

# Theorie und Implementation zu JPEG und JPEG2000

Studienarbeit  
von  
Roman Steiner & Marc Glaus  
Betreuer: Dr. Guido Schuster



JPEG bei einem Kompressionsfaktor von 97



# Inhaltsverzeichnis

Abstract .....	4
Aufgabenstellung .....	5
Kurzbeschreibung .....	5
Aufgabenstellung .....	5
Erwartete Ergebnisse .....	5
Theorie .....	6
1 Grundlagen der Komprimierungstheorie .....	6
1.1 Huffmann Codierung .....	6
1.2 Differentielle Pulscodemodulation .....	7
1.3 Arithmetische Codierung .....	8
2 JPEG .....	9
2.1 Verlustfreie Komprimierung .....	9
2.2 Verlustbehaftete Komprimierung .....	9
2.3 Baseline JPEG .....	10
2.3.1 Konvertierung des Bildes in den YCbCr-Farbraum .....	11
2.3.2 Farbsubsampling .....	12
2.3.3 Diskrete Cosinus Transformation (DCT) .....	12
2.3.4 Quantisierung der DCT-Koeffizienten .....	13
2.3.5 Codierung der Koeffizienten .....	14
2.3.6 Beispiel des Baseline JPEG .....	15
3 JPEG2000 .....	17
3.1 Einleitung .....	17
3.2 Warum ein neuer Standard? .....	17
3.3 Features von JPEG2000 .....	17
3.4 Hauptunterschiede zwischen JPEG und JPEG2000 .....	17
3.5 JPEG2000 der Algorithmus .....	17
3.5.1 Der YCbCr Farbraum .....	18
3.5.2 Die diskrete Wavelettransformation (DWT) .....	18
3.5.2.1 Was sind Wavelets? .....	18
3.5.2.2 Filterbank Implementation der diskreten Wavelettransformation .....	19
3.5.2.3 Die 2-Dimensionale diskrete Wavelettransformation .....	21
3.5.2.4 Die Wavelettransformation die JPEG2000 verwendet .....	22
3.5.3 Quantisierung .....	24
3.5.4 Entropiecodierung .....	24
3.5.5 Erläuterungen und Beispiele .....	24
3.5.5.1 Region-of-Interest (RIO) .....	24
3.5.5.2 Error resilience .....	24
3.5.5.3 Metadaten .....	24
Implementierung .....	25
4 JPEG .....	25
4.1 Allgemeines .....	25
4.2 Vorgehen und Stolpersteine .....	25
4.3 Funktionsbeschreibung .....	27
4.4 Erkenntnisse .....	32
5 JPEG2000 .....	41
5.1 Allgemeines .....	41
5.2 Vorgehen und Stolpersteine .....	41
5.3 Funktionsbeschreibung .....	41
5.4 Erkenntnisse .....	44
5.5 Nötige Ergänzungen .....	45
Schlusswort .....	46
Quellenverzeichnis .....	47
Anhang A JPEG SOURCECODE .....	
Anhang B JPEG2000 SOURCECODE .....	

## Abbildungsverzeichnis

Fig. 1 Das Morsealphabet .....	6
Fig. 2 Häufigkeiten zu Bsp.....	6
Fig. 3 Huffman Baum .....	6
Fig. 4 Huffman Tabelle .....	7
Fig. 5 Huffman, ASCII-Code.....	7
Fig. 6 Illustration zur arithmetischen Codierung.....	8
Fig. 7 DCT-basierter Codierer .....	9
Fig. 8 DCT-basierter Decodierer.....	9
Fig. 9 Sequential mode <i>Quelle 2</i> .....	10
Fig. 10 Progressiv mode <i>Quelle 2</i> .....	10
Fig. 11 Die Y,Cb,Cr - Komponenten .....	11
Fig. 12 Farbsubsampling von Cb, Cr .....	12
Fig. 13 JPEG Standard-Quantisierungstabelle für die Y-Komponente.....	13
Fig. 14 JPEG Standard-Quantisierungstabelle für die Cb, Cr-Komponenten .....	13
Fig. 15 Zic-Zac Verfahren.....	14
Fig. 16 Wertebereich der VLI's .....	15
Fig. 17 8x8 Pixelblock des Originalbild.....	15
Fig. 18 8x8 Pixelblock nach der DCT.....	15
Fig. 19 Quantisierungstabelle.....	16
Fig. 20 Quantisierter 8x8 Pixelblock.....	16
Fig. 21 Dequantisierter 8x8 Pixelblock .....	17
Fig. 22 Rekonstruierter 8x8 Pixelblock.....	17
Fig. 23 Schema eines Transformations- basierten Codierers .....	18
Fig. 24 Mexican-hat-wavelet .....	18
Fig. 25 Haar-wavelet.....	18
Fig. 26 Beispiel für die Zerlegung eines Signals mit dem Haar-wavelet.....	18
Fig. 27 Dekomposition mittels einer Filterbank.....	20
Fig. 28 Synthese des Ursprungssignals mittels einer Filterbank .....	20
Fig. 29 Dekompositionsschritte der DWT bei 2-dimensionalen Datensätzen .....	21
Fig. 30 Dekomposition von 2D-Daten nach der 3. Iteration .....	21
Fig. 31 Originalbild.....	22
Fig. 32 Transformationsergebnis nach der 2. Iteration.....	22
Fig. 33 Subband-dekomposition der tiles.....	22
Fig. 34 Daubechies 9/7 Analyse Filterkoeffizienten (LOSSY).....	23
Fig. 35 Daubechies 9/7 Synthese Filterkoeffizienten (LOSSY).....	23
Fig. 36 5/3 Analyse und Filterkoeffizienten (LOSSLESS).....	23
Fig. 37 Daubechies 9/7 Synthese Filterkoeffizienten (LOSSY).....	23
Fig. 38 Region Of Interest.....	24
Fig. 39 Komprimiert durch den JPEG Algorithmus ohne DC-level-shift .....	25
Fig. 40 Komprimiert durch ACDSec .....	25
Fig. 41 Komprimiert durch den JPEG Algorithmus mit DC-level-shift .....	26
Fig. 42 Komprimiert durch ACDSec .....	26
Fig. 43 lenna.bmp original Bild .....	33
Fig. 44 lenna.bmp jpeg-Codiert mit einem Qualitätsfaktor von 100 .....	34
Fig. 45 lenna.bmp jpeg-Codiert mit einem Qualitätsfaktor von 90 .....	35
Fig. 46 lenna.bmp jpeg-Codiert mit einem Qualitätsfaktor von 80 .....	36
Fig. 47 lenna.bmp jpeg-Codiert mit einem Qualitätsfaktor von 50 .....	37
Fig. 48 lenna.bmp jpeg-Codiert mit einem Qualitätsfaktor von 10 .....	38
Fig. 49 PSNR Diagramm von lenna.bmp.....	40
Fig. 50 PSNR versus q von lenna.bmp.....	40
Fig. 51 PSNR Diagramm von parrots.bmp .....	40
Fig. 52 PSNR versus q von parrots.bmp .....	40
Fig. 53 parrots.bmp .....	41
Fig. 54 lenna.bmp JPEG2000-codiert .....	45

## **Abstract**

Durch das Aufkommen von Multimedia Technologien wird von der Bildverarbeitung eine immer höhere Leistung gefordert. Unter Bildverarbeitung geht auch das Komprimieren von nicht bewegten Bildern (still image compression). Das heute wohl meist genutzte Verfahren ist JPEG, das auf der Diskreten-Cosinus-Transformation basiert. Da sich aber JPEG nur für natürliche Bilder eignet und nicht für computergenerierte, wird im Jahr 2001 ein neues Verfahren standardisiert. Das neue Verfahren JPEG2000 beruht auf der Wavelettransformation. JPEG2000 soll den bewährten JPEG-Algorithmus nicht ersetzen, sondern ergänzen.

Im folgenden wird auf die Theorie und die Implementierung der beiden Algorithmen eingegangen.

## Aufgabenstellung

### Kurzbeschreibung

Da die Datenmenge für ein unkomprimiertes Bild sehr hoch ist, ist die Bildkompression eine wichtige Anwendung der digitalen Bildverarbeitung. Die heutzutage wichtigste Methode für die Bildkompression ist JPEG. Der JPEG Algorithmus gehört zur Klasse der Transformations basierten Codierer, welche ein Bild zuerst in einen Frequenzbereich transformieren, im Frequenzbereich quantisieren und dann die quantisierten Koeffizienten codiert. Der heutige Algorithmus, JPEG, benutzt die Diskrete Cosinus Transformation. Der neue Algorithmus wird JPEG2000 sein, der auf den Wavelets basiert.

### Aufgabenstellung

- Erstellung eines Projektplans, welcher innert 3 Wochen abgegeben werden muss.
- Erarbeitung und Darstellung der erforderlichen theoretischen Grundlagen über JPEG und JPEG2000
- Implementation eines einfachen JPEG -Coders in Matlab/Simulink, welcher jeden Schritt des Algorithmus graphisch darstellt.

### Erwartete Ergebnisse

- Matlab/Simulink Programme des JPEG -Coders, welche jeden Schritt des Algorithmus graphisch darstellen.
- Dokumentation der Implementierung
- Dokumentation der Probleme und deren Lösungen
  - Literaturstudium JPEG, JPEG2000
  - Vergleich JPEG, JPEG2000
  - Dokumentation des Zusammenhanges zwischen der Bitrate und dem Signalrauschabstand eines codierten Bildes

# Theorie

## 1 Grundlagen der Komprimierungstheorie

### 1.1 Huffmann Codierung

Die Idee der Huffman Codierung geht auf das Prinzip des Morsealphabets zurück. Dort werden häufig vorkommende Symbole (in diesem Fall Buchstaben) kürzere Codes zugeordnet als den seltener vorkommenden.

Buchstabe	Morsezeichen	Buchstabe	Morsezeichen	Buchstabe	Morsezeichen	Buchstabe	Morsezeichen
A	•--	H	••••	O	---•	V	••••
B	--••	I	••	P	•--•	W	•---
C	--•--	J	•---	Q	---•--	X	--•••
D	--••	K	--•--	R	•--•	Y	--•---
E	•	L	•--••	S	•••	Z	--•••
F	••--•	M	---	T	--		
G	---•	N	--•	U	••--		

Fig. 1 Das Morsealphabet

Der Huffman-Algorithmus lässt sich am besten an einem Beispiel erklären. In diesem Fall sollen die Symbole Buchstaben in einer Textdatei sein. Dazu muss nun also zunächst eine Tabelle erzeugt werden, die angibt wie häufig jedes einzelne Symbol in der Datei vorkommt. In diesem Beispiel soll davon ausgegangen werden, dass die Datei aus 5 verschiedene Buchstaben (Symbole) besteht, die in folgender Häufigkeit vorhanden sind:

Symbol	A	B	C	D	E
Häufigkeit	15	7	6	6	5

Fig. 2 Häufigkeiten zu Bsp.

Als nächstes wird ein Binärbaum erstellt dessen Blätter die Symbole mit ihrer Häufigkeit darstellen und nach folgenden Regeln aufgebaut wird :

- Die beiden freien Knoten mit der geringsten Häufigkeit werden ermittelt
- Ein Elternknoten wird für diese beiden Knoten erzeugt. Diesem Elternknoten wird eine Häufigkeit zugewiesen, die der Summe der beiden Kindknoten entspricht. Der Elternknoten wird der Liste der freien Knoten hinzugefügt und die zwei Kindknoten werden aus der Liste entfernt.
- Einem der beiden Kindknoten wird ein 0-Bit (links) und dem anderen ein 1-Bit (rechts) zugewiesen.
- Die oben genannten Schritte werden solange ausgeführt, bis nur noch ein freier Knoten übrig ist. Dieser freier Knoten wird als Wurzel des Baumes bezeichnet.

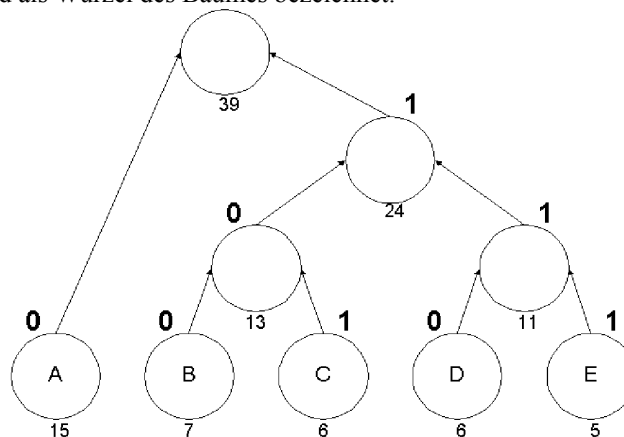


Fig. 3 Huffmann Baum

Um jetzt die Codes für die einzelnen Symbole zu erhalten, muss von der Wurzel zu den Blättern gewandert werden. Danach ergibt sich folgende Codetabelle :

Symbole	Huffmann Code
A	0
B	100
C	101
D	110
E	111

Fig. 4 Huffmann Tabelle

Die folgende Tabelle zeigt deutlich die Reduktion der Daten, wenn man die Datei nach Huffman codiert. Dabei muss allerdings berücksichtigt werden, dass in der nach Huffman codierten Datei auch die Huffman-Codetabelle enthalten sein muss.

Symbol	Häufigkeit	Huff. Länge	Huffman Bit	ASCII Länge	ASCII Bit
A	15	1	15	8	120
B	7	3	21	8	56
C	6	3	18	8	48
D	6	3	18	8	48
E	5	3	15	8	40
<b>Gesamt bits(bytes)</b>			87(11)		312(39)

Fig. 5 Huffmann, ASCII-Code

## 1.2 Differentielle Pulscodemodulation

Die differenzielle Pulscodemodulation (DCPM) versucht, die Wertebereiche von numerischen Eingabezeichen zu verkleinern und dadurch eine bessere Entropie Codierung zu erreichen. Anstatt die Eingabezeichen direkt zu codieren, wird die Differenz zu einem Referenzwert codiert.

