Echtzeit–Videoverarbeitung

Codierung
Quantisierung und Entropiecodierung

Paul Berschin

Überblick

Nachtrag Bewegungsschätzung (HHI)

Quantisierung

Prinzipien

- Grundlagen
- Lloyd Max Verfahren
- Vektorquantisierer
- Dithering
- Anwendung in der Codierung

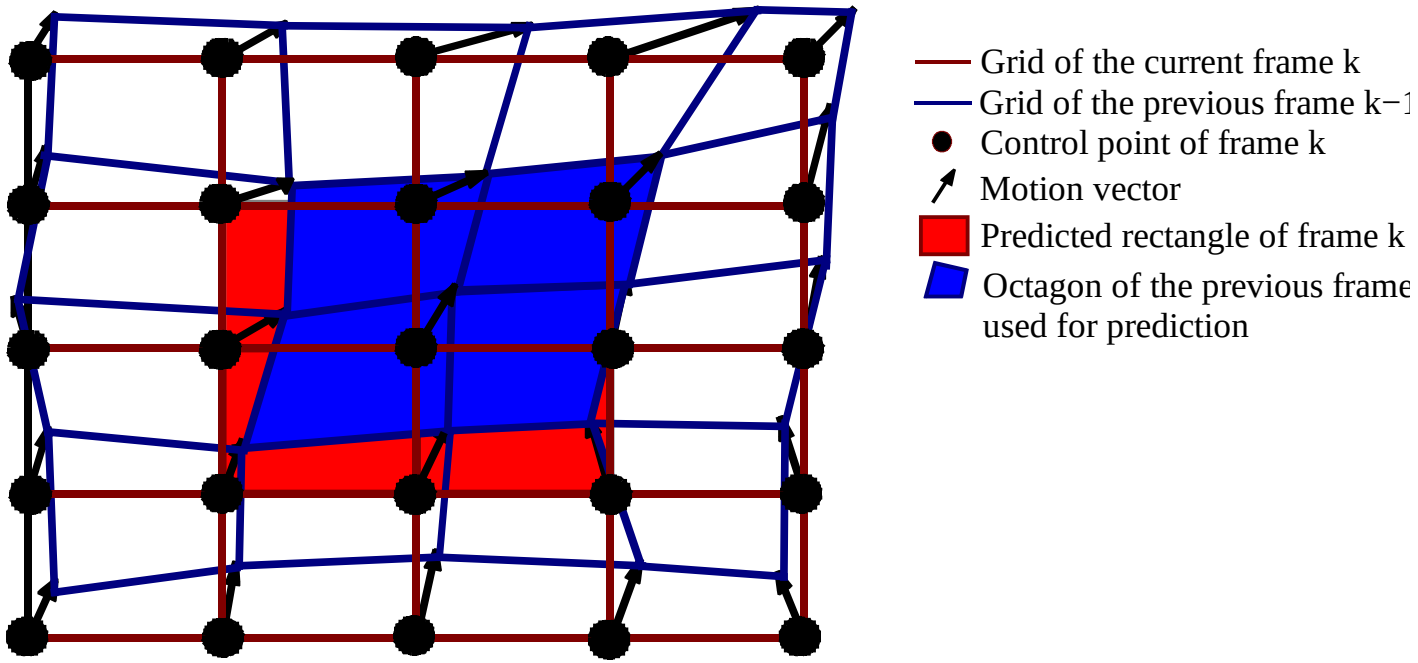
Redundanzreduktion

- Huffman Codierung
- Arithmethische Codierung

Coding Loop Filter

Nachtrag Bewegungsschätzung: Warping Prediction (HHI)

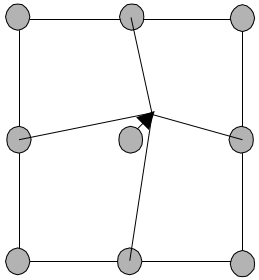
Vorschlag des HHI für H.26L:



Warping Prediction: Verfahren

Iterativer Algorithmus:

- Überlagerung des Bildes mit Netz−Gitter
- Schnittpunkte können bewegt werden → Verzerrung des Bildes
- Über 2−3 Iterationen:
  - Für jeden Kontrollpunkt:
    - Verschiebe den aktuellen Kontrollpunkt so, daß die Verzerrung der angrenzenden 4 Vierecke ein minimales MSE ergeben. (Gewöhnlich TSS mit 5 Schritten → Schrittweiten 8,4,2,1,0.5)



Erweiterung:

Hierarchische Gliederung der Suche bei großen Bewegungen
Beispiel: Zunächst 48x48 Blockgröße berechnen, dann 16x16

Quantisierung

Abbildung eines quasi−kontinuierlichen Signals x(n) auf eine (geringere) Anzahl diskreter Werte y(n)

$$y(n) = Q[x(n)]$$

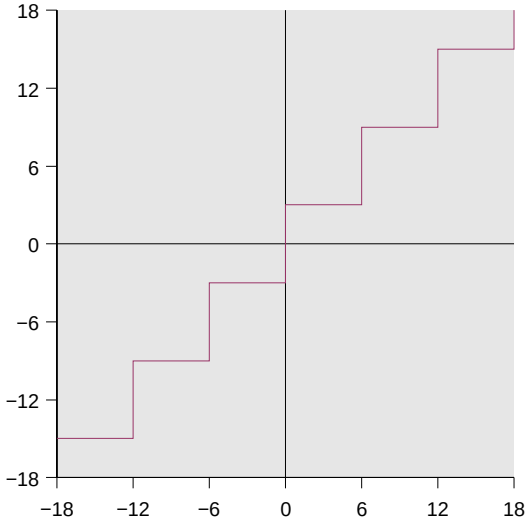
→ y soll eine gute Approximation von x darstellen

Quantisierer ist festgelegt durch seine Entscheidungsschwellen und die zugehörigen Repräsentationswerte

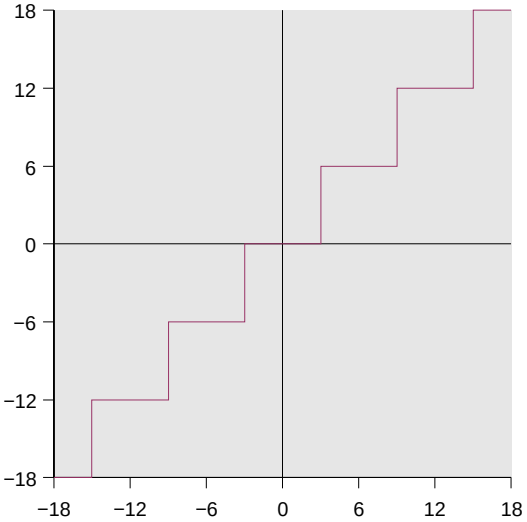
Üblicherweise sind Quantisierer symmetrisch zum Nullpunkt
Unterscheidung in:

- gerade und ungerade Anzahl von Schwellen
- lineare und nichtlineare Quantisierer
- Optimiert gemäß Wahrscheinlichkeitsdichtefunktion

