i^{(k)}\|^2 \cdot f_X(x)\,dx$$

④ Wenn $\frac{(D^{(k)} - D^{(k-1)})}{D^{(k)}} < \varepsilon$ Algorithmus beenden
beste Rekonstruktionspunkte mit gegebenem ε gefunden. ⇒ Codebook gefunden ... sonst Algorithmus weiterrechnen

⑤ $k = k+1$; neue Rekonstruktionspunkte $\{y_i^{(k)}\}_{i=1}^{M}$ ermitteln, die das Zentrum von $\{V_i^{(k-1)}\}$ sind.
Zurück zu Punkt ②

bsp:

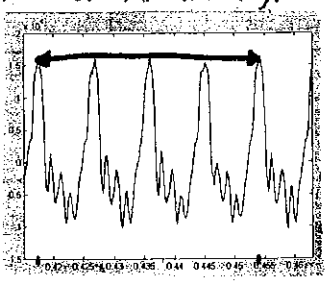


Trainingsdaten
■■■ Rekonstruktionspunkte
Endrekonstruktionswerte die kleinsten Fehler als ε aufweisen.

Spracheigenschaften ②

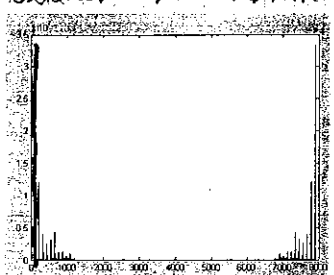
Voiced Speech

Dabei schwingen die Stimmbänder. Das heisst die Person sagt einen Vokal. z.B. "a"
bsp: Pitch Bestimmung (Abtastrate 8kHz)
Plot der Aufzeichnung:



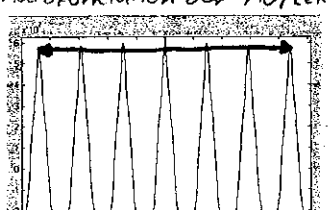
0,452 - 0,415 = 0,03
4T = 37ms ⇒ T = 9,25ms
⇒ Pitch = 1/T = 108Hz

Absolutwert der FFT der Aufzeichnung



Stärkste Spektrallinie ist Pitch
⇒ 108Hz

Autokorrelation der Aufzeichnung

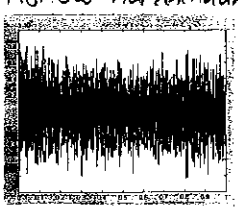


⇒ 225 + 220 = 445
⇒ 6T = 445/8000 = 0,0556
⇒ Pitch = 1/T = 108Hz

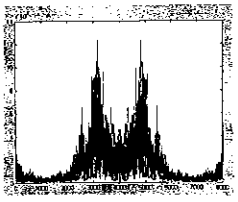
Für andere Vokale verändert sich die Pitchfrequenz.

Unvoiced Speech

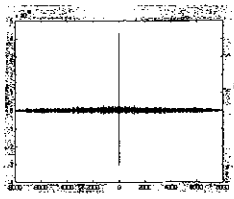
Dabei schwingen die Stimmbänder nicht. Das heisst die Person sagt einen Konsonant z.B. "sch"
Plot der Aufzeichnung



Absolutwert der FFT der Aufzeichnung



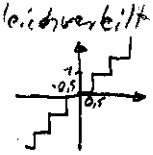
Autokorrelation der Aufzeichnung:



Der Korrelationspeak enthält den Wert für die Signalverzögerung ≈ ca. 4,3

Uniform (Gleichförmiger) Quantisierer

Alle Intervalle ausser die äussersten haben die selbe Grösse
Die Rekonstruktionswerte sind auch gleichverteilt

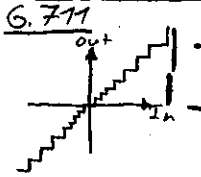


- $\sigma_q^2 = \frac{\Delta^2}{12}$; mit $q_x = x - Q(x)$
- $SNR_{dB} = n \cdot 6{,}02\,dB$ mit n Bits

Bei nonuniform Quellen, besten uniform Quantizer findet mit

$$\frac{\partial \sigma_q^2}{\partial \Delta} = 0$$ setzen und Minimum σ_q^2 finden