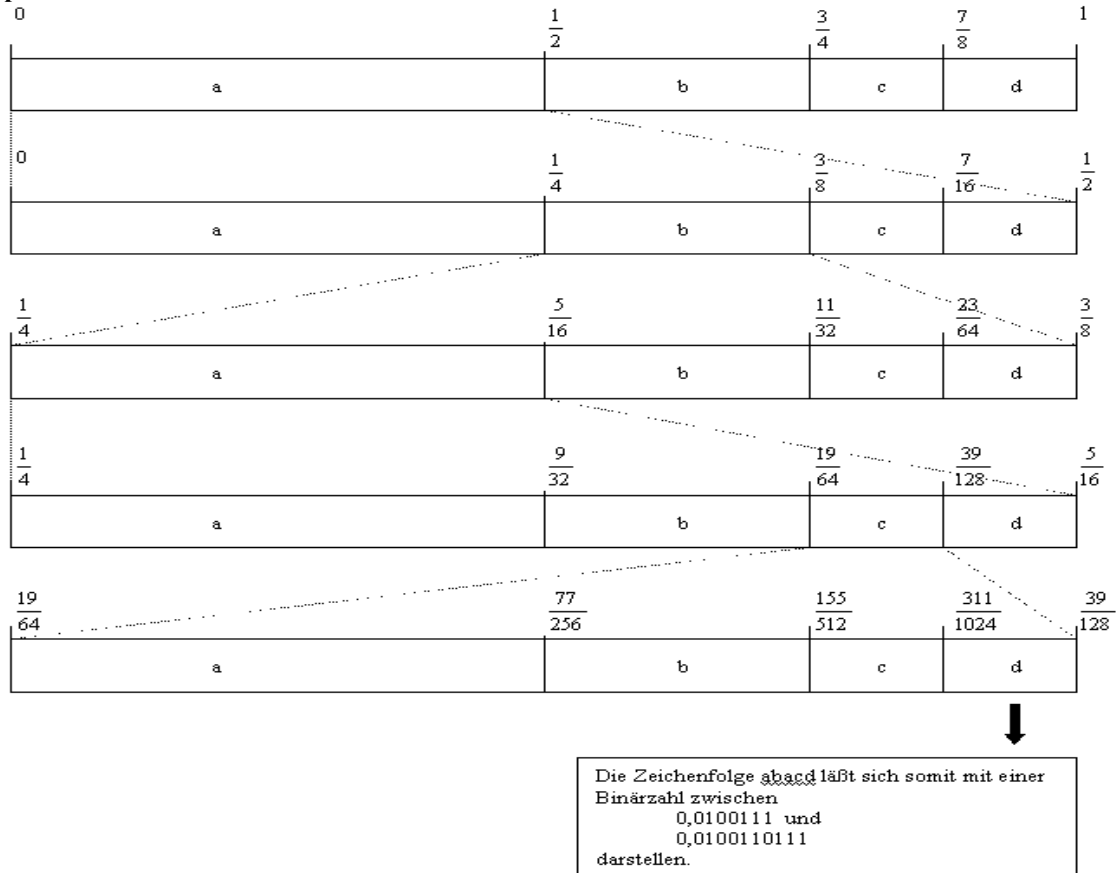
### Beispiel:

Die Folge 10, 12, 14, 16, 18, 20 soll mit der Lauflängen Codierung als Entropie Codierung codiert werden. Auf direktem Weg wird keine Kompression erreicht, da sich alle aufeinanderfolgenden Zahlen unterscheiden. Eine mögliche DCPM Codierung wäre die Folge: 10, 2, 2, 2, 2, 2. Hierbei dient jeweils die in der Ausgangsfolge vorhergehende Zahl als Referenzwert und die anschließende Lauflängen Codierung erzielt eine Kompression.

### 1.3 Arithmetische Codierung

Bei der arithmetischen Codierung werden die zu codierenden Zeichen an reellen Zahlenintervallen in  $[0,1]$  zugeordnet. Je länger die zu codierende Zeichenfolge ist, desto enger werden die Zahlenintervalle. Bei der Berechnung der Zahlenintervalle werden die Wahrscheinlichkeiten der Eingabezeichen berücksichtigt. Die Wahrscheinlichkeiten können dynamisch angepasst werden. Das Kompressionsergebnis ist gut, da Zeichenreihen und nicht einzelne Zeichen codiert werden.

**Beispiel:**



**Fig. 6 Illustration zur arithmetischen Codierung**

Im obigen Beispiel besteht das Eingabealphabet aus den Zeichen **a**, **b**, **c** und **d** mit den Auftretswahrscheinlichkeiten  $p(a)=1/2$ ,  $p(b)=1/4$ ,  $p(c)=1/8$  und  $p(d)=1/8$ . Es ist dargestellt, wie sich die Intervalle entwickeln, wenn die Zeichenreihe **abacd** codiert werden soll. Die Zeichenreihe **abacd** liegt z.B. im Intervall  $[311/1024, 312/1024 = 39/128)$ . Sie kann durch eine beliebige Zahl aus diesem Intervall dargestellt werden. Im Beispiel wurden Vielfache von  $1/2$ -Potenzen verwendet. Zur Decodierung müssen die Zeichenwahrscheinlichkeiten bekannt sein und die Länge (das Ende) der codierten Zeichenreihe.

## 2 JPEG

JPEG ist ein Standard, der möglichst allen Anforderungen bei der Bilddatenkompression gerecht werden soll. Da es nicht einen einzelnen Algorithmus gibt, der alles kann, legte das JPEG-Gremium (Joint Picture Expert Group) verschiedene Verfahren fest, die jeweils für einen bestimmten Teilbereich sehr gute Ergebnisse liefern. Grundsätzlich gibt es zwei Grundverfahren:

- verlustfreie Komprimierung (lossless compression)
- verlustbehaftete Komprimierung (lossy compression)

### 2.1 Verlustfreie Komprimierung

Das verlustfreie Kompressionsverfahren ist eine Methode, die Bilddaten so zu komprimieren, dass sie nach der Dekompression aufs Bit genau den Originaldaten entsprechen. JPEG stellt hierzu einen Operationsmodus zur Verfügung, den lossless mode. Kompressionsverfahren arbeiten alle nach dem gleichen Schema:

Die anfallenden Bilddaten werden zuerst in eine Menge von Deskriptoren transformiert. Für die Deskriptoren wird nun ein statisches Modell (Verschlüsselungstabelle) erstellt und anhand dieses Modells codiert.

Die verlustfreie Komprimierung verwendet als Transformation die differentielle Pulscodemodulation. Zum Codieren der Daten benutzt man die Huffman-Codierung oder die arithmetische Codierung. Die verlustfreie Komprimierung wird überall dort benutzt, wo Fehler nicht toleriert werden (z. B. bei mechanischer Bildauswertung).

### 2.2 Verlustbehaftete Komprimierung

Bei der verlustbehafteten Kompression wird das zu komprimierende Bild in einen neuen Farbraum (YCbCr) konvertiert. Nach der Farbkonversion erfolgt eine DC Verschiebung (DC-Level-shift, der „Gleichstromanteil“ wird entfernt). Danach wird es transformiert (Diskrete Cosinus Transformation), quantisiert und codiert. Die Bildinformationsverluste entstehen bei der Quantisierung. Die Bilddaten werden wie oben erläutert komprimiert. Die Dekompression erfolgt durch rückläufiges Anwenden (siehe Graphik).

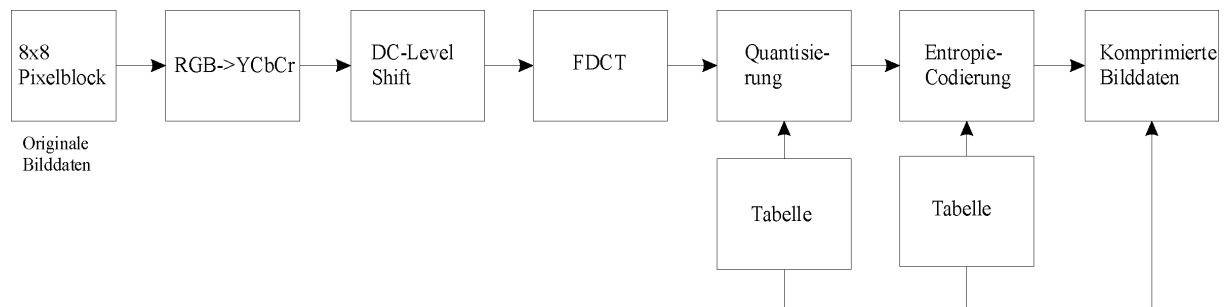


Fig. 7 DCT-basierter Codierer

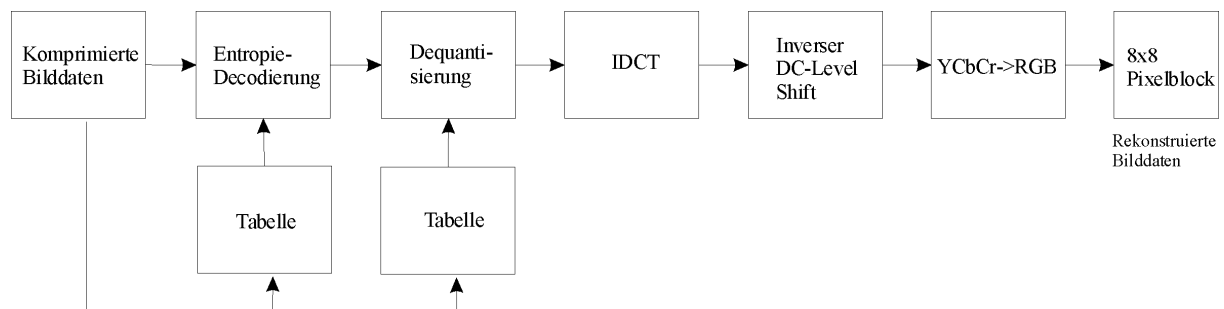


Fig. 8 DCT-basierter Decodierer

**Die verlustbehaftete Kompression sieht 3 Operationsmodi vor:**

- **Sequential mode:**  
Dieser Modus wird heute in den meisten Anwendungen, die JPEG-Bilder verwenden, genutzt. Er wird auch Baseline-JPEG genannt. Beim sequential mode werden die Bilder horizontal und vertikal transformiert und die jeweiligen Farbenen (Y-, Cb- und Cr-Ebene, siehe 2.3.1) einzeln behandelt. Dieses Verfahren ermöglicht es, sowohl Graustufen- als auch Echtfarbbilder mit sehr hohen Kompressionsraten und relativ geringen Qualitätsverlusten zu komprimieren.

Fig. 9 Sequential mode *Quelle 2*

- **Progressiv mode:**  
Bei diesem Modus wird das Bild mehrfach codiert, um so die Bildqualität schrittweise zu erhöhen. Dieses Vorgehen bietet sich insbesondere bei langsamen Datenübertragungskanälen an. Der Empfänger bekommt schnell ein Bild geliefert, das zunächst unscharf ist, dann aber etappenweise verfeinert wird. Man bekommt ziemlich schnell einen Überblick über das übertragene Bild und kann die Übertragung abbrechen, wenn die Bildqualität ausreichend ist.

Fig. 10 Progressiv mode *Quelle 2*

- **Hierarchical mode:**  
Dieser Modus ist eine Form des progressive mode. Der hierarchical mode verwendet eine Menge von Bildern mit immer groberer Auflösung, die durch filtern mit einem Tiefpass und mitteln von mehreren Pixelwerten, erzeugt wird. Zunächst wird das Bild mit der tiefsten Auflösung codiert. Dieses dient wieder als Basis für eine Vorhersage auf das Bild mit der nächstgrösseren Auflösung. Dieser Vorgang wird wiederholt, bis die volle Auflösung erreicht ist. Hauptanwendungsgebiet dürften grosse Datenbanken sein, die die niedrigen Auflösungen für ihre Inhaltsverzeichnisse verwenden und nur bei Bedarf die höhere Auflösung decodieren.

## 2.3 Baseline JPEG

Der Baseline JPEG arbeitet nach dem sequential mode und ist der weitverbreiteste JPEG-Algorithmus. Er besteht aus 5 Schritten:

- Konvertierung des Bildes in den YCbCr-Farbraum
- DC-Level-Shift
- Farbsubsampling
- Diskrete Cosinus Transformation
- Quantisierung der DCT-Koeffizienten
- Codierung der Koeffizienten (Entropie Codierung)

### 2.3.1 Konvertierung des Bildes in den YCbCr-Farbraum

Grundsätzlich werden bei digital gespeicherten Bildern zwei Codierungsdaten verwendet: Pixel- und Vektorformate. Die Pixelformate sind im Bereich der graphischen Darstellung am meisten verbreitet. Sie teilen ein Bild in Zeilen und Spalten auf und speichern den Farbwert jedes Bildpunktes. Der Farbwert befindet sich im sogenannten RGB-Farbraum, welcher Auskunft gibt, aus wie vielen Rot-, Grün- und Blau-Anteile der Bildpunkt zusammengesetzt ist. Normalerweise wird eine Farbtiefe von 24-bit verwendet, dies entspricht 16777216 Farben. Da sich gezeigt hat, dass das menschliche Auge Farbwerte in einer geringeren Auflösung als Helligkeitswerte wahrnimmt, wird der gängige RGB-Farbraum in den neuen Farbraum YCbCr umgewandelt:

- Y (Luminanz) = Grundhelligkeit
- Cb (Chrominanz) = Mass für die Abweichung von der Mittelfarbe Grau in Richtung BLAU
- Cr (Chrominanz) = Mass für die Abweichung von der Mittelfarbe Grau in Richtung ROT

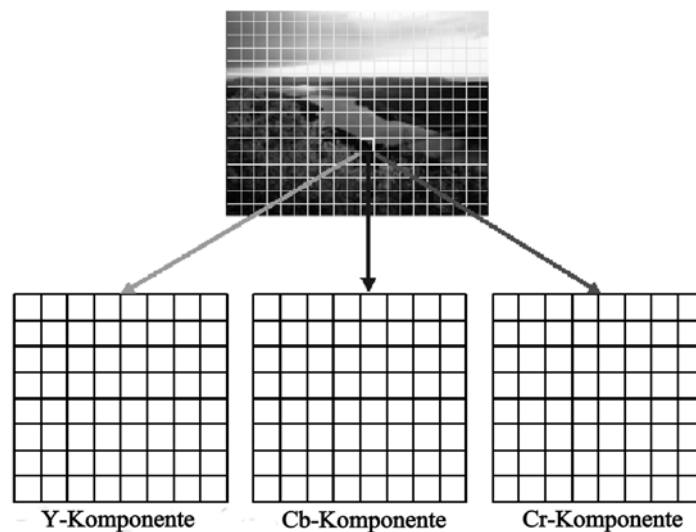


Fig. 11 Die Y,Cb,Cr - Komponenten

Um nun das Bild vom RGB-Farbraum in den YCbCr-Farbraum umzurechnen, benötigt man folgende Formel:

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.1687 & -0.3313 & 0.5 \\ 0.5 & -0.4187 & -0.0813 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

Die Rücktransformation vom YCbCr-Farbraum in den RGB-Farbraum erfolgt nach der Formel:

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1.0 & 0.0 & 1.402 \\ 1.0 & -0.34414 & -0.71414 \\ 1.0 & 1.772 & 0.0 \end{pmatrix} \cdot \begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix}$$

### 2.3.2 Farbsubsampling

Da das Auge feine örtliche Unterschiede in Cb und Cr nur schlecht wahrnimmt, macht es wenig aus, wenn man an der Ortsauflösung von Cb und Cr spart. Dies geschieht, indem für ein kleines Gebiet, die Werte Cb und Cr gemittelt werden, und anstatt Cb und Cr in jedem Punkt zu codieren, werden nur die gemittelten Werte des gesamten Gebietes codiert. Üblicherweise haben diese Gebiete eine Grösse von 2x2 Pixeln.