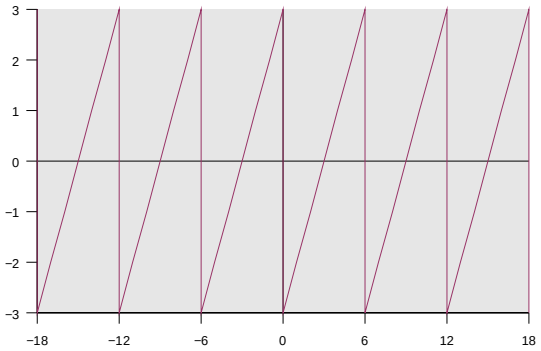
Quantisierung



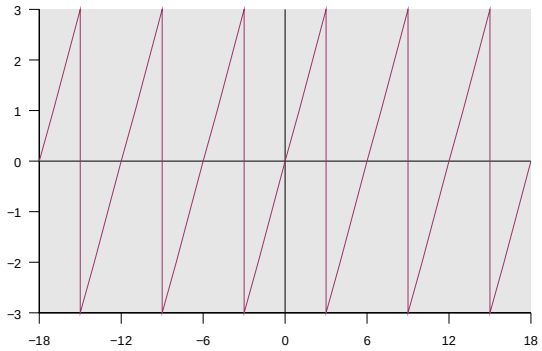
gerader Quantisierer



ungerader Quantisierer

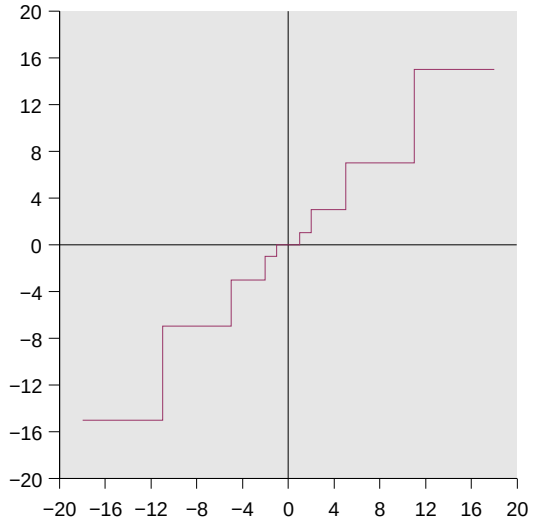


Quantisierungsfehler

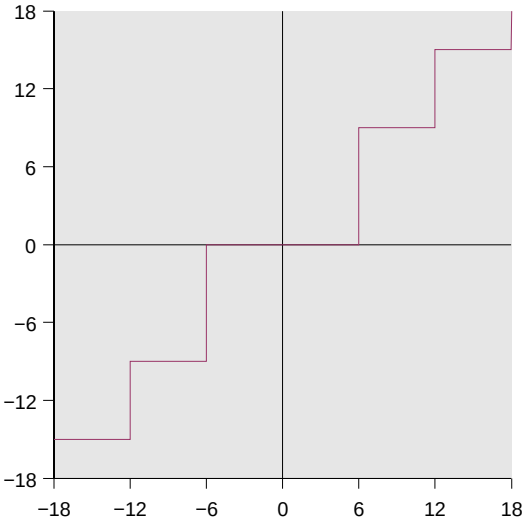


Quantisierungsfehler

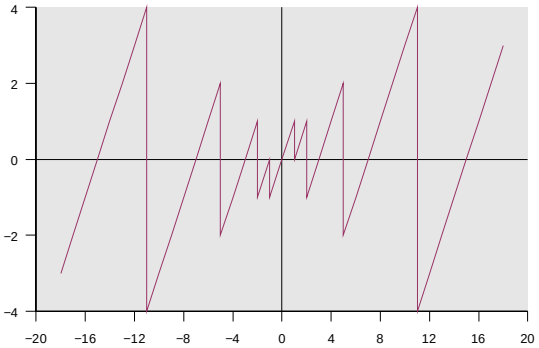
Quantisierung



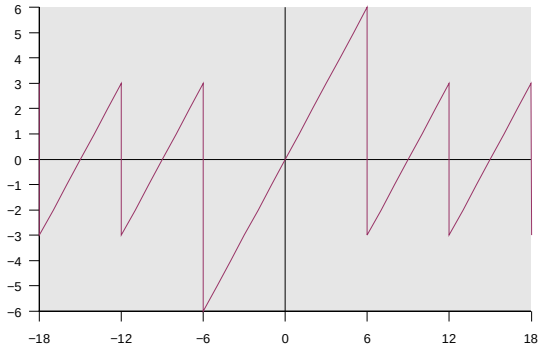
logarithmischer Quantisierer
(nichtlinearer Quantisierer)



Quantisierer mit Totzone



Quantisierungsfehler

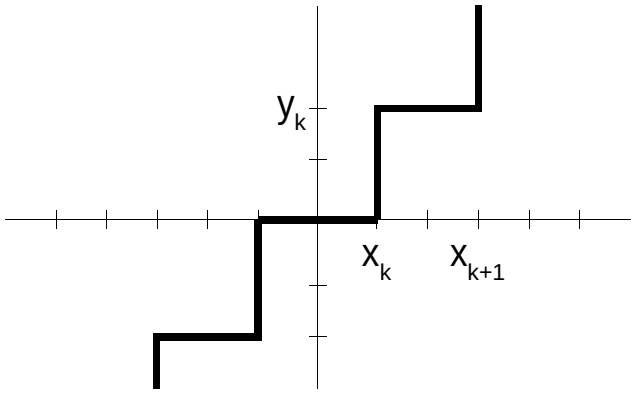


Quantisierungsfehler

Lloyd Max Quantisierer (1)

Anpassung an die Wahrscheinlichkeitsdichtefunktion
Quantisierungsfehler: q(x) = x − Q[x]
Wahrscheinlichkeitsdichtefunktion: p(x)
Varianz des Quantisierungsfehlers:

$$\sigma_q^2 = \int_{-\infty}^{\infty} (x - Q(x))^2 p(x)\,dx = \sum_{k=1}^{L} \int_{x_k}^{x_{k+1}} (x - y_k)^2 p(x)\,dx$$



Lloyd Max Quantisierer (2)

Optimierung des Quantisierers → Minimierung von σ

Iterative lösung:

Start mit gleichförmigen Quantisierer

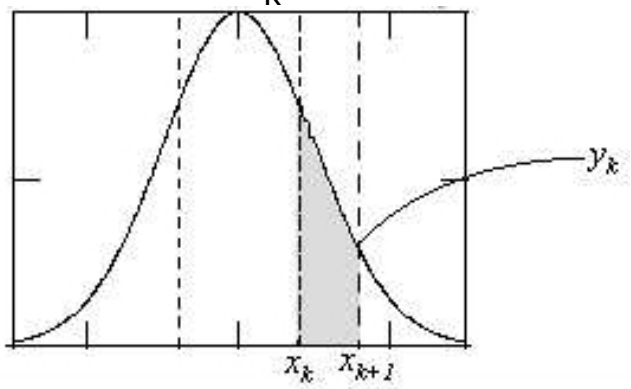
Bestimmung neuer Repräsentationswerte y_k:

$$y_k = \frac{\int_{x_k}^{x_{k+1}} x\,p(x)\,dx}{\int_{x_k}^{x_{k+1}} p(x)\,dx}$$

(Schwerpunkt der Wahr–scheinlichkeitsdichtefunktion)

Bestimmung neuer Entscheidungsschwellen x_k:

$$x_k = \frac{1}{2}(y_{k-1} + y_k)$$



Quantisierer (4)

Für eine gleichförmige Wahrscheinlichkeitsdichtefunktion generiert der Lloyd−Max Quantisierer einen gleichförmigen Quantisierer
In der Praxis werden gleichförmige Quantisierer gegenüber Lloyd−Max bevorzugt

- Einfache Implementierung mittels Division
- Einfach skalierbar zur Anpassung der Präzision

Ungerade Quantisierer haben den Vorteil der Darstellbarkeit der Null

Vektorquantisierer (1)

Eindimensionaler Quantisierer:
- quasi−kontinuierlicher Wert wird auf diskreten Wert abgebildet

Mehrdimensionale Quantisierer (Vektorquantisierer):
- quasi−kontinuierlicher Vektorraum wird auf einen Vektor aus einem vorher festzulegenden Codebuch abgebildet

Schwierigkeiten:
- Erzeugung des Codebuchs
- Zuordnung eines beliebigen mehrdimensionalen Vektors zu einem Vektor des Codebuchs (Auffinden des Codeindex)

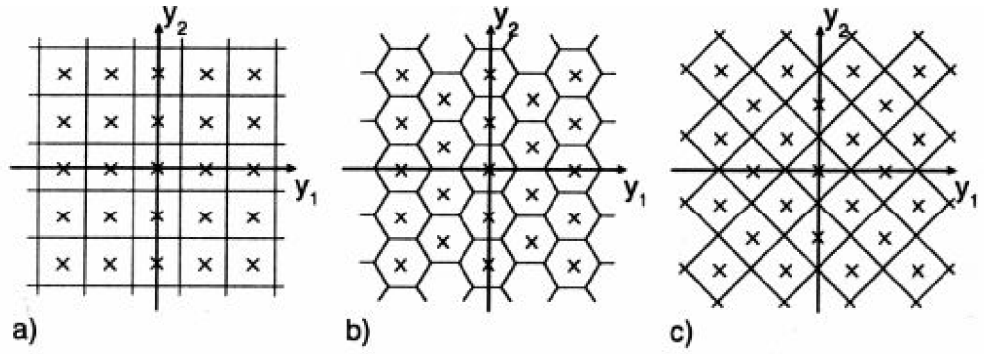
Vektorquantisierer (2)

Lattice−Vektorquantisierer: gleichförmiges Codebuch
nur geeignet für unkorreliertes Signal, für die Bildcodierung
ist vorher eine Dekorrelation nötig

a) Z_2 −Lattice

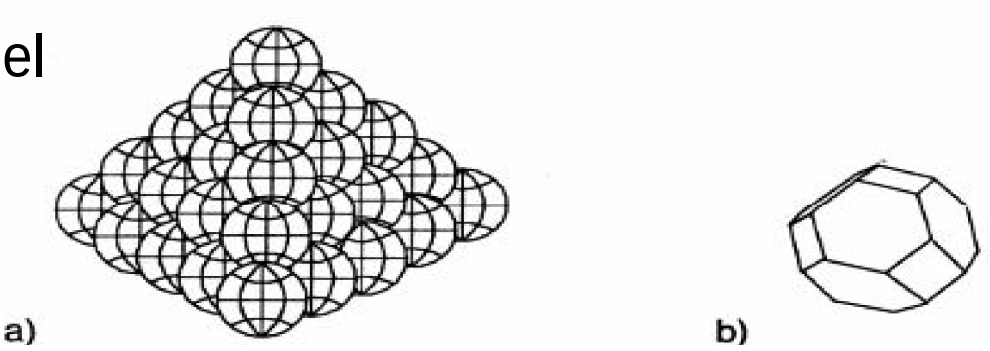
b) A_2 −Lattice

c) D_2 −Lattice



a) D_3 −Lattice als Kugelstapel

b) einzelne Voronoiregion



Vektorquantisierer (2)

Ungleichförmiges Codebuch:

Ersatz für eine Transformation, da gleichzeitige Ausführung von Dekorrelation und Quantisierung

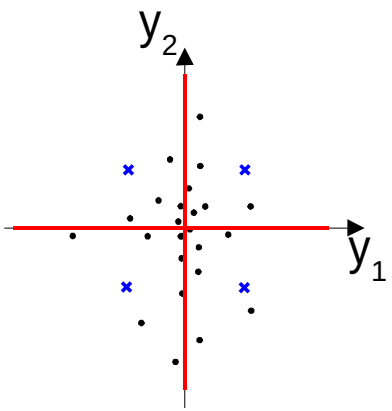
Erzeugung vergleichbar mit dem Entwurf eines Lloyd−Max Quantisierers

Mehrdimensionale Verteilungsdichtefunktion nur schwer beschreibbar → Erstellung des Codebuchs mit repräsentativer Trainingsfolge.
Anzahl der Trainingsvektoren > 200−fache Codebuchgröße.

