Seminar zur Numerischen Analysis
im Wintersemester 2008/2009

Signalverarbeitung und Wavelets

Bilddatenkompression mit JPEG und JPEG2000

Laura Hochstrat

06.02.2009

Inhaltsverzeichnis

1 Einleitung

Ein digitales Bild kann man als zweidimensionales Feld interpretieren. Ein Graubild (engl. greymap) besteht aus einer Komponente, der Wert in dem Feld gibt die Helligkeit des Bildes in dem entsprechendem Pixel wieder. Farbbilder setzen sich aus mehreren Komponenten zusammen, z.B. aus drei Komponenten, bei einer Darstellung in einem additiven Farbraum. Abbildung 1 zeigt die Werte eines Beispiel-Graustufenbildes in Abhängigkeit der Position des Pixels.

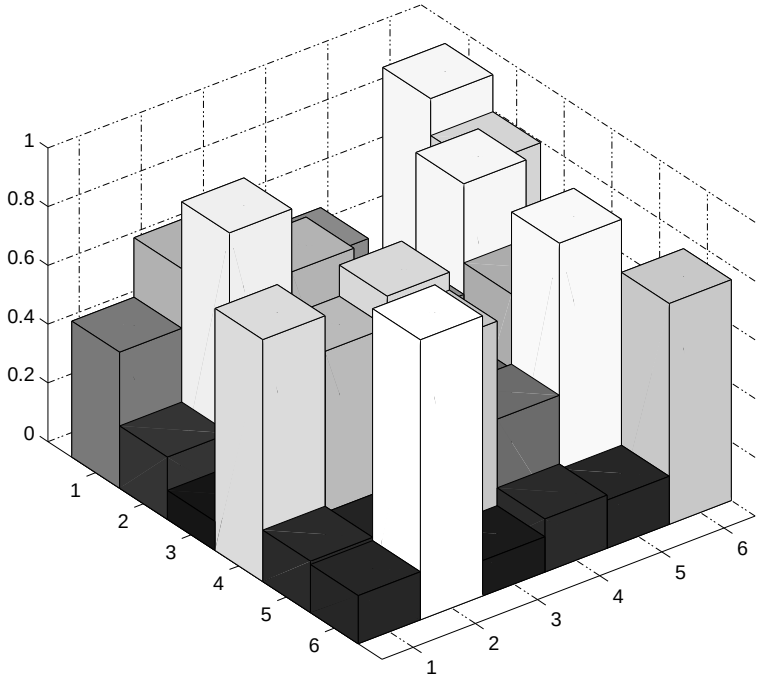


Abbildung 1: Schematische Darstellung eines Graustufenbildes als Histogramm

2 Grundlagen der Kodierung

Bevor auf die konkreten Kompressions-Standards JPEG und JPEG2000 eingegangen wird, werden zwei Arten der Kodierung vorgestellt, die in den Standards wieder aufgegriffen werden.

2.1 Arithmetische Kodierung

Ein Nachteil von Präfixcodes (z.B. Huffman-Codes, siehe [11]) ist, dass diese nur ganzzahlige Codelängen pro Symbol realisieren können, auch wenn der Informationsgehalt einen gebrochenen Wert hat. Um dieses Problem zu umgehen, kodiert die arithmetische Kodierung nicht einzelne Symbole, sondern eine ganze Folge von Symbolen in einem einzelnen Codewort, was eine bessere Annäherung an die Signalentropie bewirkt [14]. Das Prinzip der arithmetischen Kodierung wurde in einem vorangegangen Vortrag [11] bereits geschildert und an dieser Stelle kurz wiederholt.

Gegeben sei ein Alphabet von Symbolen s_i für $1 \leq i \leq n$ und dazugehörige Wahrscheinlichkeiten p_i für $0 \leq i \leq n$ gegeben, wobei diese als Summenwahrscheinlichkeiten dargestellt sein sollen. Es gilt also $p_0 = 0$ und $p_i = p_{i-1} + p(s_i)$ für $1 \leq i \leq n$, wobei $p(s_i)$ die Wahrscheinlichkeit von s_i ist. Es wird ein Startintervall definiert mit den Grenzen low und $high$, außerdem der aktuelle Intervallbereich $range = high - low$. Dieses Intervall wird anschaulich in n Teilintervalle gemäß den Wahrscheinlichkeiten der Symbole des Alphabets unterteilt. Man erhält als Teilintervalle

$$[low + range \cdot \ p_{i-1}, low + range \cdot p_i] \text{ für } 1 \leq i \leq n-1.$$