G.711



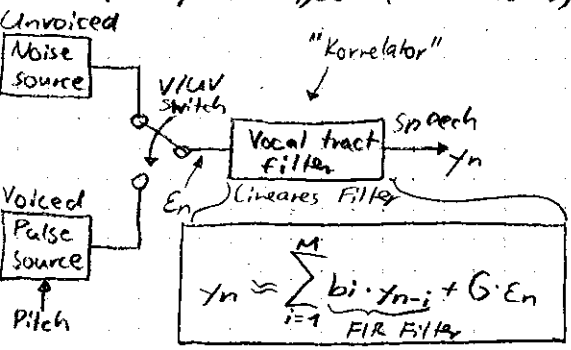
- leiser Bereich entspricht ≈ 13Bit
- lauter Bereich entspricht ≈ 5Bit

SNR ist konstant ≈ 38dB

LPC (Linear Predictive Coder)

Ein LPC Voice Coder analysiert den Pitch und die Leistung eines Sprachsignals und unterscheidet dabei zwischen Voiced und Unvoiced Speech eines Sample Blockes.
z.B. LPC-10: $f_0 = 8kHz$; Analyseblock a 180 Samples $\Rightarrow 22{,}5ms$ Sprache

Ein LPC versucht mit Hilfe eines linearen Filters, dem Pitch und der Unterscheidung zwischen Voiced und Unvoiced das Sprachsignal in Parameter zu zerlegen, die verschickt werden können.
Model für Sprachanalyse: (LPC decoder)



$$\begin{bmatrix} a & b \\ c & d \end{bmatrix}^{-1} = \frac{1}{ad-bc}\begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$$

G = Filterverstärkung
Parameter die gesendet werden:
- Pitch (in LPC-10 zwischen 2,5 und 19,5ms)
- Unvoiced / Voiced Entscheid
- Filterkoeffizienten des Vocal tract Filters die den kleinsten Fehler ergeben

Unterscheiden zwischen Voiced und Unvoiced:
- die Leistung eines Voiced Signals ist deutlich grösser als eines Unvoiced
- in LPC-10 wird AMDF (average magnitude difference function) verwendet

Ermitteln der Vocal tract Filterkoeffizienten:
- die Koeffizienten werden so bestimmt das sich der kleinste mittlere quadratische Fehler einstellt.

$$\begin{bmatrix} R_{yy}(0) & R_{yy}(1) & R_{yy}(2) & \cdots & R_{yy}(M-1) \\ R_{yy}(1) & R_{yy}(0) & R_{yy}(1) & \cdots & R_{yy}(M-2) \\ R_{yy}(2) & R_{yy}(1) & R_{yy}(0) & \cdots & R_{yy}(M-3) \\ \vdots & & & & \\ R_{yy}(M-1) & R_{yy}(M-2) & R_{yy}(M-3) & \cdots & R_{yy}(0) \end{bmatrix} \cdot \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_M \end{bmatrix} = \begin{bmatrix} R_{yy}(1) \\ R_{yy}(2) \\ R_{yy}(3) \\ R_{yy}(M) \end{bmatrix}$$

a_{1-M} = Filterkoeffizienten

Prediction in DPCM

Varianz: $\sigma_d^2 = E[(x_n - p_n)^2]$

$\sigma_d^2 = E[(x_n - \sum_{i=1}^{M} a_i \cdot x_{n-1})^2]$ MSE

$A = \begin{bmatrix} a_1 \\ a_2 \\ a_3 \\ a_M \end{bmatrix}$; $P = \begin{bmatrix} R_{xx}(1) \\ R_{xx}(2) \\ R_{xx}(3) \\ R_{xx}(M) \end{bmatrix}$

$R_{xx}(k) = E(x_n ; x_{n+k})$

$\frac{d\sigma_d^2}{da_1} = 0$; $\frac{d\sigma_d^2}{da_2} = 0$ usw $\longrightarrow R \cdot A = P \Rightarrow A = R^{-1} \cdot P$

a_n = Prädiktor Koeffizienten
M = Prädiktor Ordnung

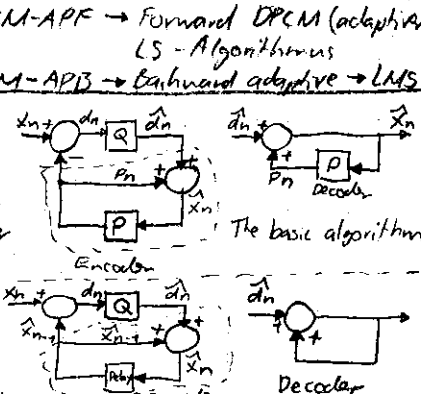
$R = \begin{bmatrix} R_{xx}(0) & R_{xx}(1) & \cdots & R_{xx}(M-1) \\ R_{xx}(1) & R_{xx}(0) & \cdots & R_{xx}(M-2) \\ R_{xx}(2) & R_{xx}(1) & \cdots & R_{xx}(M-3) \\ R_{xx}(M-1) & R_{xx}(M-2) & \cdots & R_{xx}(0) \end{bmatrix}$