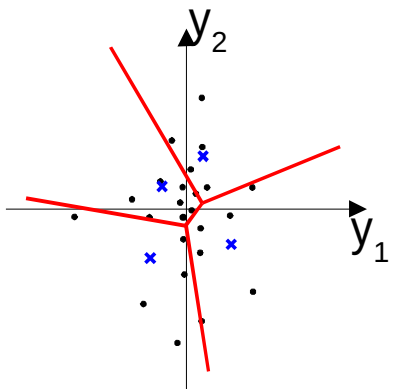
Vektorquantisierer (3)

Vorgehensweise:

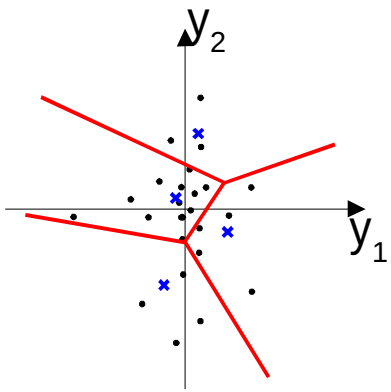
- 0) Start mit z.B. Lattice Quantisierer
- 1) Codieren der Trainingsfolge mit Codebuch
- 2) Berechnung der Gesamtverzerrung
- 3) Optimierung des Codebuches durch Neuberechnung der Clusterschwerpunkte → weiter bei 1)



Urspungs−codebuch



1.Iteration



2.Iteration

.

Dithering

Problem: Wertebereich wird auf einzelnen Wert abgebildet

→ Position innerhalb des Wertebereichs bleibt unberücksichtigt

Beispiel: Quantisierung auf Integer−Werte

{0.8, 0.8, 0.8, 0.8} → {1, 1, 1, 1}

{0.2, 0.2, 0.2, 0.2} → {0, 0, 0, 0}

Idee: Erhaltung des Mittelwertes bei mehreren Werten

Prinzip: Addition von Rauschen vor der Quantisierung

Obiges Beispiel:

{0.7, 0.9, 0.5, 1.3} → {1, 1, 0, 1}

{0.1, 0.3, −0.1, 0.7} → {0, 0, 0,1}

Auswirkung: Vermeidung von unnatürlich gleichförmigen Strukturen

Quantisierung der DCT−Koeffizienten (1)

Beispiel: H.263, MPEG 4

Linearer Quantisierer in 31 Skalierungsstufen mit gleichförmiger Stufenhöhe

→ Variation der Anzahl der Rekonstruktionswerte durch Wahl des Skalierungsparameters QP (1=fein .. 31=grob)

Einheitliche Quantisierung der Koeffizienten über alle Frequenzen

Ausnahme:

Quantisierer für INTRA DC Koeffizienten :

$$LEVEL = (COF + 4) / (2 \cdot 4)$$

Dequantisierer:

$$|REC| = 4 \cdot (2 \cdot |LEVEL|)$$

Sinn: möglichst genaue Rekonstruktion des INTRA DC −Wertes auch bei grobem Quantisierer

Quantisierung der DCT−Koeffizienten (2)

Quantisierer für INTRA AC Koeffizienten:

$$|LEVEL| = |COF| \;/\; (2 \bullet QP)$$

Quantisierer für INTER Koeffizienten:

$$|LEVEL| = (|COF| - QP/2) \;/\; (2 \bullet QP)$$

Dequantisierung für INTER und INTRA AC− Koeffizienten

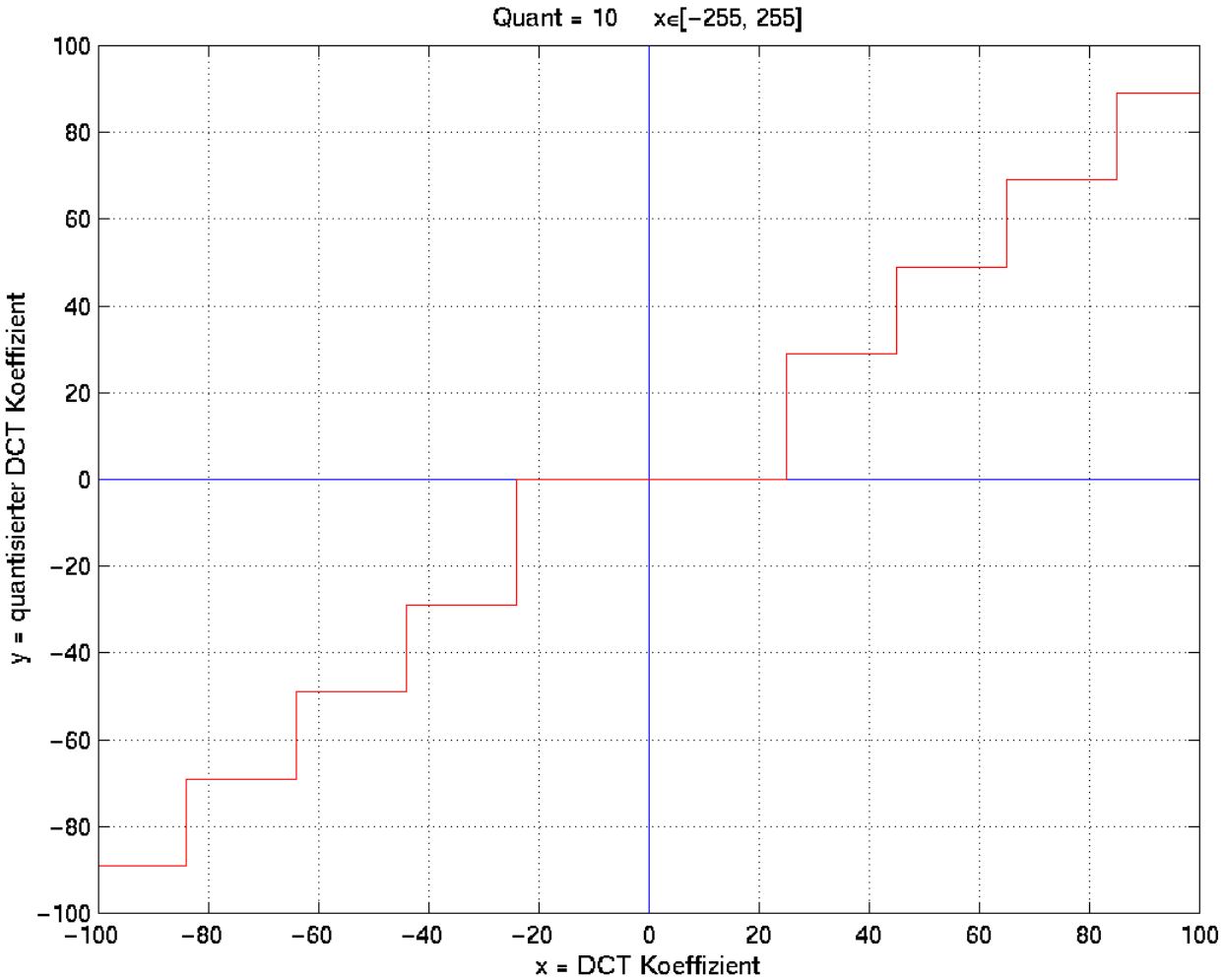
$$|REC| = QP \bullet (2 \bullet |LEVEL| + 1) \qquad \text{für ungerades QP}$$

$$|REC| = QP \bullet (2 \bullet |LEVEL| + 1) - 1 \qquad \text{für gerades QP}$$

Anmerkungen:

- Inter−Koeffizienten werden mit einer Totzone quantisiert
- Keine geradzahligen Rekonstruktionswerte, dadurch wird die Akkumulation von IDCT Fehlern verhindert

Quantisierung der DCT−Koeffizienten (3)



Quantisierung der DCT−Koeffizienten (4)

Problem:

Quantisierung beinhaltet Division

Lösungsansätze:

31 unterschiedliche Look−Up Tables für alle Koeffizientenwerte

Multiplikation mit Kehrwert und anschließendem Shift (dyadischer Bruch $\frac{z}{2^n}$ $n,z \in N$)

Beispiel: für QP = 17

|LEVEL| = (|COF| − QP/2) / (2 • QP)

$$|\text{LEVEL}| = (|\text{COF}| - (17 \gg 1) \cdot 30) \gg 10$$

1024 / (17 • 2) = 30,1176..