Sei s_k das zu kodierende Symbol, dann ist

$$high = low + range \cdot p_k, \quad low = low + range \cdot \ p_{k-1}$$

die Grenzen des neuen Intervalls. Sind alle Symbole des zu kodierenden Wortes durchlaufen worden, so wird aus dem aktuellen Intervall ein Element mit möglichst kurzer Bitdarstellung gewählt. Da die erzeugten Zahlen im Intervall $[0, 1]$ liegen, besitzen sie eine Dualdarstellung aus negativen Zweipotenzen $\sum_{i=1}^{n} c_i 2^{-i}$.

Der Dekodierer kann anhand dieses Codewortes die Entwicklung der Intervalle nachvollziehen und somit das ursprüngliche Signal wiederherstellen. Dafür muss der Kodierer jedoch die Anzahl der ursprünglichen Symbole mitgeteilt bekommen, da das Codewort in auch in beliebig vielen kleineren Intervallen enthalten ist. [14]

Beispiel 1. (vergleiche Abbildung 2)
Das Alphabet sei das binäre Alphabet $s = [0; 1]$ mit Summenwahrscheinlichkeit $p = [0; \frac{1}{4}; 1]$. Außerdem sei das Startintervall gegeben durch [0,1]. Das zu kodierende Wort sei 01101. Dann ergibt sich

$$\begin{array}{lccc}
\text{Schritt 1:} & low = 0, & high = 1, & range = 1, \\
\text{Schritt 2:} & low = 0, & high = \frac{1}{4}, & range = \frac{1}{4}, \\
\text{Schritt 3:} & low = \frac{1}{4^2}, & high = \frac{4}{4^2}, & range = \frac{3}{4^2}, \\
\text{Schritt 4:} & low = \frac{7}{4^3}, & high = \frac{16}{4^3}, & range = \frac{9}{4^3}, \\
\text{Schritt 5:} & low = \frac{28}{4^4}, & high = \frac{37}{4^4}, & range = \frac{9}{4^4}.
\end{array}$$

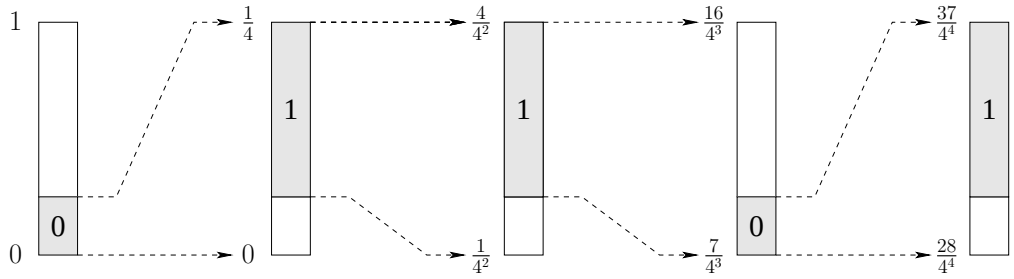


Abbildung 2: Beispiel für arithmetische Kodierung

Aus dem Endintervall wählt man nun eine Zahl, in diesem Fall bietet sich $0.125 = 2^{-3}$ an, da man nur drei Bits zur Darstellung benötigt. Man erhält als Code 001.

Dieses Grundstruktur des arithmetischen Kodierers wird in der Praxis nicht verwendet, da sie zwei entscheidende Nachteile hat: Zum einen kann das erste Bit erst nach der Kodierung des gesamten Signals übertragen werden und zum anderen ist eine beliebig hohe Genauigkeit der Intervallgrenzen praktisch unmöglich zu handhaben, da ein Rechner nur eine endliche Genauigkeit darstellen kann. [14]