$SNR_{dB} = \frac{\sum_{i=1}^{M} x_i^2}{\sum_{i=1}^{M} (x_i - \hat{x}_i)^2}$

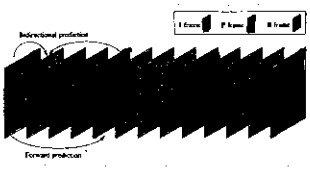
DPCM-APF → Forward DPCM (adaptiv) LS-Algorithmus
DPCM-APB → Backward adaptive → LMS

Differential Encoding

Die verlustbehaftete differentielle Quantisierung erzielt höhere Kompressionsraten als die verlustfreie. Es benötigt weniger Bits um eine Differenz zu codieren als das Original Signal / Pixel.
q_n = Quantisierungsfehler



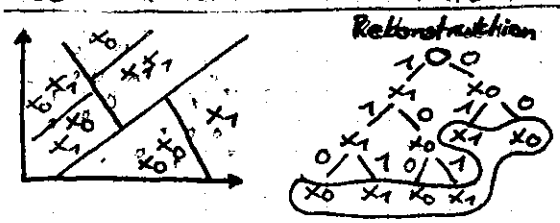
Video Compression MPEG-1 ③



- Das erste Frame eines GOP muss immer ein I-Frame sein. Ein GOP braucht somit sicher ein I-Frame, kann aber auch mehrere haben.

- Ein Videostrom wird üblicherweise in einem Groups of Pictures (GOP) zusammengefasst.
- Ein GOP besteht aus I, P und B-Frames welche unterschiedliche Eigenschaften haben.
- Das P- und I-Frame werden auch als anchor Frame bezeichnet.
- Die Kompression der Frames: höchste Komp. B-Frame
  mittlere Komp. P-Frame
  kleinste Komp. I-Frame
- Das P-Frame benutzt zur Komprimierung motion-compensated prediction vom letzten I oder P-Frame, je nach dem welches näher ist.
- Die B-Frames können nur gesetzt werden wenn vergangene und zukünftige anchor Frames vorhanden sind. B-Frames weisen damit eine hohe Kompression mit motion-compensated prediction der zukünftigen und vergangenen anchor Frames auf. Die B-Frames werden nicht zur prediktion von anderen Frames verwendet. Dadurch dürfen sie eine grössere Fehlerfortpflanzung aufweisen, da der Fehler durch prädiktion nicht vorangepflanzt wird.

Tree-Structed Vector Quantization



Blätter des Baumes sind Rekonstruktionswerte.

- Rechenaufwand kleiner als LBG, selten aber besser
- Tests minimieren ($\log_2(k)$) Vergleiche
- Speicheraufwand ist grösser
- Jedes Bit mehr ist etwas wert
- skalierbares System
- bsp: Webbild: erst verschwommen & immer schärfer
- Basis für JPEG-2000

Mittelwert, Effektivwert, Varianz

lin. Mittelwert / erstes Moment (DC):

$$\mu_S = \overline{s(t)} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} s(t) \cdot dt \quad ; \quad \mu_1 = E(x) = \int_{-\infty}^{+\infty} f_x(x) \cdot x \cdot dx$$

quadratischer Mittelwert / zweites Moment (Leistung):

$$P_S = \overline{s^2(t)} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} s^2(t) \cdot dt \quad ; \quad \mu_2 = E(x^2) = \int_{-\infty}^{+\infty} x^2 \cdot f_x(x) \cdot dx$$

Effektivwert:

$s_{eff} = \sqrt{P_S}$

Varianz (AC-Leistung):

$$V_S = \sigma^2 = \overline{(s(t) - \mu_S)^2} = \lim_{T\to\infty} \frac{1}{T} \int_{-T/2}^{T/2} (s(t) - \mu_S)^2 \cdot dt = var(x) = E(x^2) - E(x)^2$$

$$= E((x-\mu_1)^2) = \int_{-\infty}^{+\infty} (x-\mu_1)^2 \cdot f_x(x) \cdot dx$$