Quantisierung der DCT−Koeffizienten (5)

Blockquantisierung mit Matrix (MPEG2)
Für INTRA Blöcke:
Für jeden Koeffizient Wichtung der Quantisiererstufenhöhe mit einem zugeordneten Signifikanz−Wert
Ziel: Auch für INTRA Blöcke hohe Frequenzen zu Null setzen

default:

16	19	22	26	27	29	34
16	22	24	27	29	34	37
22	26	27	29	34	34	38
22	26	27	29	34	37	40
26	27	29	32	35	40	48
27	29	32	35	40	48	58
27	29	34	38	46	56	69
29	35	38	40	56	69	83

für Echtzeitberechnung günstiger:
Einmalige Übertragung einer neuen, einfacheren Wichtungsmatrix

16	16	16	16	32	32	32
16	16	16	32	32	32	32
16	16	32	32	32	32	32
16	32	32	32	32	32	32
32	32	32	32	32	32	32
32	32	32	32	32	32	64
32	32	32	32	32	64	64
32	32	32	32	64	64	64

Block−Quantisierer Steuerung

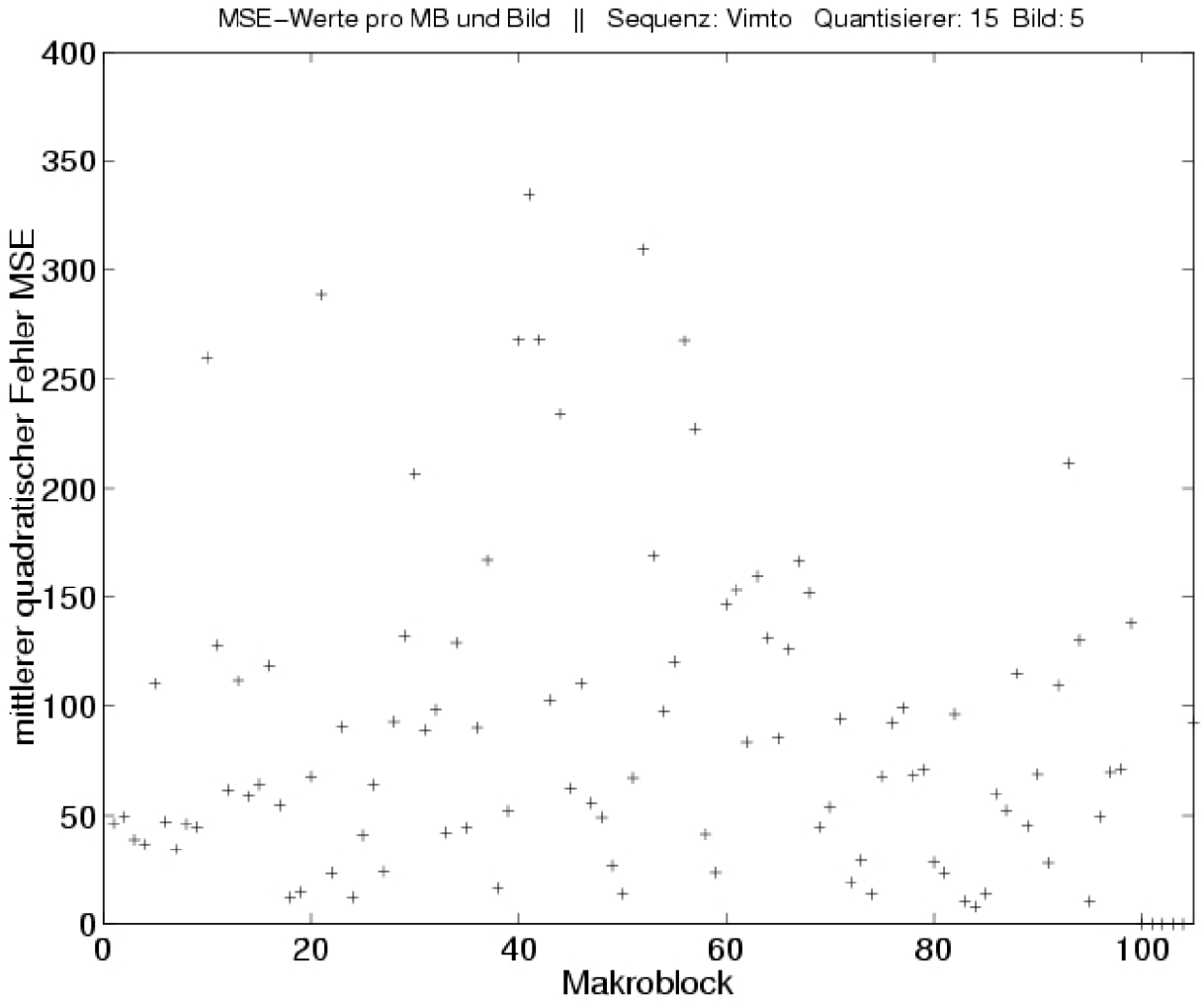
Verwendung eines festen Quantisierers für das ganze Bild

- Bitmenge pro Makroblock variiert stark
- Bildqualität variiert stark

Einstellung des Quantisierers für jeden Makroblock (16x16 Pixel)

- Zusätzliche Seiteninformation nötig
- Sehr rechenintensiv, da jede Quantisiererstufenhöhe getestet werden muß

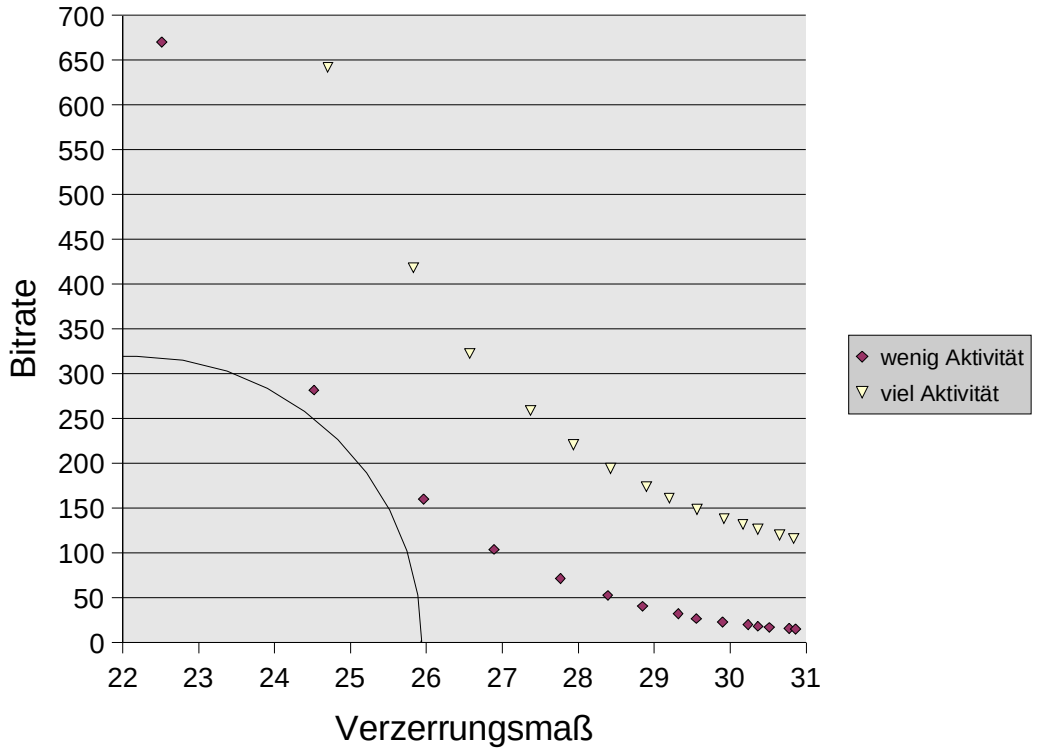
Fehlerverteilung bei konstantem Quantisierer



Rate–Distortion (RD) Ansatz

Idee: Beziehung zwischen Datenrate, Verzerrung und Quantisierung herstellen

Ansatz: Kompromiß durch Minimierung des Abstands zum Ursprung



RD–Darstellung für unterschiedliche Quantisiererstufenhöhen

Quantisierersteuerung auf Makroblockebene

Einfaches Verfahren (sehr rechenintensiv):