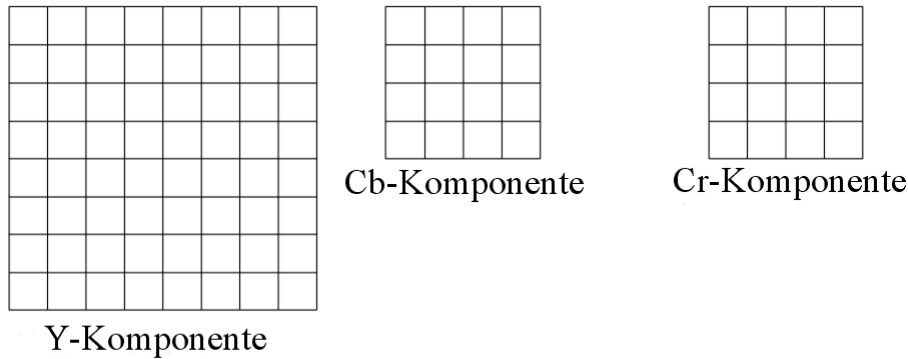


Fig. 12 Farbsubsampling von Cb, Cr

### 2.3.3 Diskrete Cosinus Transformation (DCT)

Das Bild wird in 8x8 grosse Pixelblöcke zerlegt und auf jeden Block die DCT angewendet. Falls die Dimensionen des Bildes nicht durch 8 teilbar sind, werden am linken respektive unteren Rand Werte angefügt (Bild spiegeln und periodisch fortsetzen). Der 8x8 Block wird als ein diskretes Signal mit 64 Werten aufgefasst. Diese Werte werden durch die DCT in ihr räumliches Frequenzspektrum transformiert. Die DCT ist mit der bekannten Fourier Transformation verwandt. Der Wert mit der horizontalen und der vertikalen Frequenz von 0 wird als Gleichstromkoeffizient (direct current, DC) bezeichnet, die restlichen 63 Werte als Wechselstromkoeffizienten (alternating current, AC). Im Gleichstromkoeffizient ist die meiste Information über den 8x8 grossen Pixelblock enthalten. Er ist mit Abstand der grösste Wert. Die AC-Koeffizienten repräsentieren die abrupten Farbwechsel und die scharfen Linien (Kanten) im Bild. Das dies bei natürlichen Bildern eher selten ist, werden die AC-Koeffizienten immer kleiner und kleiner (nahe bei Null). Die DCT ist verlustfrei (d. h.  $IDCT(DCT(x)) = x$ ). In der Praxis gelingt dies jedoch nicht ganz. Heutige Rechenanlagen können, bedingt durch ihren physikalischen Aufbau und ihre Arbeitsweise, manche Funktionen (wie hier der Cosinus) zwar mit grosser Genauigkeit, aber doch nicht exakt ermitteln. Dadurch ergeben sich Abweichungen der praktischen ermittelten zu den mathematisch korrekten Werten. Diese Abweichungen finden sich später im Bild wieder und ergeben einen Informationsverlust im Bild. Um eine bestimmte Bildqualität garantieren zu können, hat die JPEG-Gruppe nicht den Algorithmus zum berechnen der DCT und IDCT normiert, sondern festgelegt, wie gross der Fehler der berechneten Werte sein darf.

#### Diskrete Cosinus Transformation (DCT):

$$F(u,v) = \frac{1}{4} C(u)C(v) \left[ \sum_{x=0}^7 \sum_{y=0}^7 f(x,y) \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right) \right] \quad \begin{matrix} C(u), C(v) = \frac{1}{\sqrt{2}} & \text{für } u,v = 0 \\ C(u), C(v) = 1 & \text{sonst} \end{matrix}$$

#### Inverse Diskrete Cosinus Transformation (IDCT):

$$f(x,y) = \frac{1}{4} \left[ \sum_{u=0}^7 \sum_{v=0}^7 C(u)C(v) F(u,v) \cos\left(\frac{(2x+1)u\pi}{16}\right) \cos\left(\frac{(2y+1)v\pi}{16}\right) \right] \quad \begin{matrix} C(u), C(v) = \frac{1}{\sqrt{2}} & \text{für } u,v = 0 \\ C(u), C(v) = 1 & \text{sonst} \end{matrix}$$

### 2.3.4 Quantisierung der DCT-Koeffizienten

Die Quantisierung des DC-Koeffizienten und der 63 AC-Koeffizienten geschieht mittels Dividierung durch einen Wert aus der Quantisierungsmatrix (Tabelle). Der Wert  $F(u, v)$  wird durch den Wert  $Q(u, v)$  der Quantisierungstabelle dividiert. Das Ergebnis wird auf die nächste ganze Zahl gerundet:

$$F_Q(u, v) = \text{Integer round} \left( \frac{F(u, v)}{Q(u, v)} \right)$$

Die Quantisierungstabelle benutzt eine feinere Quantisierung für die Koeffizienten der niedrigen Frequenzen und gröbere Quantisierung für die Koeffizienten höheren Frequenzen. Deshalb werden die Koeffizienten der höheren Frequenzen (meist nahe bei Null) zu Null quantisiert. Das Einstellen des Kompressionsfaktors geschieht mittels eines Qualitätsfaktors  $q$  der von 1 bis 100 gewählt werden kann. Die dann tatsächlich verwendete Quantisierungsmatrix ergibt sich aus  $q$  nach den folgenden Beziehungen:

$$q = 100 \Rightarrow Q(u, v) = \begin{bmatrix} 1 & \dots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \dots & 1 \end{bmatrix}$$

$$q \geq 50 \Rightarrow Q(u, v) = \text{round} \left( Q(u, v)_{orig} \left( \frac{100 - q}{50} \right) \right)$$

$$q < 50 \Rightarrow Q(u, v) = \text{round} \left( Q(u, v)_{orig} \frac{50}{q} \right)$$

$Q(u, v)$  ist die tatsächlich verwendete Quantisierungsmatrix und  $Q(u, v)_{orig}$  die Matrize aus **Fig. 14** oder **Fig. 13**. Die Umkehrung der Quantisierung ist eine einfache Multiplikation des Wertes mit dem entsprechenden Eintrag in der Quantisierungstabelle:

$$F'(u, v) = F_Q(u, v) \cdot Q(u, v)$$

Hierbei wird deutlich, dass der Originalwert nicht wieder hergestellt werden kann. Die Quantisierung ist der verlustbehaftete Teil des Verfahrens und bietet die Grundlage zur effizienten Codierung. Die Quantisierungstabellen sind frei wählbar. Das JPEG-Komitee schlägt zwei Standard-Quantisierungs-Tabellen vor:

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

**Fig. 13 JPEG Standard-Quantisierungstabelle für die Y-Komponente**

$$\begin{pmatrix} 17 & 18 & 24 & 47 & 99 & 99 & 99 & 99 \\ 18 & 21 & 26 & 66 & 99 & 99 & 99 & 99 \\ 24 & 26 & 56 & 99 & 99 & 99 & 99 & 99 \\ 47 & 66 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \\ 99 & 99 & 99 & 99 & 99 & 99 & 99 & 99 \end{pmatrix}$$

**Fig. 14 JPEG Standard-Quantisierungstabelle für die Cb, Cr-Komponenten**

### 2.3.5 Codierung der Koeffizienten

Vor der Codierung werden die Werte des 8x8 Pixelblocks, durch ein Zic-Zac-Verfahren, in einen Wertestrom überführt:

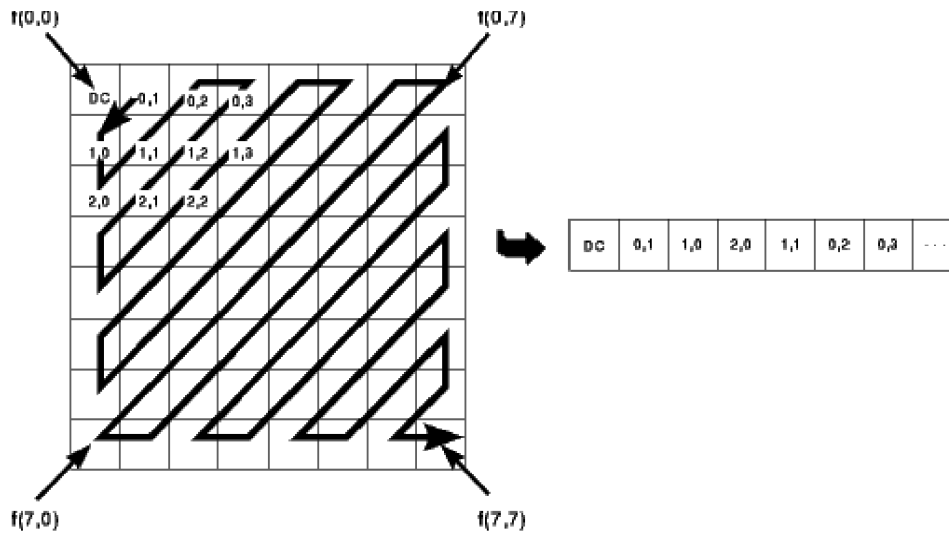


Fig. 15 Zic-Zac Verfahren

Der sequentielle Wertestrom beginnt mit dem DC-Koeffizient und wird fortgesetzt mit den AC-Koeffizienten, wodurch eine Sortierung zur höheren Ortsfrequenz entsteht. Da gerade die hohen Frequenzanteile oft sehr klein sind (beziehungsweise Null), entsteht für die weitere Kompression der Bilddaten eine günstige Reihenfolge. Nach dem Zic-Zac-Verfahren wird zuerst die Lauflängen Codierung und dann die Entropie Codierung durchgeführt. Grundsätzlich werden von JPEG zwei Entropie Codierungen unterstützt:

- die Huffman Codierung
- die arithmetische Codierung

Beim Baseline JPEG benutzt man als Entropie Codierung nur die Huffman Codierung. Bei dieser Art von Codierung wird für die Zuordnung des Codes zu den Symbolen keine externe Tabelle benötigt, für die spätere Dekomprimierung, sofern man die Standard Huffman Tabelle verwendet. Dies bewirkt wieder eine Datenreduktion. Die Lauflängen Codierung nützt es aus, dass die Sequenz der AC-Koeffizienten eine grosse Anzahl von Werten beinhaltet, die gleich Null sind. Jeder AC-Koeffizient, der ungleich Null ist, wird mit zwei Symbolen dargestellt, die folgendermassen aufgebaut sind:

$$\begin{matrix} \text{Symbol 1} & \text{Symbol 2} \\ (\text{Lauflänge, Grösse}) & (\text{Amplitude}) \end{matrix}$$

Wobei das Symbol 2 (Amplitude) den Wert des AC-Koeffizienten wiedergibt. Das Symbol 1 beschreibt, wie viele AC-Koeffizienten mit dem Wert Null von dem aktuellen AC-Koeffizienten stehen (Lauflänge). Wie viele Bits zur Codierung des Amplituden Werts benötigt werden, steht wieder im Symbol 1 (Grösse). Der Zusammenhang zwischen der Amplitude und der Grösse ist in der folgenden Tabelle erklärt:

Grösse	Amplitude
1	-1,1
2	-3,-2,2,3
3	-7,-4,4,7
4	-15,-8,8,15
5	-31,-16,16,31
6	-63,-32,32,63
7	-127,-64,64,127
8	-255,-128,128,255
9	-511,-256,256,511
10	-1023,-512,512,1023
.	.
15	-32767,-16384,16483,32767

Fig. 16 Wertebereich der VLI's

Die Lauflänge des Symbols 1 repräsentiert die Anzahl aufeinander folgenden Nullen, welche höchstens 15 betragen kann. Da die Lauflänge aber grösser als 15 werden kann, wird Symbol 1 mit dem Wert (15,0) als 16 interpretiert. Es kann maximal 3 mal der Ausdruck (15,0) nacheinander erscheinen. Folgt darauf kein von Null verschiedener AC Koeffizient mehr, wird das Block-Endzeichen (EOB), mit dem Wert (0,0) gesetzt. Es markiert das Ende des 8x8 Pixelblocks. Die Darstellungsweise der DC Werte, nach der DPCM (siehe 1.2) ist ähnlich:

*Symbol 1    Symbol 2*  
*(Grösse)    (Amplitude)*

Das Symbol 1 repräsentiert nur die Information über die Grösse und das Symbol 2 repräsentiert die Information über die Amplitude. Das Symbol 1 vom DC- und den AC-Koeffizienten wird codiert mit dem variable length code (VLC) von der Huffmanntabelle. Jedes Symbol 2 wird mit dem variable length integer (VLI Codierungstabelle) codiert. VLC ist ein Huffman Code, wobei VLI kein Huffman Code ist.