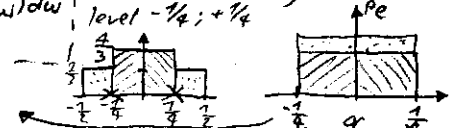
Energie:

$$W_S = \int_{-\infty}^{+\infty} s^2(t) \cdot dt = \frac{1}{\pi} \int_{-\infty}^{+\infty} E(\omega) \, d\omega$$

Varianz = AC-Leistung

$\frac{q^2}{12} = \frac{(1/2)^2}{12} = \frac{1}{48}$

bsp: Varianz von sym. Quelle → 1 Bit skalare Quantisierung mit Rekonstruktionslevel $-1/4$; $+1/4$



JPEG (Joint Photographic Experts Group)

Encoder:

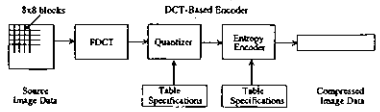


Figure 1. DCT-Based Encoder Processing Steps

Decoder:

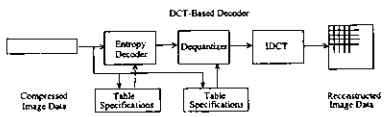
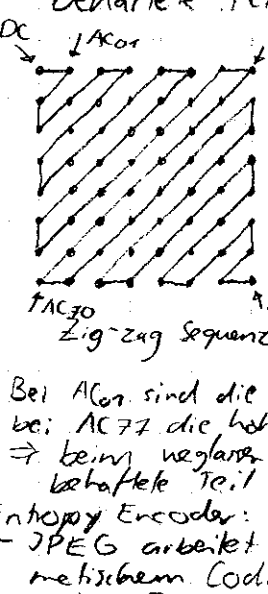


Figure 2. DCT-Based Decoder Processing Steps

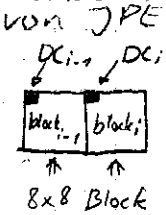
Möglichkeiten:
- kompression von 1/10 bis 1/50 der originalen Dateigrösse ohne sichtbare Qualitätseinbussen
- komprimieren von Farb- und Graustufenbildern, wobei ein farbiges Bild in mehrere Graustufenbilder zerlegt wird und einzel komprimiert.

8×8 FDCT und IDCT:
- Es werden immer Blöcke von 8×8 Pixel gemeinsam durch eine diskrete Kosinus Transformation in die Frequenzebene transformiert.
- der FDCT und IDCT Algorithmuss wird im JPEG Standard nicht vorgeschrieben.
- JPEG Bilde können 8 Bit oder 12 Bit pro Sample enthalten. Für jede Variante muss der DCT-Algorithmus ausgelegt sein.

Quantizer:
- Nach der FDCT wird jeder der 64 DCT Koeffizienten gleichförmig Quantisiert mit Hilfe von 64-Element Quantisierungstabellen.
- im Quantisierer entsteht der Verlust-behaftete Teil von JPEG



Zig-zag Sequenz der 8×8 Blöcke stark korreliert



Diff = DCi − DCi-1

DC = Helligkeit des Bildes

AC = Kontrast des Bildes (je mehr Frequenzen desto kontrastreicher)

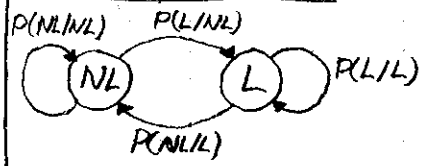
Der DC wert eines Blockes ist ein Mittelwert aller 64 Image-samples. Die DC-Werte sind mit Beeinflussungen, AC-Werten der 8×8 Blöcke stark korreliert.

Bei AC01 sind die tiefen Frequenzen des Bildes und bei AC77 die hohen, welche entbehrlich sind, (Kontrast)
⇒ beim weglassen (Quantisieren) wird der Verlust-behaftete Teil von JPEG durchgeführt.

Entropy Encoder:
- JPEG arbeitet mit Huffmanraum oder Arithmetischem Coding. Arithmetisches Coding ist dabei 5-10% effektiver, aber im Gegenzug massiv komplexer.