- Für jeden Makroblock:
  - Für jede Quantisiererstufenhöhe:
    - Quantisiere die Koeffizienten
    - Berechne die inverse DCT
    - Bestimme das Verzerrungsmaß gegenüber dem Original
    - Bestimme die benötigte Bitmenge
  - Bestimme den minimalen Abstand vom Ursprung
    → Wieviele Bits sollte dieser Makroblock anteilig erhalten
- Verteile die verfügbare Bitmenge auf die einzelnen Makroblöcke gemäß ihrer anteiligen Bitmenge und bestimme die nächst–liegende Quantisiererstufenhöhe

Quantisierersteuerung auf Makroblockebene

Geschwindigkeitssteigerung

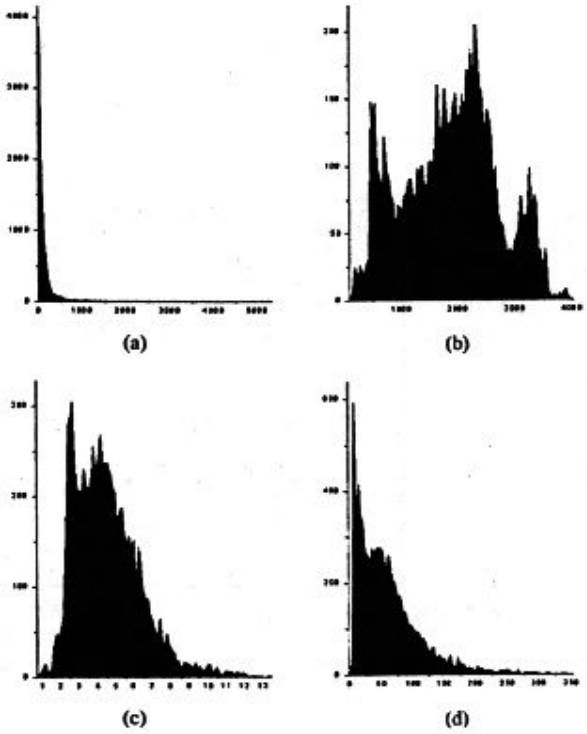
Idee: Bei Makroblöcken mit „hoher Aktivität“, d.h. geringer Korrelation liegt die RD−Kurve weiter vom Ursprung entfernt

Aktivitätsmaße (Auswahl):

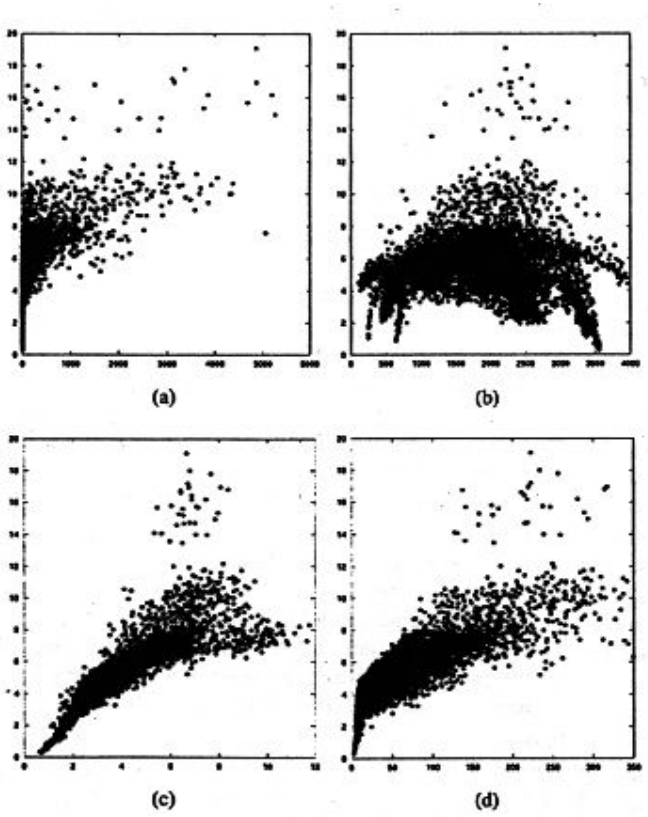
- Varianz
  Beispiel: Minimum der 4 Varianzen der 8x8−Blöcke
- DCT
  Beispiel: Normierte Summe aller AC−Koeffizienten
- Gradient
  Beispiel: Normierte Summe aller Spalten− und Zeilen−differenzen
- Kantendetektion
  Beispiel: Normierte Summe aller 3x3 Sobelfilter−Koeffizienten

Quantisierersteuerung auf Makroblockebene

Aktivitätsmaß−Auswahl: (Kim, Yi, Kim 7/1999)



Häufigkeitsverteilung

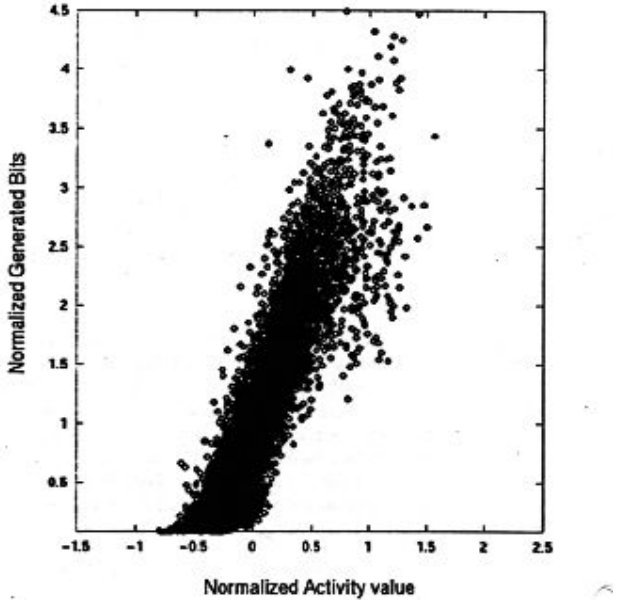


Korrelation mit nach RD−Verfahren optimierter Funktion

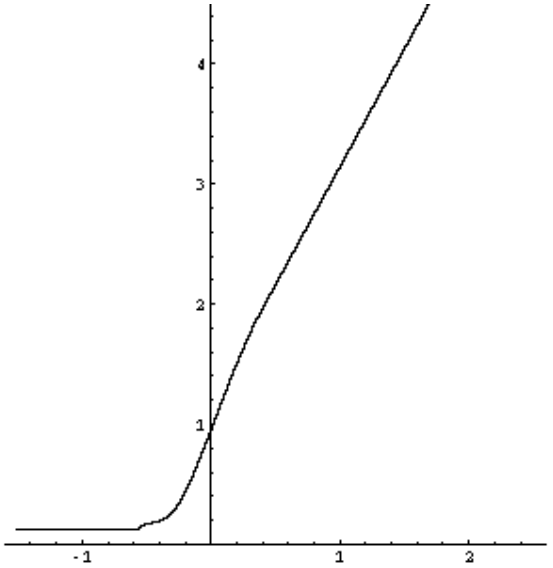
Vergleich der Aktivitätsmaße für RD−Optimierung

Verfahren	Geschwindigkeit	Häufigkeits−verteilung	Rate−Distortior Näherung
Varianz	1	–	–
DCT	4	+	–
Gradient	3	+	+
Kantendetektion	2	o	o

Erzeugte Bitmenge bei gegebener Aktivität (normiert):



Messung



Modell

RD−Optimierung mit Aktivitätsmaß−Verfahren

- Für jeden Makroblock
  - Berechne das Aktivitätsmaß nach dem Gradientenverfahren
  - Berechne den normierten Bitverbrauch für diese Aktivität nach der Modell−Kurve
- Berechne den normierten Gesamt−Bitbedarf und teile die verfügbare Bitmenge auf die Makroblöcke auf
- Für jeden Makroblock
  - Startintervall [1;31]
  - Solange das Intervall nicht aus einem Wert besteht
    - Berechne den Bitbedarf des QPs, der in der Mitte des Intervalls liegt
  - Halbiere die Intervallbreite entsprechend der geforderten Bitmenge

Anmerkung: Man beachte, daß weder die inverse DCT, noch das Qualitätsmaß berechnet werden müssen

Redundanzreduktion

Aufgabe:

Statistische Abhängigkeiten innerhalb eines Signals oder zwischen mehreren Signalen eleminieren

Idee:

Jedem zu übertragenen Symbol wird ein Code zugewiesen, dessen Länge umgekehrt proportional zu seiner Wahr−scheinlichkeit gewählt wird.