### 2.3.6 Beispiel des Baseline JPEG

Bei diesem Beispiel sieht man die Codierung und Decodierung eines 8x8 Pixelblocks der Komponente Y (Luminanz). Es werden die Standard-Quantisierungsmatrizen verwendet (ohne *q*).

$$\begin{pmatrix} 139 & 144 & 149 & 153 & 155 & 155 & 155 & 155 \\ 144 & 151 & 153 & 156 & 159 & 156 & 156 & 156 \\ 150 & 155 & 160 & 163 & 158 & 156 & 156 & 156 \\ 159 & 161 & 162 & 160 & 160 & 159 & 159 & 159 \\ 159 & 160 & 161 & 162 & 162 & 155 & 155 & 155 \\ 161 & 161 & 161 & 161 & 160 & 157 & 157 & 157 \\ 162 & 162 & 161 & 163 & 162 & 157 & 157 & 157 \\ 162 & 162 & 161 & 161 & 163 & 158 & 158 & 158 \end{pmatrix}$$

Fig. 17 8x8 Pixelblock des Originalbild

$$\begin{pmatrix} 235.6 & -1.0 & -12.1 & -5.2 & 2.1 & -1.7 & -2.7 & 1.3 \\ -22.6 & -17.5 & -6.2 & -3.2 & -2.9 & -0.1 & 0.4 & -1.2 \\ -10.9 & -9.3 & -1.6 & 1.5 & 0.2 & -0.9 & -0.6 & -0.1 \\ -7.1 & -1.9 & 0.2 & 1.5 & 0.9 & -0.1 & 0.0 & 0.3 \\ -0.6 & -0.8 & 1.5 & 1.6 & -0.1 & -0.7 & 0.6 & 1.3 \\ 1.8 & -0.2 & 1.6 & -0.3 & -0.8 & 1.5 & 1.0 & -1.0 \\ -1.3 & -0.4 & -0.3 & -1.5 & -0.5 & 1.7 & 1.1 & -0.8 \\ -2.6 & 1.6 & -3.8 & -1.8 & 1.9 & 1.2 & -0.6 & -0.4 \end{pmatrix}$$

Fig. 18 8x8 Pixelblock nach der DCT

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

Fig. 19 Quantisierungstabelle

$$\begin{pmatrix} 15 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -2 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & -1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 20 Quantisierter 8x8 Pixelblock

Nun erfolgt die Lauflängen Codierung für den quantisierten 8x8 Pixelblock. Es wird angenommen, dass der DC-Koeffizient des vorherigen Blocks 12 beträgt. Die Symbolsequenz (beginnend mit dem DC Koeffizient) sieht wie folgt aus:

(2)(3), (1,2)(-2), (0,1)(-1), (0,1)(-1), (0,1)(-1), (2,1)(-1), (15,0), (15,0), (15,0), (0,0)

Nun wird die Symbolsequenz mit der Huffmann Tabelle codiert. Der DC VLC in diesem Beispiel ist:

(2) 011

Die AC VLC's in diesem Beispiel sind:

(0,0) 1010  
 (0,1) 00  
 (1,2) 11011  
 (2,1) 11100  
 (15,0) 11111111001

Die VLI's sind:

(3) 11  
 (-2) 01  
 (-1) 0

Es resultiert ein Bitstrom mit 64 Bit, für 64 Koeffizienten, der folgendermassen aussieht:

0111111011010000000011100011111111001111111100111111110011010

Als nächstes wird der Bitstrom wieder decodiert und man erhält wieder den quantisierten 8x8 Pixelblock. Nun dequantisiert man ihn:

$$\begin{pmatrix} 240 & 0 & -10 & 0 & 0 & 0 & 0 & 0 \\ -24 & -12 & 0 & 0 & 0 & 0 & 0 & 0 \\ -14 & -13 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Fig. 21 Dequantisierter 8x8 Pixelblock

$$\begin{pmatrix} 144 & 146 & 149 & 152 & 154 & 156 & 156 & 156 \\ 148 & 150 & 152 & 154 & 156 & 156 & 156 & 156 \\ 155 & 156 & 157 & 158 & 158 & 157 & 156 & 155 \\ 160 & 161 & 161 & 162 & 161 & 159 & 157 & 155 \\ 163 & 163 & 164 & 163 & 162 & 160 & 158 & 156 \\ 163 & 164 & 164 & 164 & 162 & 160 & 158 & 157 \\ 160 & 161 & 162 & 162 & 162 & 161 & 159 & 158 \\ 158 & 159 & 161 & 161 & 162 & 161 & 159 & 158 \end{pmatrix}$$

Fig. 22 Rekonstruierter 8x8 Pixelblock

## 3 JPEG2000

### 3.1 Einleitung

JPEG2000 ist ein neues Komprimierungsverfahren für unbewegte Bilder. JPEG2000 befindet sich gegenwärtig in der Endentwicklung, 2001 wird es dann zum Standard. JPEG2000 ist nicht als Ersatz für JPEG gedacht, sondern als Ergänzung.

### 3.2 Warum ein neuer Standard?

- Kompression mit kleinen Bitraten: z.B. unter 0.25 bpp (Bits per Pixel)
- Kompression von computergenerierten Bildern:  
JPEG war für natürliche Bilder optimiert und hatte Probleme bei computergenerierten Bildern ("verpixelung" sogenanntes „ringing“ in der Umgebung von scharfen Kanten).
- Übertragung über gestörte Kanäle: die Bildqualität von JPEG litt sehr unter bit errors (Bit Fehler).

### 3.3 Features von JPEG2000

- Hohe Kompressionseffizienz
- Verlustfreie Farbtransformation
- Lossy und lossless Codierung in einem Algorithmus (verlustbehaftet, verlustlos)
- Eingebettete lossy to lossless Codierung
- Progressive in Qualität und Auflösung
- Progressive in Auflösung: Bild wird immer grösser bis es seine maximale Grösse erreicht hat.
- Progressive in Qualität: Bild wird immer schärfer bis es seine maximale Schärfe erreicht hat.
- Region-of-Interest: Erlaubt Bereiche des Bild schonender zu komprimieren (bessere Qualität)
- Error resilience: "Widerstand" gegen bit errors
- Das Dateiformat wird so gestaltet sein, dass es dem Autor möglich ist meta-daten unterzubringen.

### 3.4 Hauptunterschiede zwischen JPEG und JPEG2000

- Neue Funktionen
  - RIO (Region-of-Interest)
  - Error resilience
- Bessere Kompression bei tiefen Bitraten
- Bessere Qualität bei Bildern die Text und Grafik enthalten (bei gleichbleibender Kompression)

### 3.5 JPEG2000 der Algorithmus

JPEG2000 ist ein transformations- basierter Codierer. Das zu Codierende Bild wird zuerst in einem geeigneten Farbraum YCbCr dargestellt, dann transformiert, quantisiert und anschliessend entropiecodiert. Im Decoder werden diese Schritte rückwärts durchlaufen. JPEG2000 verwendet als Transformation die diskrete schnelle Wavelettransformation.

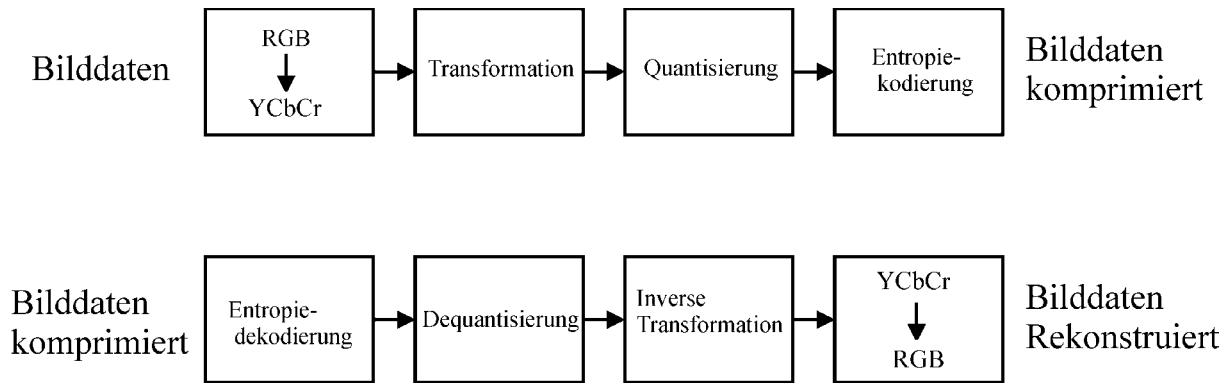


Fig. 23 Schema eines Transformations- basierten Codierers

### 3.5.1 Der YCbCr Farbraum

Siehe Kapitel 2.3.1

Die Y-Komponente, Cb-Komponente und Cr-Komponente durchlaufen den Algorithmus separat.

### 3.5.2 Die diskrete Wavelettransformation (DWT)

#### 3.5.2.1 Was sind Wavelets?

Wavelets ist Englisch und heisst übersetzt etwa „kleine Wellen“. Die Wavelets bilden eine Basis (Wie bei der Fourierzerlegung Sinus und Cosinus) . Ein Signal lässt sich also als Linearkombination darstellen. Die Koeffizienten bei der Fourierzerlegung sind die Amplituden und die Frequenzen der Basis (Sinus, Cosinus). Bei der Wavelettransformation sind die Koeffizienten die Amplituden, die Zeitverschiebung und die Dauer des Wavelets.

Zwei Beispiele für Wavelets:

$$(1-x^2) \cdot e^{-\frac{x^2}{2}}$$

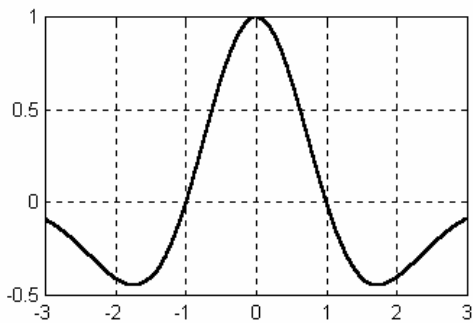


Fig. 24 Mexican-hat-wavelet

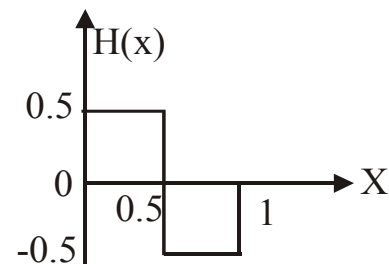


Fig. 25 Haar-wavelet

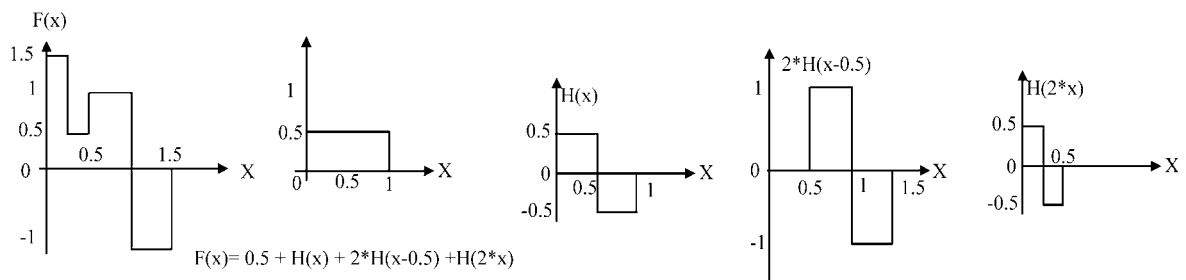


Fig. 26 Beispiel für die Zerlegung eines Signals mit dem Haar-wavelet

### 3.5.2.2 Filterbank Implementation der diskreten Wavelettransformation

Im Bereich der Digitalen Signalverarbeitung wird eine Filterung dadurch erreicht, dass man eine Eingangssignalfolge mit einer anderen Signalfolge, den sogenannten Filterkoeffizienten, faltet. Bei der kausalen linearen zeitinvarianten Signalverarbeitung unterscheidet man Systeme mit endlich langer Impulsantwort und solche mit unendlich langer Impulsantwort. Erstere heissen FIR-Systeme (Finite Impulse Response), letztere IIR-Systeme (von Infinite Impulse Response). Das Systemverhalten beider Systemtypen wird durch die allgemeine diskrete Faltungssumme beschrieben.

$$y(n) = \sum_{k=0}^{\infty} x(n-k) \cdot h(k)$$

Man kann zeigen, dass sich die Dekomposition eines Signals durch Wavelets ebenfalls auf eine besondere Faltung mit den sogenannten Filter-Koeffizienten des Wavelets (auch Entwicklungskoeffizienten genannt) reduzieren lässt. Auf die Berechnung dieser Filter-Koeffizienten soll hier nicht näher eingegangen werden. Bei der praktischen Anwendung der Wavelettransformation wird nur mit den Entwicklungskoeffizienten der Signale gerechnet. Die Skalierungsfunktionen und Wavelets selber werden nicht benötigt. Um die Merkmale der Skalierungsfunktionen und Wavelets untersuchen und ihre Existenz nachweisen zu können, geht man den umgekehrten Weg. Man konstruiert aus den Koeffizientensätzen  $h(n)$  und  $g(n)$  (Scaling bzw. Wavelet Filterkoeffizienten) heraus die Skalierungsfunktionen und Wavelets.

**Die Formel zur Berechnung der Scaling Koeffizienten  $c_j$  des Signals bei der Dekomposition lautet**

$$c_{j-1}(m) = \sum_m h(m-2k) \cdot c_j(k)$$

**und für die Wavelet Koeffizienten  $d_j$  entsprechend**

$$d_{j-1}(m) = \sum_m g(m-2k) \cdot d_j(k)$$

Diese Gleichungen zeigen, wie die Wavelet und Scaling Koeffizienten der verschiedenen Betrachtungsebenen durch Faltung der Koeffizienten der  $j$ -ten Stufe mit dem zeitinversen Filterkoeffizienten und einem anschliessenden Downsampling von 2 (d.h. nur jeder zweite Wert wird berücksichtigt) zu den Koeffizienten der  $(j-1)$ -ten Stufe führen, d.h. der nächst groberen Detailstufe. Die Filterkoeffizienten für  $h(n)$  entsprechen einem Tiefpass und für  $g(n)$  entsprechen sie einem Hochpass. Beim Haar Wavelet wird der Tiefpass aus der Folge  $h = [1/\sqrt{2} \ 1/\sqrt{2}]$  und der Hochpass aus der Folge  $g = [1/\sqrt{2} \ -1/\sqrt{2}]$  dargestellt.

Falls man diesen Prozess der Filterung mit anschliessender Dezimierung über mehreren Iterationen fortführt, erhält man die untenstehende Dekomposition.

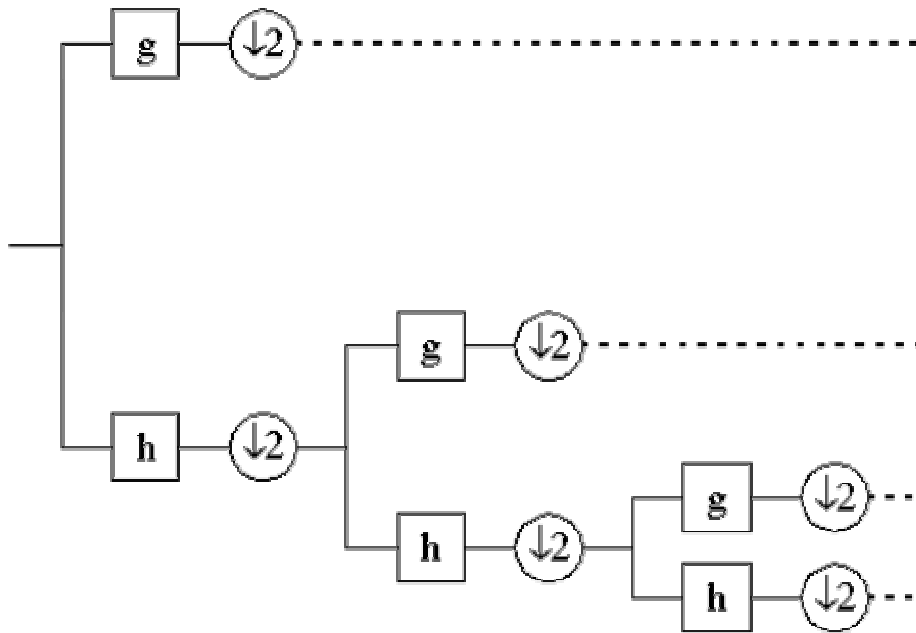


Fig. 27 Dekomposition mittels einer Filterbank

Für die Rekonstruktion muss die j-te Auflösungsstufe der Scaling und Wavelet Funktionen zurück zur nächst detailreicheren Auflösung der (j+1)-ten Stufe gebracht werden. Die Rekonstruktionsformel sieht wie folgt aus:

$$c_{j+1}(k) = \sum_m g(k-2m) \cdot d_j(m) + \sum_m h(k-2m) \cdot c_j(m)$$

Bei der Rekonstruktion werden demnach die beiden Teilsignale niedrigerer Auflösung einem upsampling von 2 unterzogen, d. h. an jeder 2. Stelle wird eine Null eingefügt und anschliessend wieder mit den Rekonstruktionsfiltern, die nicht identisch mit den Dekompositionsfiltren sein müssen, es aber in unserem Fall sind, gefiltert.