Festkomma-Implementierung Aus diesen Gründen wurde die Festkomma-Implementierung der arithmetischen Kodierung erfunden, die die Intervallgrenzen mit ganzen Zahlen annähert. Sei n wieder die Anzahl der Symbole des Alphabets $[s_1; \ldots; s_n]$. Statt der relativen Häufigkeiten, die zuvor benutzt wurden, werden jetzt die absoluten Häufigkeiten $[H_0; \ldots; H_n]$ verwendet, um Gleitkommaoperationen zu vermeiden. Diese sind wieder als summierte Wahrscheinlichkeiten zu verstehen. Zunächst wird die Auflösungsbreite B festgelegt, das ist die Anzahl der Bits, die zur Darstellung der Intervallgrenzen verwendet wird. Diese sollte nicht zu klein gewählt werden, um eine hinreichend gute Approximation der eigentlich reellen Intervallgrenzen zu ermöglichen. Die größte darstellbare

Zahl ist $M := 2^B - 1$. Die Intervallgrenzen werden initialisiert mit $low = 0$ und $high = M$. Dann gilt für das Intervall bei jedem zu kodierendem Symbol

$$\begin{aligned} range &:= high - low + 1, \\ high &:= low + range \cdot \frac{H_{i+1}}{H_n} - 1, \\ low &:= low + range \cdot \frac{H_i}{H_n}. \end{aligned}$$

Die Division durch H_n dient dabei der Skalierung. Diese Version der arithmetischen Kodierung überträgt nicht erst Bits, wenn alle Symbole kodiert sind, sondern überträgt ein Bit, sobald es in der Darstellung eindeutig ist, also sobald das erste Bit von low und $high$ übereinstimmen. Dies ist immer dann der Fall, wenn sich das aktuelle Intervall komplett in einer Hälfte des Zahlenbereiches $[0, M]$ befindet. Als Hilfe definieren wir die Variablen

$$Q1 = \frac{M}{4} + 1, \quad Q2 = 2 \cdot Q1, \quad Q3 = 3 \cdot Q1,$$

außerdem initialisieren wir den Zähler $k = 0$.

Um zu entscheiden, wann Bits herausgeschrieben werden, überprüft man zunächst in welchem Teil des Zahlenbereiches das aktuelle Intervall liegt, dabei sind drei Fälle zu beachten:

1. Das Intervall befindet sich in der unteren Hälfte, d.h. $high < Q2$, dann sendet man eine '0' und k Einsen und setzt $k = 0$.

2. Das Intervall befindet sich in der oberen Hälfte, d.h. $low \geq Q2$, dann sendet man eine '1' und k Nullen, setzt $k = 0$ und verschiebt das Intervall durch $low = low - Q2$, $high = high - Q2$.

3. Das Intervall befindet sich in der mittleren Hälfte, d.h. $low \geq Q1$ und $high < Q3$, man setzt $k = k + 1$ und $low = low - Q1$ sowie $high = high - Q1$.

Tritt einer dieser Fälle ein, so wird das Intervall neu skaliert: $low = 2 \cdot low$, $high = 2 \cdot high + 1$.

2.2 Lauflängenkodierung

Die Lauflängenkodierung (engl. run-length coding, RLC) basiert auf der einfachen Idee, Sequenzen eines Zeichen nicht durch Wiederholung desselben Zeichens abzuspeichern, sondern als einmal das Zeichen und die Länge der Sequenz. Besonders effizient ist diese Kodierung bei Daten in denen oft lange Ketten desselben Zeichens auftreten. Hingegen sollte die Lauflängenkodierung nicht benutzt werden, um Daten zu speichern, die oft wechselnde Zeichen enthalten, da man bei solchen im schlimmsten Fall durch das Speichern von Zeichen und Anzahl die Größe einer Datei verdoppelt. Anwendung findet die Lauflängenkodierung z.B. beim Fax, da dort große Flächen weiß (W) sind unterbrochen von einzelnen schwarzen (S) Pixeln.

Ein kurzes Beispiel: Kodiert man die Zeichenkette
WWWWWWWWWWWWSWWWWWWWSWWWWWWWWWWWWWWWW
mit einfacher Lauflängenkodierung erhält man 11W1S6W1S16W, was interpretiert wird als elf W's, ein S, sechs W's, einem S und 16 W's.

Es muss darauf geachtet werden, dass der Code eindeutig ist. In diesem Beispiel wird die Lauflängenkodierung zeichenweise angewendet, wobei ein Zeichen einem Byte entsprechen soll. Kodiert man die Anzahl auch mir einer festen Anzahl Bits, hat man die Eindeutigkeit garantiert.