Verfahren:

- Huffman: Feste Zuordnung: 1 Code pro Symbol
- arithmetische Codierung: Mehrere Symbole werden gemeinsam codiert

Grundlagen

Definitionen:

Anzahl der Symbole: N

Symbol−Alphabet: $\{a_0,a_1,a_2,...a_{N-1}\}$

Wahrscheinlichkeiten: $\{P(a_0),P(a_1),P(a_2),...P(a_{N-1})\}$

Für statistisch unabhängiges Signal:

Wahrscheinlichkeit für das Auftreten der Sequenz $\{a_0,a_1\}$:

$$P(a_1|a_0)=P(a_0)\cdot P(a_1)$$

Informationsgehalt der Nachricht a_i in Bits/Symbol:

$$I(a_i)=-\log_2(P(a_i))$$

Informationsgehalt der Sequenz $\{a_0,a_1\}$ in Bits/Symbol:

$$I(a_0,a_1)=-\log_2(P(a_1|a_0))=-\log_2(P(a_0)\cdot P(a_1))=$$
$$[-\log_2(P(a_0))]+[-\log_2(P(a_1))]=I(a_0)+I(a_1)$$

Entropie

Entropie der Quelle (mittlerer Informationsgehalt):

$$H(U_0)=E(I(a_i))=-\sum_{i=0}^{N-1} P(a_i)\cdot\log_2(P(a_i))=-\sum_{u_0} P(u_0)\cdot\log_2(P(u_0))$$

Minimale Entropie der Quelle:

$H(U_0)=0$, für 1 Symbol mit 100% Wahrscheinlichkeit

→ Keine Übertragung notwendig (a priori bekannt)

Maximale Entropie der Quelle:

$H(U_0)=\log_2 N$, für gleiche Wahrscheinlichkeit aller Symbole

→ Für Einzelsymbolübertragung Binär−Code optimal

Codewortlänge

Mittlere Codewortlänge: $L_{av} = \sum_{i=0}^{i<N} L_i \cdot P(a_i)$

$L_{av} \geq H(U_0)$

Minimale mittlere Codewortlänge $L_{av}=H(U_0)$ für

$L(a_i) = I(a_i) = -\log_2(P(a_i))$

Voraussetzungen für Erreichbarkeit der minimalen Codewortlänge:

entweder alle $P(a_i)$ sind Zweierpotenzen

oder Symbole werden gruppiert

Redundanz eines Codes (Güte):

$R = L_{av} - H(U_0)$

Huffman−Code (1952)

Anwendung:

Unabhängige Codierung/Decodierung jeden Symbols
(unter diesen Bedingungen optimales Verfahren)

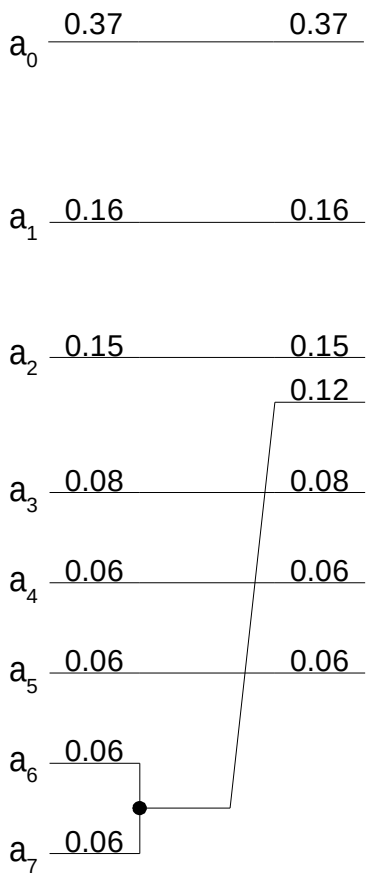
Verfahren:

Aufsuchen des Symbols in einer Tabelle und Übertragung des korrespondierenden Codes

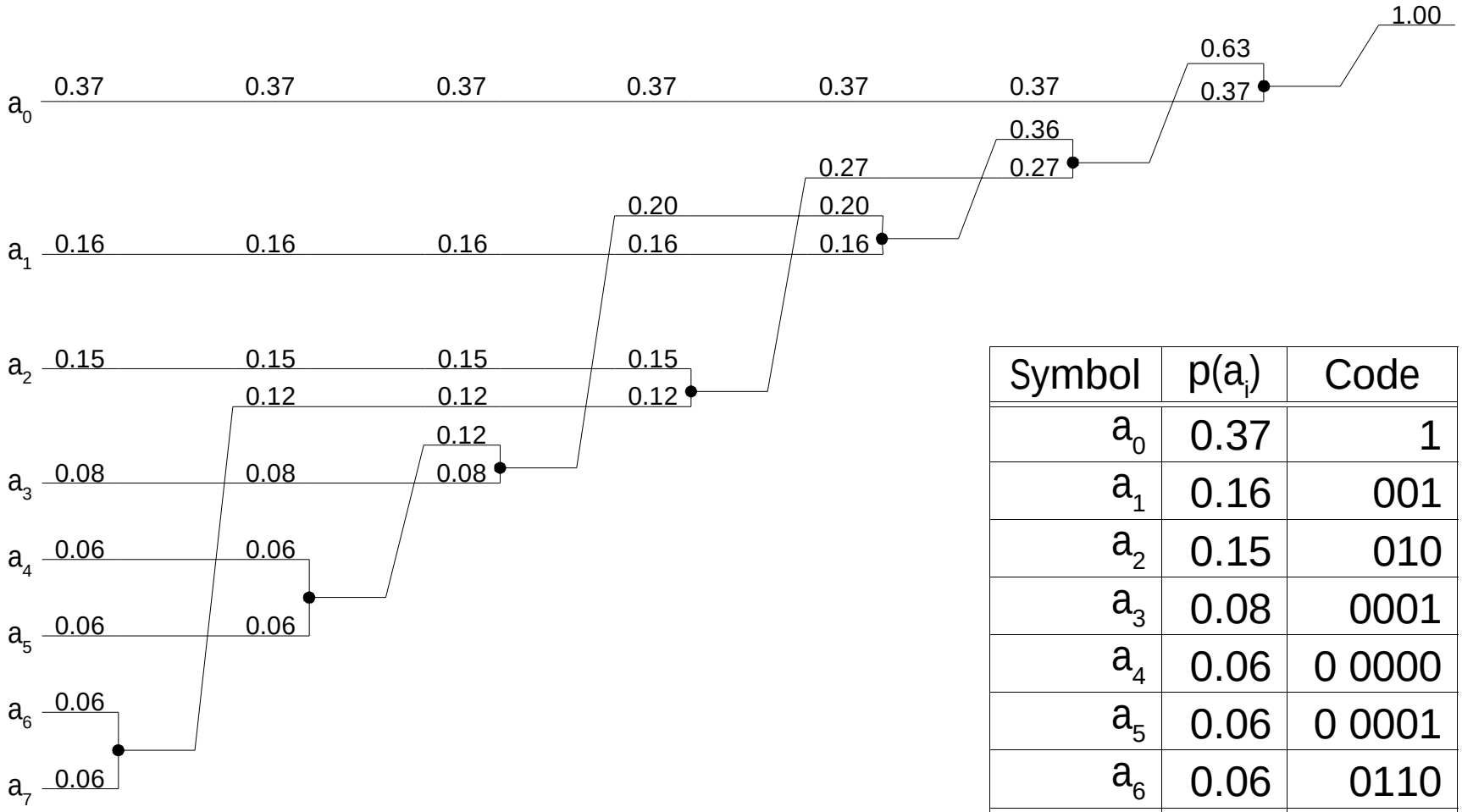
Algorithmus zur Generierung der Tabelle:

- Symbole nach Wahrscheinlichkeit sortieren
- Solange noch mehr als 1 Symbol übrig ist:
  - Die zwei Symbole mit der geringsten Wahrscheinlichkeit zusammenfassen, Wahrscheinlichkeiten addieren
  - Erneut sortieren
- Codes durch Ablaufen der Pfade ablesen (LSB first)

Huffman−Code Generierung 1.Schritt



Huffman−Code Generierung



Symbol	$p(a_i)$	Code
a_0	0.37	1
a_1	0.16	001
a_2	0.15	010
a_3	0.08	0001
a_4	0.06	0 0000
a_5	0.06	0 0001
a_6	0.06	0110
a_7	0.06	0111

Vergleich Huffman – Binär Code

Quelle			Binär–Code		Huffman–Code	
Symbol	$p(a_i)$	$I(a_i)$	Code	Länge	Code	Länge
a_0	0.37	1.43	000	3	1	1
a_1	0.16	2.64	001	3	001	3
a_2	0.15	2.74	010	3	010	3
a_3	0.08	3.64	011	3	0001	4
a_4	0.06	4.06	100	3	0 0000	5
a_5	0.06	4.06	101	3	0 0001	5
a_6	0.06	4.06	110	3	0110	4
a_7	0.06	4.06	111	3	0111	4

Entropie der Quelle: 2.63

Mittlere Codewortlänge für Binär–Code: 3 (Redundanz: 0.37)

Mittlere Codewortlänge für Huffman–Code: 2.7 (Redundanz: 0.07)

Decodierung

Einfachster Ansatz:

Einlesen von idx= [L_{max}–Bits] aus dem Bitstrom

Tabelle[idx] liefert Symbol und Codewortlänge

Nachteil: Bei vielen Symbolen mit stark variierender Wahrscheinlichkeit großes L_{max}, daher große Tabelle nötig