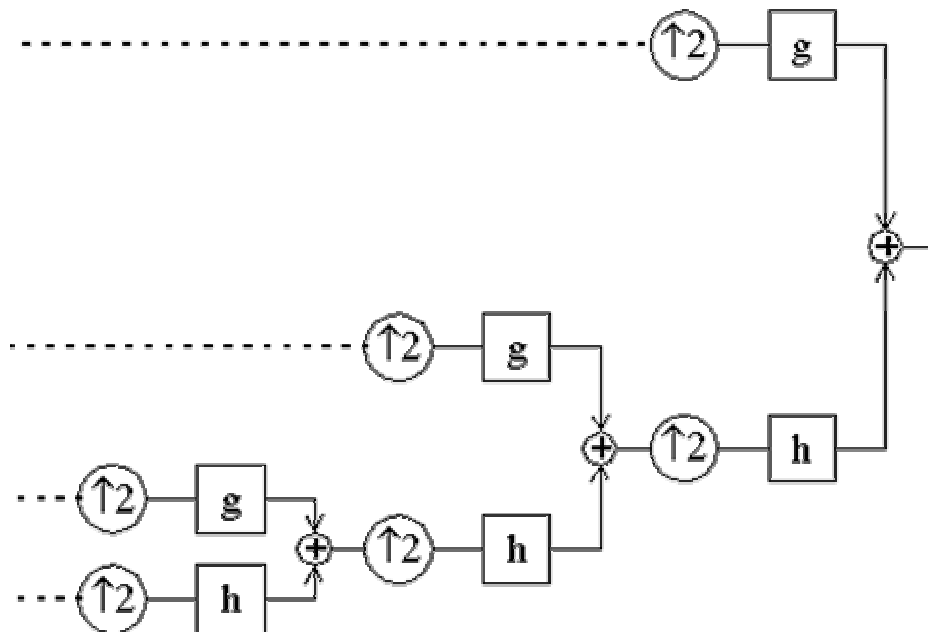


Fig. 28 Synthese des Ursprungssignals mittels einer Filterbank

Diese Filterbank Implementierung gewährleistet eine schnelle numerische Realisierung der Diskreten Wavelet Transformation. Dadurch lassen sich ähnlich wie bei der FFT schnelle Algorithmen für deren Berechnungen schreiben..

### 3.5.2.3 Die 2-Dimensionale diskrete Wavelettransformation

Die Diskrete Wavelet Transformation stellt im Falle von mehrdimensionalen Daten und orthogonalen Wavelets als Basisfunktionen eine direkte Erweiterung der Transformation eindimensionaler Daten dar. Ein N-dimensionales Problem wird auf N Teilprobleme der Dimension N-1 heruntergebrochen. Z. B. ein Bild, welches als NxN Matrix, d.h. als 2-dimensionale Datenstruktur, dargestellt wird, kann in N Zeilen zerlegt werden, bei denen man dann die eindimensionale DWT anwenden kann. Nach dieser eindimensionalen Transformation der Zeilen werden die Teildatensätze wieder zu einer Matrix zusammengesetzt. Daraufhin werden anstelle der Zeilen die Spalten einzeln betrachtet. Auf diese Spalten wird ebenfalls die eindimensionale DWT angewendet und anschliessend wieder zum Bild zusammgefügt. Die Rekonstruktion funktioniert nach dem gleichen Prinzip. Mit Hilfe der Rekonstruktionsfilter werden zuerst die Spalten und dann die Zeilen nach der eindimensionalen Rekonstruktion wieder zurücktransformiert. Das folgende Diagramm zeigt schematisch die Dekomposition, wobei L für den mit dem Tiefpass (lowpass) gefilterten Bereich steht and H entsprechend den Hochpass (high pass) gefilterten Anteil bezeichnet.

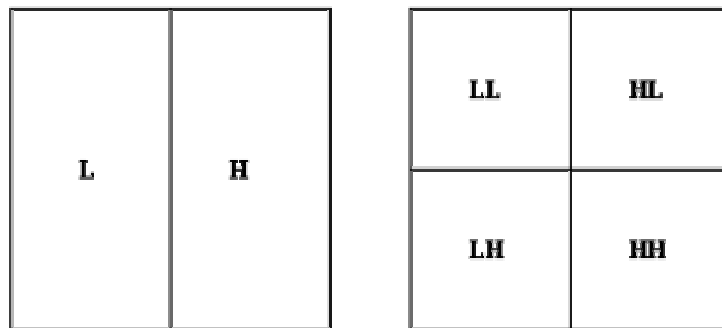


Fig. 29 Dekompositionsschritte der DWT bei 2-dimensionalen Datensätzen

Entsprechend bedeutet eine HL Filterung, bei der zuerst die Zeilen mit einem Hochpass und dann die Spalten mit einem Tiefpass gefiltert wurden. Die Unterteilung in LL, HL, LH und HH ist demnach die erste Iteration der Dekomposition eines 2-dimensionalen Datensatzes. Wie auch im eindimensionalen Fall werden die weiteren Iterationen nur am Tiefpassanteil, hier also dem LL, vollzogen. Es ist in der Literatur üblich die einzelnen gefilterten Bereiche ebenfalls mit der Iterationsnummer zu versehen. Dies wird im untenstehenden Diagramm für den Fall der abgeschlossenen 3. Iteration gezeigt.

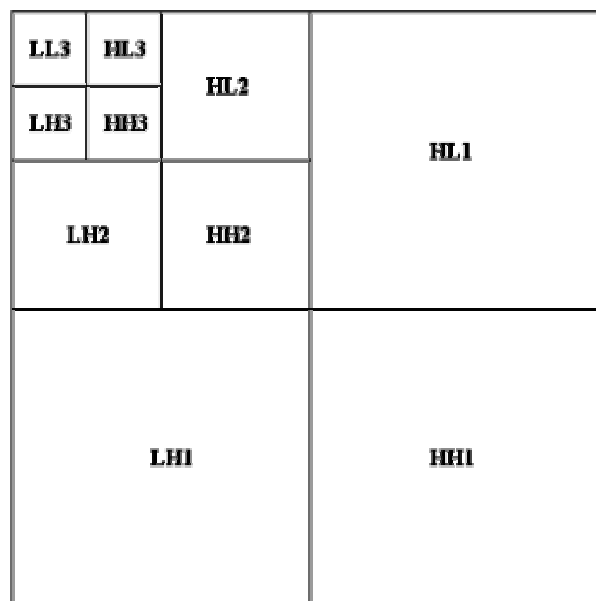


Fig. 30 Dekomposition von 2D-Daten nach der 3. Iteration

Die bisher schematischen Erklärungen sollen anhand eines Beispiels verdeutlicht werden. Im oberen Bild(Originalbild) sind Balken verschieden angeordnet. Durch die Transformation bis einschliesslich der 2. Iteration entsteht das darunter stehende Ergebnis. Die vier bei jeder Iteration erhaltenen Teilbilder können visuell folgendermassen interpretiert werden:

- LL entspricht einer geglätteten und verkleinerten Version des Originalbildes (gewissermassen eine Darstellung des Originals bei verringerter Auflösung)
- in LH treten besonders die horizontalen Bildelemente hervor
- in HL dominieren die vertikalen Bildelemente
- HH betont die diagonalen Strukturen des Original-Bildes

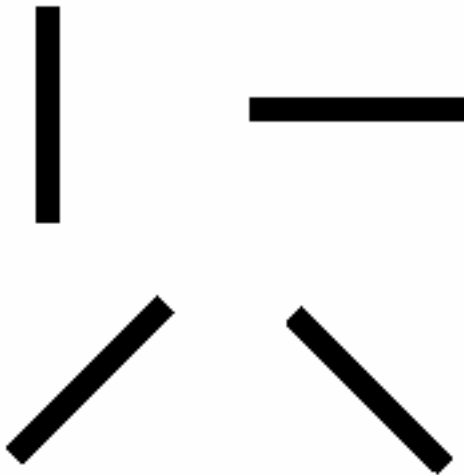


Fig. 31 Originalbild

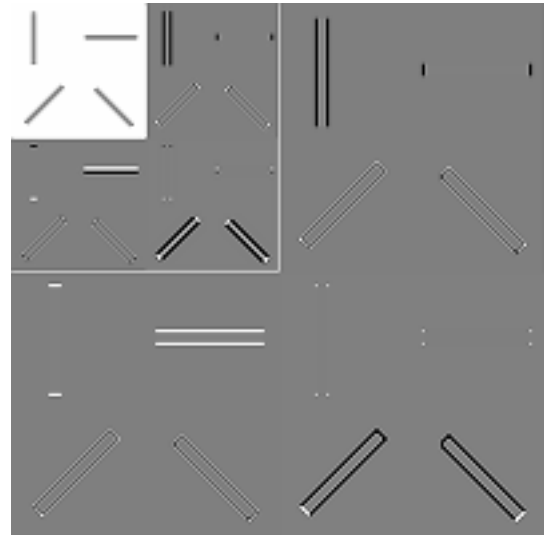


Fig. 32 Transformationsergebnis nach der 2. Iteration

### 3.5.2.4 Die Wavelettransformation die JPEG2000 verwendet

Bei JPEG2000 kann, wie bei JPEG, das Bild in Blöcke aufgeteilt werden (die Grösse der Blöcke liegt im Bereich von: 1 bis  $2^{32}-1$ ). Diese Blöcke werden dann durch den ganzen Algorithmus wie eigenständige Bilder behandelt. Das Zerlegen des Bildes in einzelne Blöcke wird im Englischen mit TILING bezeichnet.

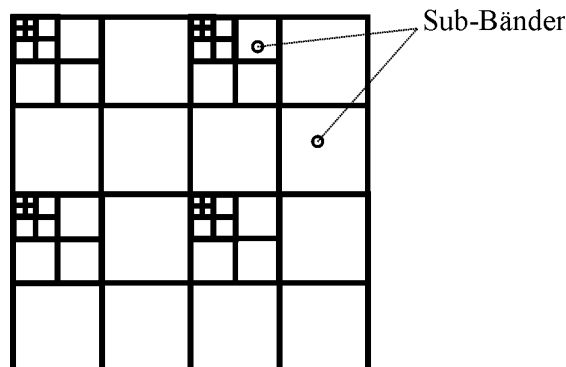


Fig. 33 Subband-dekomposition der tiles

Diese Blockbildung ist für die Wavelettransformation nicht unbedingt nötig, da die Wavelettransformation im Gegensatz zur DCT eine gute Ortsauflösung besitzt. Anders gesagt, man weiss nach der Transformation immer noch, an welcher Stelle im Bild die hohen Frequenzen auftreten. Tiling reduziert den für das ausführen der Transformation benötigten Speicher. Ausserdem bildet tiling die Grundlage für die Extraktion eines bestimmten Bildbereichs (RIO). Bevor die einzelnen tiles transformiert werden, werden sie „DC-entkoppelt“ d.h. von jedem Sample eines tiles wird der gleiche Wert subtrahiert (z.B. die Komponententiefe). Die Tatsache, dass der Analyse Hochpass einen Koeffizienten kürzer ist hat zur Folge, dass beim downsampling keine Information verloren geht. Diese „Informationserhaltung“ lässt sich dadurch erklären, dass durch die unterschiedliche Filterlänge die gefilterten Signale eine Verzögerung zueinander erfahren. Das hat wiederum zur Folge, dass beim downsampling verschiedene Werte verwendet werden.

Analyse Filterkoeffizienten		
	Tiefpass Filter $h_L(i)$	Hochpass Filter $h_H(i)$
0	0.6029490182363579	1.115087052456994
$\pm 1$	0.2668641184428723	-0.59127177631142470
$\pm 2$	-0.07822326652898785	-0.05754352622849957
$\pm 3$	-0.01686411844287495	0.09127176311424948
$\pm 4$	0.02674875741080976	

**Fig. 34 Daubechies 9/7 Analyse Filterkoeffizienten (LOSSY)**

Synthese Filterkoeffizienten		
	Tiefpass Filter $g_L(i)$	Hochpass Filter $g_H(i)$
0	1.115087052456994	0.6029490182363579
$\pm 1$	-0.59127177631142470	0.2668641184428723
$\pm 2$	-0.05754352622849957	-0.07822326652898785
$\pm 3$	0.09127176311424948	-0.01686411844287495
$\pm 4$		0.02674875741080976

**Fig. 35 Daubechies 9/7 Synthese Filterkoeffizienten (LOSSY)**

Analyse Filterkoeffizienten		
	Tiefpass Filter $h_L(i)$	Hochpass Filter $h_H(i)$
0	6/8	1
$\pm 1$	2/8	-1/2
$\pm 2$	-1/8	

**Fig. 36 5/3 Analyse und Filterkoeffizienten (LOSSLESS)**

Analyse Filterkoeffizienten		
	Tiefpass Filter $h_L(i)$	Hochpass Filter $h_H(i)$
0	1	6/8
$\pm 1$	-1/2	2/8
$\pm 2$		-1/8

**Fig. 37 Daubechies 9/7 Synthese Filterkoeffizienten (LOSSY)**

### 3.5.3 Quantisierung

Die Quantisierung ist grundsätzlich verlustbehaftet. Falls aber das 5/3 Filter benutzt wird und mit 1 quantisiert wird ist die Quantisierung verlustlos.

$$q(u, v) = \text{sign}(a(u, v)) \left\lfloor \frac{|a(u, v)|}{\Delta b} \right\rfloor$$

Wobei  $a(u, v)$  einen Waveletkoeffizienten darstellt und  $q(u, v)$  einen quantisierten Waveletkoeffizienten darstellt. Mit  $\Delta b$  kann man die Quantisierung einstellen. Jedes Subband wird mit einem separaten  $\Delta b$  quantisiert.  $\Delta b$  ist Gegeben durch:

$$\Delta b = 2^{R_b - \varepsilon_b} \left( 1 + \frac{\mu_b}{2^{11}} \right)$$

Mit dieser Mantissen-Exponent-Darstellung kann  $\Delta b$  über einen sehr grossen Bereich mit guter Genauigkeit und wenig Bits eingestellt werden.  $R_b$  ist abhängig von der Bittiefe des Originalbildes. Mit  $\varepsilon_b$  und  $\mu_b$  kann der quantisierungs-Schritt  $\Delta b$  eingestellt werden.