Video over IP ④

Paketverlustmodel:



NL = not Lost
L = Lost
P(L/NL) = Wahrscheinlichkeit für Verlust wenn vorhe kein Verlust bestand

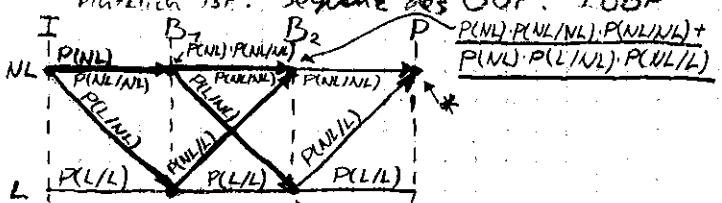
Modelbedingungen:

$$\Rightarrow P(L) + P(NL) = 1$$
$$P(NL/NL) + P(L/NL) = 1$$
$$P(L/L) + P(NL/L) = 1$$

Fehlerfortpflanzung:

Gesamme Theorieinhalt über MPEG (I, P und B-Frame auf Blatt 3 (Video Compression).

bsp: Wahrscheinlichkeit das Information in P Frame nützlich ist? Sequenz des GOP: IBBP



P(NL)·P(L/NL) P(NL)·P(NL/NL)·P(L/NL)+P(NL)·P(L/NL)·P(L/L)

*[P(NL)·P(NL/NL)·P(NL/NL)+P(NL)·P(L/NL)·P(NL/NL)]·P(NL/NL)+
[P(NL)·P(NL/NL)·P(L/NL)+P(NL)·P(L/NL)·P(L/L)]·P(NL/L)

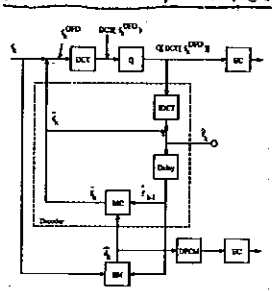
P-Frame hängt nur vom I-Frame ab, darum müssen beide wege durch die B-Frames berücksichtigt werden.

bsp: Wahrscheinlichkeit das Informationen in B-Frame nützlich sind? Gleiche voraussetzung wie oben.

$$P(B_1\ \text{nützlich}) = P(NL)\cdot P(NL/NL)^3 + P(NL)\cdot P(L/NL)\cdot P(NL/L)\cdot P(NL/NL)$$
$$P(B_2\ \text{nützlich}) = P(NL)\cdot P(NL/NL)^3 + P(NL)\cdot P(L/NL)\cdot P(NL/L)\cdot P(NL/NL)$$

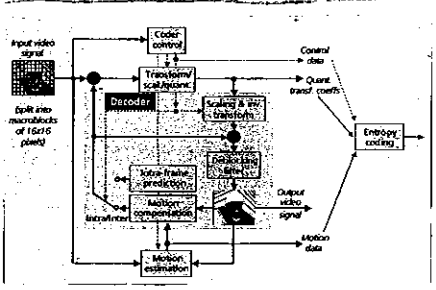
B-Frame hängt von I- und P-Frame ab, von Zukunft und Vergangenheit.

Motion compensated DCT (wie in MPEG I und II)

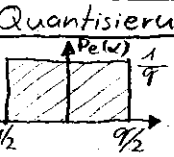


BM = Block matching
MC = Motion Compensation

Basic coding structure of H.264/AVC



Quantisierungsfehler



Formeln gelten bei guter Aussteuerung:

$$Pe = \frac{q^2}{12} \quad ; \quad de = \frac{q}{\sqrt{12}} \quad ; \quad q = \frac{App}{2^B}$$

$$Pe = \frac{App^2}{12 \cdot 2^{2B}} \quad ; \quad N = 2^B$$

$$SNR_{lin} = \frac{P_s}{Pe} = \frac{P_s \cdot 12}{App^2} \cdot 2^{2B}$$

$$SNR_{dB} = 10 \cdot \log\left(\frac{P_s}{Pe}\right) = K + 6{,}02 \cdot B$$

Gleichverteilung: K=0
Sinusförmig: K=+1,67dB
Normalverteilt: K=-8,5dB

eistungen:
inus: $P_s = \frac{A^2}{2}$
Rechteck: $P_s = \frac{A^2}{3}$

B = Wortlänge (Bit/Abtastwert)
= Quantisierungsschrittweite
p = Aussteuerbereich
N = Quantisierungsschrittanzahl
e = Quantisierungsfehlerleistung, P_s = Leistung des Signals

Bei schlechter Aussteuerung muss jeder Fall individuell angeschaut werden.