Index dezimal	Code (Index binär)	Symbol	verwendete Länge
0	**0 0000**	a_4	5
1	**0 0001**	a_5	5
2	**0 001**0	a_3	4
3	**0 001**1	a_3	4
4	**0 01**00	a_1	3
5	**0 01**01	a_1	3
6	**0 01**10	a_1	3
7	**0 01**11	a_1	3
8	**0 10**00	a_2	3
9	**0 10**01	a_2	3
10	**0 10**10	a_2	3
11	**0 10**11	a_2	3
12	**0 110**0	a_6	4
13	**0 110**1	a_6	4
14	**0 111**0	a_7	4
15	**0 111**1	a_7	4

Index dezimal	Code (Index binär)	Symbol	verwendete Länge
16	**1** 0000	a_0	1
17	**1** 0001	a_0	1
18	**1** 0010	a_0	1
19	**1** 0011	a_0	1
20	**1** 0100	a_0	1
21	**1** 0101	a_0	1
22	**1** 0110	a_0	1
23	**1** 0111	a_0	1
24	**1** 1000	a_0	1
25	**1** 1001	a_0	1
26	**1** 1010	a_0	1
27	**1** 1011	a_0	1
28	**1** 1100	a_0	1
29	**1** 1101	a_0	1
30	**1** 1110	a_0	1
31	**1** 1111	a_0	1

Decodierung

Erhöhte Speichereffizienz:

Aufspaltung in Einzeltabellen für Codewort−Teilbereiche

Voraussetzung: Erkennbare Regelmäßigkeit im Codeaufbau

Symbol	Code
a_0	1
a_1	0 01
a_2	0 10
a_3	0 001
a_6	0 110
a_7	0 111
a_4	0 000 0
a_5	0 000 1

Falls das 1.Bit = 1, dann decodiere zu a_0

Falls nächste 3 Bits ungleich 0, wende Tabelle zur Decodierung an

Falls das nächste Bit 0, dann a_4, sonst a_5

Weitergehende Verfahren zur Huffman−Codierung

Escape−Coding:

Bei unwahrscheinlichen Symbolen keine Zuweisung von eigenen Codes, sondern Sammelcode−Präfix und binäre Codierung des Symbols → Gleiche Wahrscheinlichkeiten

Beispiel: H.263 Koeffiziententabelle

Anzahl der Symbole: 32768 (64 Runs · 256 Levels · 2 Last)

davon 101 tabelliert (L_{max}=13)

sonst: Escape−Symbol

L_{escape}=7 (Escape Marker) + 6 (Run) + 8 (Level) + 1 (Last)

Arithmetische Codierung

Idee:

Steigerung der Effizienz durch gemeinsame Codierung mehrerer aufeinanderfolgender Symbole

Verfahren:

Startintervall für Codeintervall: [0 .. 1]

Startintervall für Sendeintervall: [0 .. 1]

Solange Symbole zu senden sind:

Codeintervallzerlegung entsprechend der Wahrscheinlichkeit der Symbole

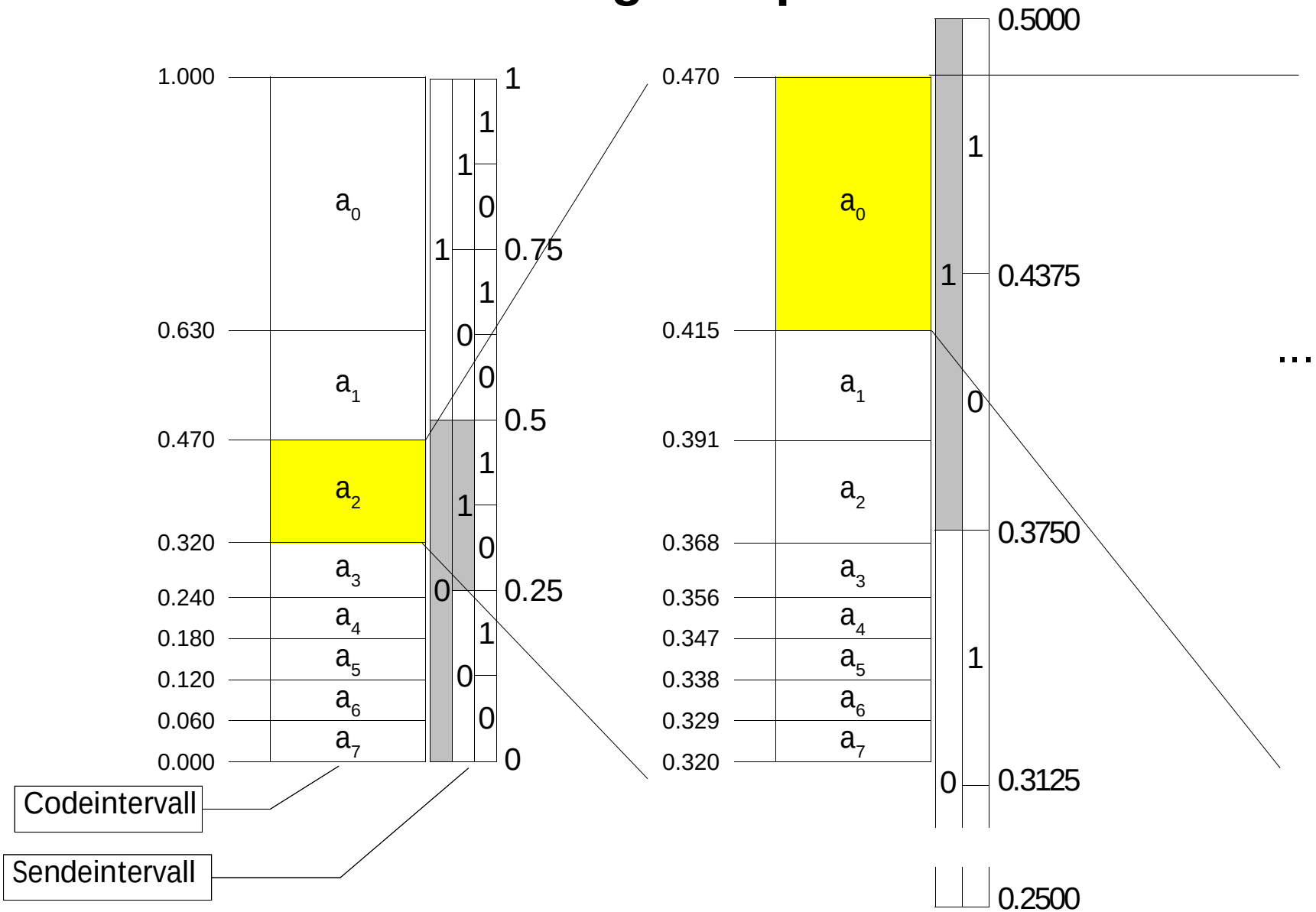
Auswahl desjenigen Codeintervalls, dessen Symbol gesendet werden soll

Solange das neue Intervall vollständig in einer Hälfte des Sendeintervalls liegt:

Übertragung von 1/0 für die obere/untere Hälfte

Reduktion des Intervalls

Arithmetische Codierung: Beispiel



Arithmetische Codierung: Eigenschaften

Vorteile:

- Anpassung an wechselnde Signalstatistik sehr einfach
- Beliebig lange Folgen gemeinsam codierter Symbole möglich

Nachteile:

- Algorithmisch komplexes Verfahren
- Schnelle Verfahren patentrechtlich geschützt

Fehlerrobustheit entropiecodierter Sequenzen

Problem:

- Bei einem einzigen Bitfehler ist der Code nicht mehr korrekt dekodierbar!

Lösungsansätze:

- Detektion:
  - Einfügen von bekannten, unikaten Bitmustern (Sync–Wörter)
  - Feste Übertragungslänge und Verwendung von Prüfsummen (Framing)
- Wiederherstellung der Information:
  - Anfrage nochmaliger Übertragung an den Sender
  - Verwendung von reversiblen VLCs: Lesbarkeit von rechts nach links und umgekehrt, daher kann aus zwei Richtungen bis zur fehlerhaften Information decodiert werden

Beispiel zu reversiblen VLCs

H.263+ Annex D:

Absolute value of vector difference in half−pixel units	number of bits	Codes
0	1	1
1	3	0s0
"x_0"+2 (2:3)	5	0$x_0$1s0
"x_1x_0"+4 (4:7)	7	0$x_1$1$x_0$1s0
"$x_2x_1x_0$"+8 (8:15)	9	0$x_2$1$x_1$1$x_0$1s0
"$x_3x_2x_1x_0$"+16 (16:31)	11	0$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_4x_3x_2x_1x_0$"+32 (32:63)	13	0$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_5x_4x_3x_2x_1x_0$"+64 (64:127)	15	0$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_6x_5x_4x_3x_2x_1x_0$"+128 (128:255)	17	0$x_6$1$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_7x_6x_5x_4x_3x_2x_1x_0$"+256 (256:511)	19	0$x_7$1$x_6$1$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_8x_7x_6x_5x_4x_3x_2x_1x_0$"+512 (512:1023)	21	0$x_8$1$x_7$1$x_6$1$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_9x_8x_7x_6x_5x_4x_3x_2x_1x_0$"+1024 (1024:2047)	23	0$x_9$1$x_8$1$x_7$1$x_6$1$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0
"$x_{10}x_9x_8x_7x_6x_5x_4x_3x_2x_1x_0$"+2048 (2048:4095)	25	0x_{10}1$x_9$1$x_8$1$x_7$1$x_6$1$x_5$1$x_4$1$x_3$1$x_2$1$x_1$1$x_0$1s0