### 3.5.4 Entropiecodierung

Jedes Subband der Wavelettransformation wird in sogenannte Codeblöcke unterteilt, die dann separat arithmetisch codiert werden. Arithmetische Codierung siehe 1.3

### 3.5.5 Erläuterungen und Beispiele

#### 3.5.5.1 Region-of-Interest (RIO)

Die RIO-Funktion erlaubt einen Bereich im Bild schon beim Bildaufbau scharf darzustellen. Das funktioniert etwa so: Die Daten, die den RIO-Bereich darstellen, werden am Anfang des Datenstroms eingefügt, dadurch wird beim Bildaufbau zuerst dieser Bereich scharf dargestellt.



Fig. 38 Region Of Interest

#### 3.5.5.2 Error resilience

Komprimierte Daten sind grundsätzlich sehr fehleranfällig (sehr wenig Redundanz enthalten). JPEG2000 wird eine gewisse „Abwehr“ gegen Bitfehler enthalten. Erreicht wird dies mit Fehlererkennung und Resynchronisation im Bitstrom.

#### 3.5.5.3 Metadaten

In JPEG2000 wird es möglich sein Informationen einzubetten, wie Autor, Copyright oder sonstiges.

# Implementierung

## 4 JPEG

### 4.1 Allgemeines

In diesem Kapitel wird auf die Implementierung des JPEG Algorithmus eingegangen. Auf eine detaillierte Erklärung der Programm-Source wird verzichtet. Die Programm-Source befindet sich im Anhang A `JPEG SOURCECODE`.

### 4.2 Vorgehen und Stolpersteine

Es wird nur das grobe Vorgehen der Implementierung und dessen Stolpersteine beschrieben.

Das grundlegendste Problem war das Umdenken vom eindimensionalen in den zweidimensionalen Raum. Da wir uns täglich mit eindimensionalen Signalen auseinandersetzen, war es für uns nicht gerade geläufig, im zweidimensionalen Raum zu denken. Dieses Problem hatten wir aber schnell im Griff und konnten nun mit der Implementierung des JPEG Algorithmus beginnen.

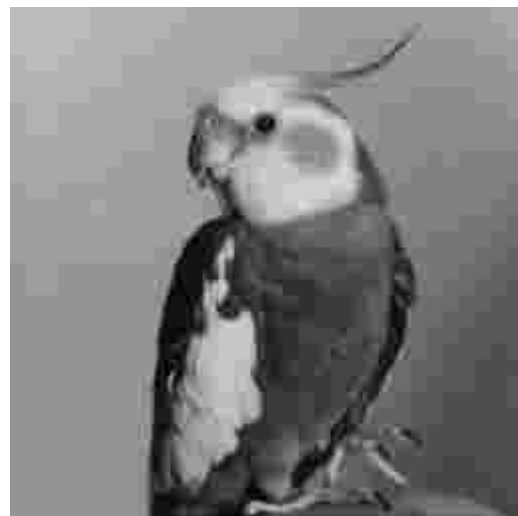
Das Herz des JPEG Algorithmus ist die Diskrete Cosinus Transformation und deshalb wollten wir zuerst diese implementieren. Schon recht bald hatten wir eine funktionsfähige DCT implementiert, die aber zu unserem Erstaunen sehr langsam war. Wir realisierten die DCT mit for-Schleifen, was man unbedingt vermeiden sollte, wenn man mit Matlab arbeitet. Der DS-Assistent schlug uns dann vor, die DCT mittels Matrizenmultiplikation zu implementieren. Nach diesem Gespräch liessen wir von den for-Schleifen ab und konzentrierten uns auf die Matrizenrechnung. Dies war ungewöhnlich für uns, da wir von C++ gewohnt waren mit Schleifen zu arbeiten. Es wurde uns bewusst, dass die Geschwindigkeit ein wesentliches Problem der Implementierung werden würde. Vor allem bei den Funktionen `farbsubsampling`, `farbupsampling`, `zig-zac-encoding` und `runlength-encoding` wurden wir vor dieses Problem gestellt. Durch Anwenden der Matrizenrechnung konnten wir aber die Geschwindigkeit wieder etwas steigern.

Nach einiger Zeit war der erste Prototyp des JPEG Algorithmus, der die wesentlichsten Funktionen beinhaltet, fertiggestellt. Durch Vergleiche mit dem Bildverarbeitungsprogramm ACDSee sah man Unterschiede vom komprimierten Bild unseres JPEG Algorithmus, zu demjenigen des ACDSee. Der Unterschied lag darin, dass sich die „Verpixelungen“ örtlich unterschieden.

Durch intensive Fehlersuche und zusätzlichem Literaturstudium fanden wir den Fehler. Der Fehler bestand darin, dass wir vor der FDCT den DC-level-shift vergessen hatten (natürlich auch inverser-DC-level-shift nach der IDCT). Nach dem Korrigieren dieses Fehlers sah man nur noch vernachlässigbare Unterschiede ( bei Graubildern).



**Fig. 39** Komprimiert durch den JPEG Algorithmus ohne DC-level-shift



**Fig. 40** Komprimiert durch ACDSee



**Fig. 41** Komprimiert durch den JPEG Algorithmus mit DC-level-shift



**Fig. 42** Komprimiert durch ACDSee

Bei Farbbildern mit tiefem Qualitätsfaktor besteht immer noch ein Unterschied. Es könnte sein, dass ACDSee leicht anders implementiert ist.

Jetzt hatten wir einen Prototypen, der die Grundfunktionen beinhaltet. Nun ging es darum, die Einzelschritte des Algorithmus graphisch darzustellen und um die Berechnung des PSNR und der Entropien. Wie schon vorher beschrieben, war die Geschwindigkeit ein grosses Problem, wie auch das Interpretieren der erhaltenen Resultate. Zu Vergleichszwecken berechneten wir noch die Entropie des rekonstruierten Bildes. Zuerst glaubten wir, dass bei der Entropierechnung etwas schief gegangen sei, weil die Entropie des rekonstruierten Bildes manchmal höher war als diejenige des Originalbildes. Doch dem war nicht so. Man stelle sich ein Bild vor, das einen schwarzen Hintergrund hat und darüber eine weisse Linie. Mit kleiner werdendem Qualitätsfaktor werden die hohen Frequenzen immer mehr zu null quantisiert. Dadurch entsteht an scharfen Farbübergängen ein Überschwingen (Gibb's-Phänomen), was man auch „ringing“ nennt. Durch dieses „ringing“ wird die Entropie des rekonstruierten Bildes grösser als diejenige des Originalbildes.

Ein weiteres Problem war die Programmierung der Entropieberechnung nach der Runlength-Codierung (siehe 2.3.5). Da das Symbol 1 keine Zahl ist, mussten wir die Runlength-Codierung approximieren (Matlab kann nur Histogramme von Zahlenwerten berechnen). Anstatt der in der Theorie erklärten Runlength-Codierung mit den Symbolen 1 und 2, verwendeten wir eine abgekürzte Form. Wir liessen nämlich die VLI's (Grössen) des Symbols 1 weg. Übrig blieb das Symbol 1 mit der Lauflänge und Symbol 2 mit dem Amplitudenwert. Nun konnten wir wieder, unter Einhaltung der restlichen Regeln, die Entropie berechnen.

Wir hatten den JPEG Algorithmus implementiert, der die einzelnen Schritte graphisch darstellt und die Entropien berechnet.

### 4.3 Funktionsbeschreibung

Es werden die implementierten und die von der IMAGE PROCESSING TOOLBOX zur Verfügung gestellten Funktionen der Reihe nach, wie sie im JPEG Algorithmus gebraucht werden, beschrieben (Programm-Source befindet sich im Anhang A JPEG SOURCECODE).

#### **bild = jpeg(imm,q,x,runl)**

Übergabeparameter: **imm** entspricht dem Bild, das eingelesen werden soll, in „hochkomma“  
**q** entspricht dem Qualitätsfaktor  
**x** entspricht dem Wahlparameter zur graphischen Darstellung der einzelnen Schritte.  
**runl** entspricht dem Wahlparameter, ob man die Entropie nach der Runlength-Codierung berechnen soll oder nicht.

Rückgabeparameter: **bild** entspricht dem rekonstruierten Bild

Beschreibung: Ist das Hauptprogramm.

#### **y = imread(x)** (Funktion der Image Processing Toolbox)

Übergabeparameter: **x** entspricht dem Bild, das eingelesen werden soll, in „hochkomma“

Rückgabeparameter: **y** entspricht einem NxMx3 Array bei Farbbildern und einer NxM Matrize bei Graubildern.

Beschreibung: Liest ein Bild ein.

#### **imshow(x)** (Funktion der Image Processing Toolbox)

Übergabeparameter: **x** entspricht einer NxM Matrize (Graubild) vom Typ uint8, oder einem NxMx3 Array (Farbbild) vom Typ uint8.

Beschreibung: Öffnet Ausgabefenster und stellt Bild dar (unkaliert).

#### **y = rgb2ycbcr(x)** (Funktion der Image Processing Toolbox)

Übergabeparameter: **x** entspricht dem Bild, das umgewandelt werden soll.

Rückgabeparameter: **y** entspricht einem Bild, das in den YCbCr-Farbraum transformiert wurde.

Beschreibung: Transformiert ein Bild im RGB-Farbraum vom Typ uint8 in den YCbCr-Farbraum.

#### **y = erweitern\_8(x)**

Übergabeparameter: **x** entspricht einer NxM Matrize.

Rückgabeparameter: **y** entspricht dem periodisch fortgesetzten **x**.

Beschreibung: **x** wird periodisch fortgesetzt (spiegeln), so das N und M durch 8 teilbar sind (für Graubilder).

**$y = \text{erweitern\_16}(x)$**

Übergabeparameter:  $x$  entspricht einem  $N \times M \times 3$ -Array.

Rückgabeparameter:  $y$  entspricht dem periodisch fortgesetzten  $x$ .

Beschreibung:  $x$  wird periodisch fortgesetzt (spiegeln), so dass  $N$  und  $M$  durch 16 teilbar sind (für Farbbilder).

**$y = \text{farbsubsampling}(x)$**

Übergabeparameter:  $x$  entspricht einer  $N \times M$  Matrize der Cb- oder der Cr-Komponente.

Rückgabeparameter:  $y$  entspricht einer  $N \times M$  Matrize.

Beschreibung: Fasst jeweils vier Werte von  $x$  zusammen und mittelt sie, so dass  $y$  halb so gross wie  $x$  ist.

**$[y,z] = \text{mydctY}(x,q)$**

Übergabeparameter:  $x$  entspricht der Y-Komponente eines Bildes.  
 $q$  entspricht dem Qualitätsfaktor.

Rückgabeparameter:  $y$  entspricht dem transformierten und quantisierten  $x$  (im Frequenzbereich).  
 $z$  entspricht einem Vektor.

Beschreibung: Wendet auf jeden  $8 \times 8$  Block die DCT an und quantisiert ihn anschliessend.  
 $z$  wird nach dem zic-zac-Verfahren aus  $y$  gebildet.

**$y = \text{quantisierungY}(x,q)$**

Übergabeparameter:  $x$  entspricht dem transformierten  $8 \times 8$  Block der Y-Komponente eines Bildes.  
 $q$  entspricht dem Qualitätsfaktor.

Rückgabeparameter:  $y$  entspricht dem quantisierten  $x$ .

Beschreibung: Quantisiert jeden transformierten  $8 \times 8$  Block.  
**quantisierungY()** wird in der Funktion **mydctY()** aufgerufen.

**$y = \text{zigzag}(x)$**

Übergabeparameter:  $x$  entspricht dem transformierten  $8 \times 8$  Block der Y-, Cr- oder Cb-Komponente eines Bildes.

Rückgabeparameter:  $y$  entspricht einem Vektor.

Beschreibung: Ordnet Matrize  $x$  nach dem zic-zac-Verfahren in  $y$  ein.

**[y,z] = mydctCbCr(x,q)**

- Übergabeparameter: **x** entspricht der Cb- oder der Cr-Komponente eines Bildes.  
**q** entspricht dem Qualitätsfaktor.
- Rückgabeparameter: **y** entspricht dem transformierten und quantisierten **x** (im Frequenzbereich).  
**z** entspricht einem Vektor.
- Beschreibung: Wendet auf jeden 8x8 Block die DCT an und quantisiert ihn anschliessend.  
**z** wird nach dem zic-zac-Verfahren aus **y** gebildet.

**y = quantisierungCbCr(x,q)**

- Übergabeparameter: **x** entspricht dem transformierten 8x8 Block der Cb- oder der Cr-Komponente eines Bildes.  
**q** entspricht dem Qualitätsfaktor.
- Rückgabeparameter: **y** entspricht dem quantisierten **x**.
- Beschreibung: Quantisiert jeden transformierten 8x8 Block von **x**.  
**quantisierungCbCr()** wird in der Funktion **mydctCbCr()** aufgerufen.

**y = myidctY(x,q)**

- Übergabeparameter: **x** entspricht der transformierten Y-Komponente (im Frequenzbereich) eines Bildes.  
**q** entspricht dem Qualitätsfaktor.
- Rückgabeparameter: **y** entspricht der rücktransformierten **x**.
- Beschreibung: Dequantisiert jeden 8x8 Block von **x** und wendet die IDCT auf ihn an.

**y = dequantisierungY(x,q)**

- Übergabeparameter: **x** entspricht der transformierten Y-Komponente (im Frequenzbereich) eines Bildes.  
**q** entspricht dem Qualitätsfaktor.
- Rückgabeparameter: **y** entspricht dem dequantisierten **x**.
- Beschreibung: dequantisiert jeden 8x8 Block von **x**.  
**dequantisierungY()** wird in der Funktion **myidctY()** aufgerufen.

**y = myidctCbCr(x,q)**

- Übergabeparameter: **x** entspricht der transformierten Cb- oder Cr-Komponente (im Frequenzbereich) eines Bildes.  
**q** entspricht dem Qualitätsfaktor.
- Rückgabeparameter: **y** entspricht der rücktransformierten **x**.
- Beschreibung: Dequantisiert jeden 8x8 Block von **x** und wendet die IDCT auf ihn an.

**y = dequantisierungCbCr(x,q)**

- Übergabeparameter: **x** entspricht der transformierten Cb- oder Cr-Komponente (im Frequenzbereich) eines Bildes.  
**q** entspricht dem Qualitätsfaktor.
- Rückgabeparameter: **y** entspricht dem dequantisierten **x**.
- Beschreibung: dequantisiert jeden 8x8 Block von **x**.  
**dequantisierungCbCr()** wird in der Funktion `myidctCbCr()` aufgerufen.

**y = farbupsampling(x)**

- Übergabeparameter: **x** entspricht einer NxM Matrize der Cb- oder der Cr-Komponente.
- Rückgabeparameter: **y** entspricht einer NxM Matrize der Cb- oder der Cr-Komponente.
- Beschreibung: Der Wert eines Bildpunktes von **x** wird in vier Bildpunkte von **y** geschrieben, so dass **y** doppelt so gross wie **x** ist.

**y = schneiden(x,d)**

- Übergabeparameter: **x** entspricht einer NxM Matrize der Y-, Cb- oder der Cr-Komponente.  
**d** entspricht der Dimension des Originalbildes (Vektor der Länge 2).
- Rückgabeparameter: **y** entspricht einer NxM Matrize der Y-, Cb- oder der Cr-Komponente.
- Beschreibung: **x** wird auf die Dimension des Originalbildes zugeschnitten

**y = ycbcr2rgb(x)**

**(Funktion der Image Processing Toolbox)**

- Übergabeparameter: **x** entspricht dem Bild, das umgewandelt werden soll.
- Rückgabeparameter: **y** entspricht einem Bild, das in den RGB-Farbraum transformiert wurde.
- Beschreibung: Transformiert ein Bild im YCbCr -Farbraum, vom Typ uint8, in den RGB-Farbraum.

**z = berechne\_psnr(x,y)**

- Übergabeparameter: **x** entspricht dem Originalbild im YcbCr-Farbraum.  
**y** entspricht dem rekonstruierten Bild im YCbCr-Farbraum.
- Rückgabeparameter: **z** entspricht dem peak signal noise ratio.
- Beschreibung: Berechnet das peak signal noise ratio.

**y = berechne\_entropie(x)**

Übergabeparameter: **x** entspricht einer MxN Matrize des Originalbild im YCbCr-Farbraum.

Rückgabeparameter: **y** entspricht der Entropie.

Beschreibung: Berechnet die Entropie, der Y-, Cb- oder Cr-Komponenten, des Originalbildes.

**y = runlength(x)**

Übergabeparameter: **x** entspricht einem Vektor, der nach dem ziczac-Verfahren angeordnet wurde.

Rückgabeparameter: **y** entspricht dem Runlength-Codierten Vektor.

Beschreibung: Berechnet die Runlength-Codierung

**y = berechne\_run\_entropie(z)**

Übergabeparameter: **z** entspricht dem Runlength-Codierten Vektor.

Rückgabeparameter: **y** entspricht der Entropie des Runlength-Codierten Vektors.

Beschreibung: Berechnet die Entropie des Runlength-Codierten Vektors.

## 4.4 Erkenntnisse

Im Laufe dieser Arbeit wurde uns immer bewusster, wie clever der JPEG-Algorithmus ist. Er modelliert recht genau das menschliche Sehverhalten nach und ist ausserdem eine gelungene Kombination von Transformation und Codierung. Nachdem wir uns mit räumlichen Frequenzen und mit dem zweidimensionalen Denken angefreundet hatten, fanden wir diesen Algorithmus relativ einfach verständlich.

Matlab präsentierte sich uns als das optimale Werkzeug zur Bildbearbeitung. Etwas ungewohnt für uns „C++ Denker“ war der Umstand, dass in Matlab nicht jedes Element eines Arrays einzeln behandelt werden muss, sondern alles aufs Mal gemacht werden kann (meistens zumindest). Dank der IMAGE-PROCESSING-TOOLBOX war die Darstellung und das Einlesen von Bilddateien recht einfach und wir konnten uns dadurch mehr auf die Details des Algorithmus konzentrieren.

Bei unserer Informationssuche im Internet stellten wir fest, dass man nicht allem trauen kann. So war z.B. der JPEG-Standard das einzige Dokument, das erwähnte, dass vor der Transformation ein DC-level-shift erfolgen muss.

Für zusätzliche Verwirrung sorgten noch die beiden Farbräume YUV und YCbCr. In praktisch allen Dokumenten, die wir gefunden hatten, wurden diese Farbräume als äquivalent bezeichnet. Es stellte sich jedoch heraus, dass es zwei unterschiedliche Farbräume sind.

Besonders aufschlussreich waren die Entropierechnungen und das PSNR. Das PSNR (peak signal noise ratio) ist gegeben durch:

$$PSNR = 10 \cdot \text{Log} \left[ \frac{255^2}{\frac{1}{\text{zeilen} \cdot \text{spalten}} \cdot \sum_{i=1}^{\text{zeilen}} \sum_{j=1}^{\text{spalten}} (P(i, j) - Q(i, j))^2} \right]$$

Wobei  $P(i, j)$  dem Originalbild entspricht und  $Q(i, j)$  dem rekonstruierten Bild. *spalten*, *zeilen* entsprechen der Anzahl Zeilen respektive Spalten. Das PSNR eignet sich nur grob zur Abschätzung des subjektiven Fehlers. Ein PSNR von 25dB...30dB entspricht einer schlechten, 30dB...40dB einer guten und 40dB...50dB einer sehr guten Bildqualität.

Nachfolgend einige Vergleiche des Originalbildes mit den rekonstruierten Bildern, bei verschiedenen Qualitätsfaktoren.



**Fig. 43 lenna.bmp original Bild**

## BERICHT ZU JPEG UND JPEG2000

Wie untenstehend zu sehen ist, liegt das PSNR der beiden Chrominanzn deutlich tiefer als bei der Luminanz, trotzdem lassen sich noch keine Unterschiede erkennen. Die tiefen PSNR der beiden Chrominanzn stammen vom Farbsubsampling.

```
*****
qualitätsfaktor 100.00
lenna.bmp
*****
Das Peak-Signal-Noise-Ratio der Luminanz Y beträgt 58.91 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cb beträgt 43.07 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cr beträgt 42.06 dB
*****
Die Entropie der original Y -Komponente beträgt 7.07 bit/pixel
Die Entropie der original Cb-Komponente beträgt 5.84 bit/pixel
Die Entropie der original Cr-Komponente beträgt 5.97 bit/pixel
*****
Die Entropie der runlength codierten Y -Komponente beträgt 3.82 bit/pixel
Die Entropie der runlength codierten Cb-Komponente beträgt 2.86 bit/pixel
Die Entropie der runlength codierten Cr-Komponente beträgt 3.05 bit/pixel
*****
```



**Fig. 44 lenna.bmp jpeg-Codiert mit einem Qualitätsfaktor von 100**

Auch bei einem Qualitätsfaktor von 90 lässt sich noch kein Unterschied feststellen, obschon jetzt auch das PSNR der Luminanz deutlich gesunken ist.

```
*****
qualitätsfaktor 90.00
lenna.bmp
*****
Das Peak-Signal-Noise-Ratio der Luminanz Y beträgt 43.41 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cb beträgt 39.96 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cr beträgt 38.99 dB
*****
Die Entropie der original Y -Komponente beträgt 7.07 bit/pixel
Die Entropie der original Cb-Komponente beträgt 5.84 bit/pixel
Die Entropie der original Cr-Komponente beträgt 5.97 bit/pixel
*****
Die Entropie der runlength codierten Y -Komponente beträgt 1.60 bit/pixel
Die Entropie der runlength codierten Cb-Komponente beträgt 1.02 bit/pixel
Die Entropie der runlength codierten Cr-Komponente beträgt 1.04 bit/pixel
*****
```



Fig. 45 lenna.bmp jpeg-Codiert mit einem Qualitätsfaktor von 90

BERICHT ZU JPEG UND JPEG2000

```
*****
qualitätsfaktor 80.00
lenna.bmp
*****
Das Peak-Signal-Noise-Ratio der Luminanz Y beträgt 40.88 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cb beträgt 38.63 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cr beträgt 37.57 dB
*****
Die Entropie der original Y -Komponente beträgt 7.07 bit/pixel
Die Entropie der original Cb-Komponente beträgt 5.84 bit/pixel
Die Entropie der original Cr-Komponente beträgt 5.97 bit/pixel
*****
Die Entropie der runlength codierten Y -Komponente beträgt 1.21 bit/pixel
Die Entropie der runlength codierten Cb-Komponente beträgt 0.77 bit/pixel
Die Entropie der runlength codierten Cr-Komponente beträgt 0.82 bit/pixel
*****
```



**Fig. 46 lenna.bmp jpeg-Codiert mit einem Qualitätsfaktor von 80**

Auch mit einem Qualitätsfaktor von 50 zeigen sich noch keine störenden Qualitätseinbussen.

```
*****
qualitätsfaktor 50.00
lenna.bmp
*****
Das Peak-Signal-Noise-Ratio der Luminanz Y beträgt 37.77 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cb beträgt 36.90 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cr beträgt 35.96 dB
*****
Die Entropie der original Y -Komponente beträgt 7.07 bit/pixel
Die Entropie der original Cb-Komponente beträgt 5.84 bit/pixel
Die Entropie der original Cr-Komponente beträgt 5.97 bit/pixel
*****
Die Entropie der runlength codierten Y -Komponente beträgt 0.90 bit/pixel
Die Entropie der runlength codierten Cb-Komponente beträgt 0.52 bit/pixel
Die Entropie der runlength codierten Cr-Komponente beträgt 0.56 bit/pixel
*****
```



Fig. 47 lenna.bmp jpeg-Codiert mit einem Qualitätsfaktor von 50

Mit einem Qualitätsfaktor von 10 zeigen sich schon enorme Qualitätseinbussen. JPEG2000 verspricht auch bei so tiefen Bitraten, wie sich hier ergeben, noch eine gute Bildqualität zu liefern.

```
*****
qualitätsfaktor 10.00
lenna.bmp
*****
Das Peak-Signal-Noise-Ratio der Luminanz Y beträgt 31.35 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cb beträgt 31.83 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cr beträgt 31.42 dB
*****
Die Entropie der original Y -Komponente beträgt 7.07 bit/pixel
Die Entropie der original Cb-Komponente beträgt 5.84 bit/pixel
Die Entropie der original Cr-Komponente beträgt 5.97 bit/pixel
*****
Die Entropie der runlength codierten Y -Komponente beträgt 0.42 bit/pixel
Die Entropie der runlength codierten Cb-Komponente beträgt 0.25 bit/pixel
Die Entropie der runlength codierten Cr-Komponente beträgt 0.25 bit/pixel
*****
```



Fig. 48 lenna.bmp jpeg-Codiert mit einem Qualitätsfaktor von 10

Nach den Bildbeispielen werden nun die Entropieberechnungen erläutert.

Die Entropie  $H(X)$  (wird auch als der mittlere Informationsgehalt bezeichnet) berechnet sich zu:

$$H(X) = - \sum_{i=1}^n P(x_i) \cdot \text{Ld}[P(x_i)]$$

Wobei  $P(x_i)$  der Wahrscheinlichkeit des Zeichens  $x_i$  entspricht und  $\text{Ld}$  dem Zweier-Logarithmus

$$P(x_i) = \frac{\text{Häufigkeit}(x_i)}{\text{anzahl Zeichen}}$$

#### **Interpretation „Entropie der runlength codierten Komponenten“:**

Die „Entropie der runlength codierten Komponenten“ wurde auf bit pro pixel normiert. Das heisst, wir haben zuerst die theoretische Dateigrösse der runlength-Daten berechnet ( Dateigrösse = Entropie \* Anzahl Entropie-Symbole). Daraus ergibt sich die auf bit pro pixel normierte Entropie, indem man die Dateigrösse durch die totale Anzahl pixel dividiert. Mit sinkendem Qualitätsfaktor werden immer mehr AC-Koeffizienten zu Null quantisiert. Dadurch entstehen immer längere Null-Lauflängen. Was wiederum zur Folge hat, dass die Entropie sinkt.

Mit der Original- und Runlength-entropie lässt sich jetzt ein theoretischer Kompressionsfaktor  $n$  berechnen. Farbtiefe ist im Fall von „normalen“ Bitmaps 24 bit pro pixel.

$$n = \frac{\text{Farbtiefe}}{\text{RunlengthentropieY} + \text{RunlengthentropieCb} + \text{RunlengthentropieCr}}$$

#### **Beispiel:**

lenna.bmp      Qualitätsfaktor = 80

$$n = \frac{24 \text{ bit / pixel}}{1.22 \text{ bit / pixel} + 0.77 \text{ bit / pixel} + 0.82 \text{ bit / pixel}} = 8.54$$

Unter Verwendung von ACDSee mit dem selben Qualitätsfaktor ergab sich ein Kompressionsfaktor von

$$n = \frac{438\text{KB}}{26\text{KB}} = 16.84$$

Wobei 438KB die BMP Dateigrösse und 26KB die JPEG Dateigrösse ist. Diese zwei Ergebnisse lassen sich aber nicht direkt miteinander vergleichen, da unsere runlength-Daten noch nicht Huffman codiert sind.

Untenstehend ist das PSNR in Funktion der Bitrate und in Funktion des Qualitätsfaktors  $q$  zu sehen. Wie in den Grafiken zu sehen ist, liegen die PSNR der beiden Farbkomponenten (Cb, Cr) tiefer als das der Y-Komponente. Dieser Effekt ist auf das Farbsubsampling zurück zu führen. Damit ist sogleich auch erklärt, wieso die PSNR der Farbkomponenten bei grösser werdenden Bitraten nur noch schwach zunehmen. Das PSNR (in den beiden oberen Grafiken) der Cb-Komponente (Abweichung von Grau in Richtung Blau) liegt über alle Bitraten höher, als das der Cr-Komponente. Das rührt daher, dass das verwendete Testbild (lenna.bmp) praktisch kein Blau enthält aber sehr viel Rot. Mit anderen Worten: Die Fehler von Cb fallen weniger ins Gewicht als die von Cr.