Fehlerrobustheit bei Videocodierung

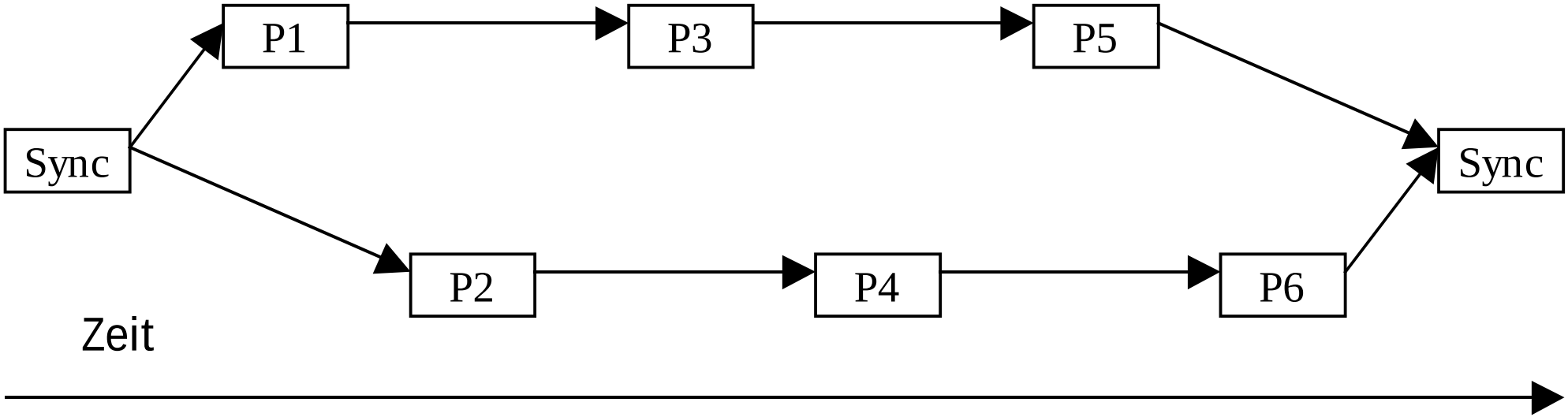
Problem:

Bei Echtzeitanwendung ist im Fehlerfall keine erneute Übertragung möglich

Lösungsansatz:

Unabhängige Decodierbarkeit von Teilinformationen vorsehen

z.B. durch Video Redundancy Coding (VRC):



Coding Loop Filter (1)

Idee:

Reduktion von Codierartefakten durch Filter, deren Wirkung dem Encoder bekannt ist

Prinzip:

Filter wird innerhalb der Rekonstruktionsschleife gerechnet

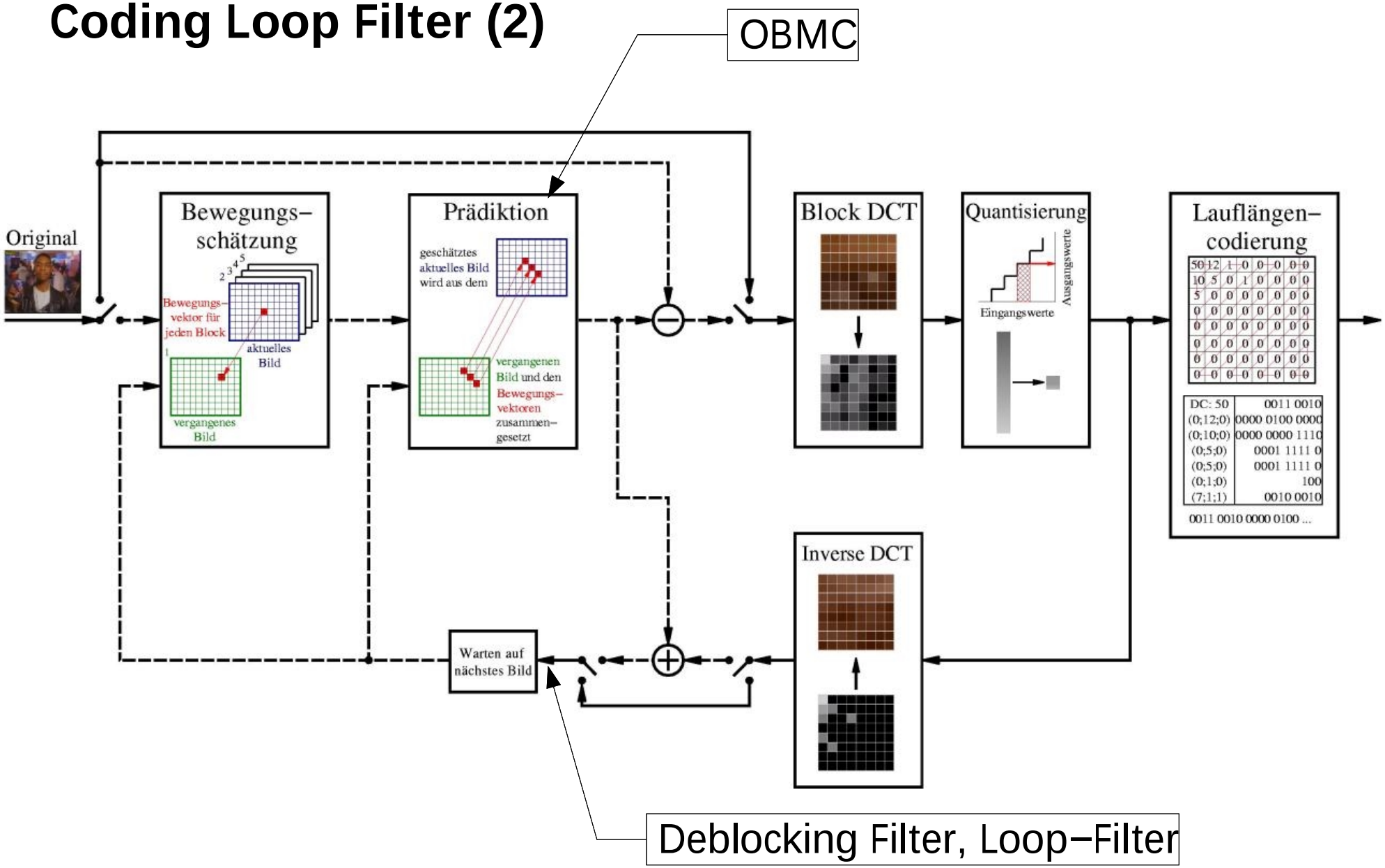
Nachteil:

Keine Skalierungsmöglichkeit bei Rechenzeitproblemen am Dekoder oder Encoder, Filter müssen immer gerechnet werden

Verwendete Filtertypen:

- Verwaschungsfilter
  - H.261: Loop Filter
  - H.263 und MPEG 4:
    - Overlapped Block Motion Compensation (OBMC)
- Blockkantenfilter
  - H.263: Deblocking Filter

Coding Loop Filter (2)



Coding Loop Filter (3)

H.261 Loop Filter:

- Zweidimensionales separierbares Filter
- Koeffizienten: 1/4,1/2,1/4
- Anwendung auf das gesamte prädizierte Bild

Coding Loop Filter (4)

Overlapped Block Motion Compensation (OBMC):

Erzeugung des Luminanzpixels aus gewichteter Summe dreier Prädiktionswerte

Generierung der Prädiktionswerte aus 3 Bewegungsvektoren:

1.) Aktueller Block

2.) Block oberhalb/unterhalb

3.) Block rechts/links

Wichtungsmatrizen für die gewichtete Summe:

aktueller

4	5	5	5	5	5	5	4
5	5	5	5	5	5	5	5
5	5	6	6	6	6	5	5
5	5	6	6	6	6	5	5
5	5	6	6	6	6	5	5
5	5	6	6	6	6	5	5
5	5	5	5	5	5	5	5
4	5	5	5	5	5	5	4

oberer/unterer

2	2	2	2	2	2	2	2
1	1	2	2	2	2	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	2	2	2	2	1	1
2	2	2	2	2	2	2	2

rechter/linker Block

2	1	1	1	1	1	1	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	2	1	1	1	1	2	2
2	1	1	1	1	1	1	2

Coding Loop Filter (5)

H.263 Deblockingfilter:

$d = (3A - 8B + 8C - 3D) / 16$

$d1 = \text{sign}(d) \cdot \max(0, |d| - \max(0, 2|d| - QP))$

$B_{neu} = B + d1$

$C_{neu} = C - d1$

zuerst auf horizontale, dann auf vertikale Blockkanten anwenden. Durch die Struktur von d1 werden kleine Unterschiede an Blockkanten als „blockiness“ eingeebnet, große als Bildkanten unverändert gelassen