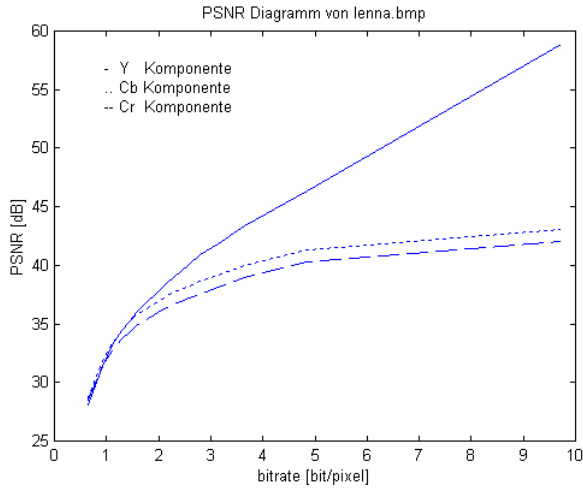


Fig. 49 PSNR Diagramm von lenna.bmp

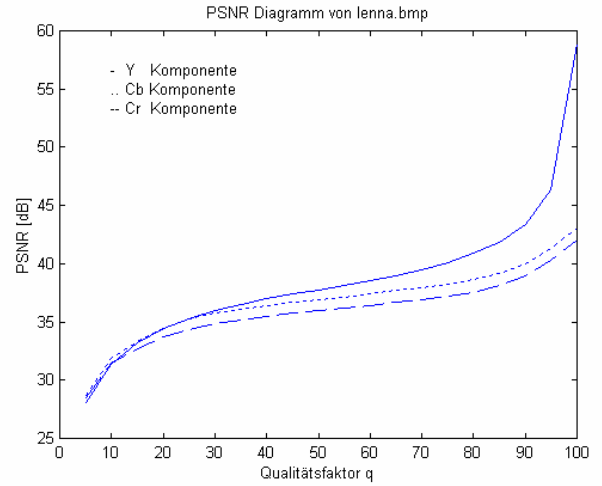


Fig. 50 PSNR versus  $q$  von lenna.bmp

Untenstehend wurden die gleichen Berechnungen am Bild parrots.bmp durchgeführt. Über alle Bitraten hat jetzt die Cr-Komponente ein höheres PSNR. Das ist auch wieder auf die Rot- und Blau-Anteile zurück zu führen.

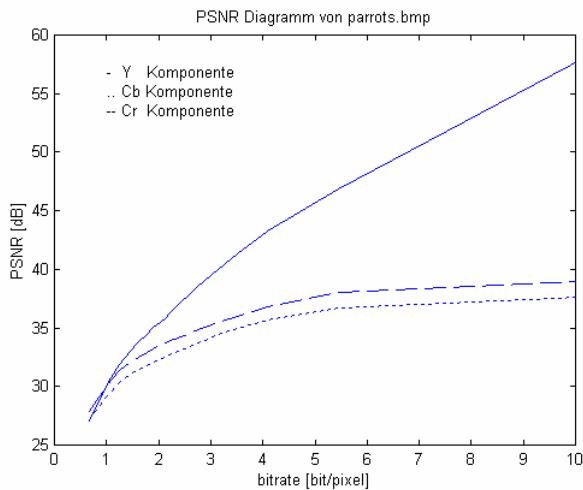


Fig. 51 PSNR Diagramm von parrots.bmp

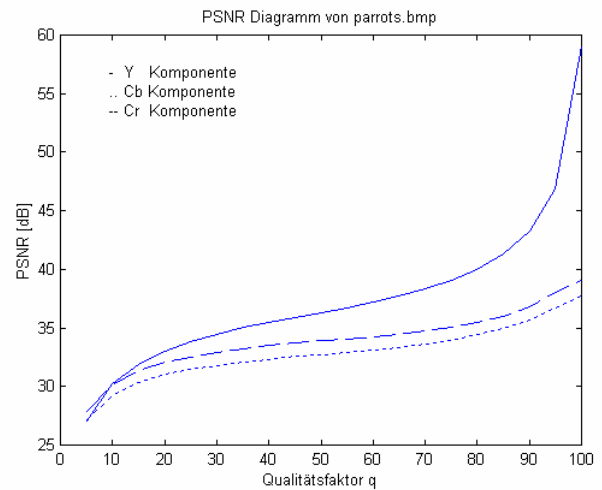


Fig. 52 PSNR versus  $q$  von parrots.bmp



Fig. 53 parrots.bmp

## 5 JPEG2000

### 5.1 Allgemeines

In diesem Kapitel wird auf die Implementierung des JPEG2000 Algorithmus eingegangen. Auf eine detaillierte Erklärung der Programm-Source wird verzichtet. Die Programm-Source befindet sich im Anhang B JPEG2000 SOURCECODE.

### 5.2 Vorgehen und Stolpersteine

Es wird nur das grobe Vorgehen der Implementierung und dessen Stolpersteine beschrieben.

Da der JPEG2000-Algorithmus noch kein offizieller Standard ist, war das Informationsangebot eher dürftig. So benutzten wir hauptsächlich das Dokument *JPEG2000 The Upcoming Still Image Compression Standard* und *JPEG 2000 Part I Final Committee Draft Version 1.0, 16. März 2000*. Der Standard war relativ schwer zu lesen und bot uns praktisch keine Anhaltspunkte über die Implementierung.

Die meisten Probleme bei der Implementierung des JPEG2000-Algorithmus hatten wir bei der Filterung. So hatten wir jeweils nach jeder Faltung des Bildes mit den Filtern, die entstehenden Ränder abgeschnitten, um die Grösse des Bildes nicht zu ändern. Bei der Rücktransformation erhielten wir dann ein recht verschmiertes Bild. Wir probierten alles Mögliche um diesen Effekt zu beseitigen. Die Lösung hatten wir aber erst knapp eine Woche vor Abgabetermin. So zeigte sich schlussendlich, dass die entstehenden Ränder dringend notwendig waren, um bei der Rücktransformation wieder ein scharfes Bild zu erhalten. Der Grund, warum man die Ränder nicht abschneiden darf ist, dass sich jeder Bildpunkt (bei der Faltung) auf das ganze Bild auswirkt.

Ein anderes Problem war die Quantisierung der Waveletkoeffizienten. Da wir keine Zahlenwerte für  $\Delta b$  gefunden hatten, konnten wir die Quantisierung nicht implementieren.

### 5.3 Funktionsbeschreibung

Es werden die implementierten Funktionen der Reihe nach, wie sie im JPEG2000 Algorithmus gebraucht werden, beschrieben (Programm-Source befindet sich im Anhang B JPEG2000 SOURCECODE).

#### **b = jpeg2000(imm)**

Übergabeparameter: **imm** entspricht dem Bild, das eingelesen werden soll, in „hochkomma“

Rückgabeparameter: **b** entspricht dem Differenzbild (Originalbild – rekonstruiertes Bild)

Beschreibung: Ist das Hauptprogramm.

**[y,z] = zeilenfiltern\_analyse(x)**

Übergabeparameter: **x** entspricht dem Bild, das gefiltert werden soll

Rückgabeparameter: **y** entspricht dem **x**, bei dem die Zeilen hochpassgefiltert und „downgesamplet“ sind.  
**z** entspricht dem **x**, bei dem die Zeilen tiefpassgefiltert und „downgesamplet“ sind.

Beschreibung: Filtert Zeilen und „downsamplet“ sie.

**[y,z] = spaltenfiltern\_analyse(x)**

- Übergabeparameter: **x** entspricht dem Bild, das gefiltert werden soll
- Rückgabeparameter: **y** entspricht dem **x**, bei dem die Spalten hochpassgefiltert und „downgesamlet“ sind.  
**z** entspricht dem **x**, bei dem die Spalten tiefpassgefiltert und „downgesamlet“ sind.
- Beschreibung: Filtert Spalten und „downsamlet“ sie.

**z = spaltenfiltern\_synthese(x,y)**

- Übergabeparameter: **x** entspricht dem Bild, das gefiltert werden soll.  
**y** entspricht dem Bild, das gefiltert werden soll.
- Rückgabeparameter: **z** entspricht dem „zusammengefügt“ Bild.
- Beschreibung: „upsamlet“ die Spalten, filtert sie und addiert dann die Ergebnisse.

**z = zeilenfiltern\_synthese(x,y)**

- Übergabeparameter: **x** entspricht dem Bild, das gefiltert werden soll.  
**y** entspricht dem Bild, das gefiltert werden soll.
- Rückgabeparameter: **z** entspricht dem „zusammengefügt“ Bild.
- Beschreibung: „upsamlet“ die Zeilen, filtert sie und addiert dann die Ergebnisse.

## 5.4 Erkenntnisse

Da wir leider keine konkreten Informationen über die Quantisierung hatten, können wir nachfolgend nur unquantisierte Bilder erläutern.

Daten der Y-Komponente:

```
*****
lenna.bmp
*****
Das Peak-Signal-Noise-Ratio beträgt 276.98 dB
*****
Die Entropie des Originalbildes beträgt 7.07 bit/pixel
*****
Die Entropie der Waveletkoeffizienten beträgt 10.05 bit/pixel
*****
```

Daten der Cb-Komponente:

```
*****
lenna.bmp
*****
Das Peak-Signal-Noise-Ratio beträgt 284.13 dB
*****
Die Entropie des Originalbildes beträgt 5.84 bit/pixel
*****
Die Entropie der Waveletkoeffizienten beträgt 9.55 bit/pixel
*****
```

Daten der Cr-Komponente:

```
*****
lenna.bmp
*****
Das Peak-Signal-Noise-Ratio beträgt 281.73 dB
*****
Die Entropie des Originalbildes beträgt 5.97 bit/pixel
*****
Die Entropie der Waveletkoeffizienten beträgt 9.67 bit/pixel
*****
```

JPEG Daten zum Vergleich:

```
*****
qualitätsfaktor 100.00
lenna.bmp
*****
Das Peak-Signal-Noise-Ratio der Luminanz Y beträgt 51.18 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cb beträgt 42.64 dB
Das Peak-Signal-Noise-Ratio der Chrominanz Cr beträgt 41.71 dB
*****
Die Entropie der original Y -Komponente beträgt 7.07 bit/pixel
Die Entropie der original Cb-Komponente beträgt 5.84 bit/pixel
Die Entropie der original Cr-Komponente beträgt 5.97 bit/pixel
*****
Die Entropie der runlength codierten Y -Komponente beträgt 3.82 bit/pixel
Die Entropie der runlength codierten Cb-Komponente beträgt 2.85 bit/pixel
Die Entropie der runlength codierten Cr-Komponente beträgt 3.04 bit/pixel
*****
```



**Fig. 54 lenna.bmp JPEG2000-codiert**

#### **Vergleich zwischen JPEG und JPEG2000:**

Das PSNR der Luminanz beim JPEG2000 codierten Bild ist deutlich höher, als das des JPEG codierten. Diese Tatsache ist darauf zurück zu führen, dass bei JPEG auch bei einem Qualitätsfaktor von 100 trotzdem noch gerundet wird (siehe 2.3.4). Die Chrominanz können nicht direkt miteinander verglichen werden, da JPEG2000 nach Empfehlung des Standards kein Farbsubsampling durchführt. Die Entropie der Waveletkoeffizienten lässt sich nicht mit der Entropie der runlength codierten DCT-Koeffizienten vergleichen, da sie nicht den gleichen Datentypen haben. Die DCT-Koeffizienten werden durch die Quantisierung wieder zu Integerwerten. Da wir die Waveletkoeffizienten nicht quantisiert haben, bleiben sie Fließkommazahlen.

## **5.5 Nötige Ergänzungen**

Um JPEG und JPEG2000 fair miteinander vergleichen zu können, muss folgendes getan werden:

- Die Waveletkoeffizienten müssen korrekt quantisiert werden (siehe 3.5.3).
- Die Entropierechnung der Waveletkoeffizienten muss angepasst werden. (Anzahl Container bei der Histogrammrechnung  $\rightarrow$  hist( ) )

## Schlusswort

Diese Studienarbeit verschaffte uns einen Einblick in die Bildkomprimierung. Es war interessant zu sehen, wie ein so viel genutzter Algorithmus ,wie JPEG, funktioniert. Es zeigte sich schliesslich, dass der JPEG-Algorithmus „auch nur“ auf grundlegenden Prinzipien der Bildverarbeitung basiert. Eine besondere Herausforderung war die Implementierung von JPEG2000, da es auch in der Industrie noch nicht vollständig implementiert ist. Leider konnten wir nicht den gesamten Algorithmus von JPEG2000 implementieren, da uns wesentliche Informationen über die Quantisierung fehlten.

Es war eine besondere Motivation zu hören, dass die Implementierung des JPEG2000-Algorithmus auch in folgenden Semesterarbeiten fortgesetzt wird und man dann von unseren Ergebnissen profitieren kann.

Wir danken Herrn Schuster für seine kompetente und hilfreiche Betreuung. Er nahm sich viel Zeit bei den Besprechungen (auch neben den regulären) und gab uns hilfreiche Tips beim Lösen von Problemen.

## Quellenverzeichnis

1. *Gregory K. Wallace*: The JPEG Still picture Compression Standard,  
[http://www.cs.cmu.edu/~bumba/filing\\_cabinet/papers/wallace-jpeg.pdf.gz](http://www.cs.cmu.edu/~bumba/filing_cabinet/papers/wallace-jpeg.pdf.gz)
2. *Sandra Baril*: Vortrag 11: JPEG, <http://goethe.ira.uka.de/redundanz/vortrag11/>
3. *Joachim Schwarz und Guido Soermann*: Tutorial,  
[http://ttrip1.fh-worms.de/sem/ws95\\_96/kompressionsalgorithmen/node1.html](http://ttrip1.fh-worms.de/sem/ws95_96/kompressionsalgorithmen/node1.html)
4. *Thorsten Milde*: Videokompressionsverfahren im Vergleich, dpunkt, 1.Auflage, ISBN 3-920993-23-3
6. *A. N. Skodras, C. A. Christopoulos und T. Ebrahimi* : JPEG2000 The Upcoming Still Image Compression Standard, [http://etro.vub.ac.be/~chchrist/recpad00\\_paper.pdf](http://etro.vub.ac.be/~chchrist/recpad00_paper.pdf)
7. *A. N. Skodras, C. A. Christopoulos und T. Ebrahimi* : The next generation still image coding system, <http://etro.vub.ac.be/~chchrist/recpad2000.zip>
8. *Martin Boilek, C. A. Christopoulos, E. Majani* : JPEG 2000 Part I Final Committee Draft Version 1.0, 16. März 2000  
<http://www.jpeg.org/fcd15444-1.zip>

## **Anhang A      JPEG SOURCECODE**

## **Anhang B      JPEG2000 SOURCECODE**