Realistisch sind auch Zeichenketten, in denen längere Sequenzen von gleichen Zeichen vorkommen, jedoch auch viele Zeichen nur einzeln oder in sehr kurzen Sequenzen auftauchen. In diesem Fall kodiert man nicht alle Zeichen als (Anzahl,Zeichen), sondern nur die längeren Zeichen. In einem solchen Code muss man markieren, an welcher Stelle die Lauflängenkodierung genutzt wird, dazu wird ein Marker verwendet. Ein Marker ist ein Symbol, dass nicht im Codestream vorkommt.

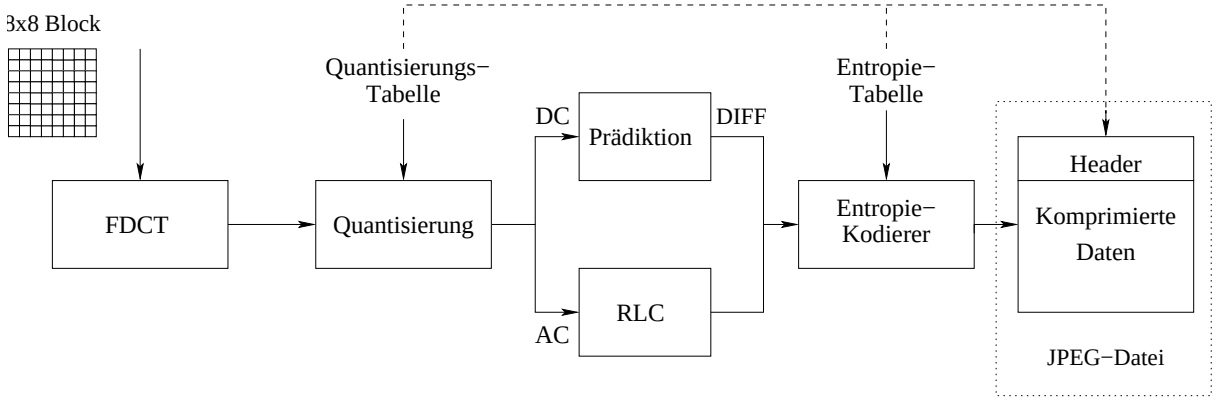


Abbildung 3: Kompression bei JPEG

Existiert ein solches Symbol nicht, kann man das Problem umgehen, indem man z.B. als Marker ein seltenes Symbol B einsetzt und die wirklich vorkommenden B's als BB kodiert. Sequenzen von Zeichen werden jetzt als (Marker,Anzahl,Zeichen) dargestellt, man benötigt also drei Bytes. Aus diesem Grund lohnt es sich auch nur Sequenzen von mehr als drei Zeichen mit Lauflänge zu kodieren. Die restlichen Zeichen bleiben in ihrer ursprünglichen Darstellung. Es sei noch angemerkt, dass je nach Verwendungszweck verschiedene Variationen der Lauflängenkodierung existieren, die auch oft mit anderen Arten der Kodierung kombiniert werden.

3 JPEG

Die Joint Photographic Expert Group (JPEG) ist ein Komitee, das sich 1986 aus Arbeitsgruppen der ISO (International Organization for Standardization) und der ITU-T (International Telecommunications Union-Telecommunications-Sector) zusammensetzte, mit dem Ziel einen Standard von Bildkompressionsverfahren zu definieren. Im folgenden beschränken wir uns auf die Beschreibung des mit „Baseline“ bezeichneten Teil des JPEG-Standards. Dieser umfasst die grundlegenden Methoden, die eine Implementierung von JPEG mindestens beinhalten muss.

Nachdem verschiedene Vorschläge abgewogen wurden, einigte man sich auf einen Standard, der auf einer diskreten Kosinus-Transformation und einer Variation des Huffman-Codes für verlustbehaftete Kompression basiert. Zusätzlich lässt der Standard auch arithmetische Kodierung, sowie progressive und hierarchische Kodierung zu. Weiterhin ist im JPEG-Standard eine unabhängige Funktion zur verlustfreien Kompression definiert.

Dieser als JPEG bekannter Standard mit der offiziellen Bezeichnung ISO/IEC 10918-1, etablierte sich in den 90'ger Jahren und ist Dank zahlreicher Implementierungen ein weit verbreitetes und bekanntes Mittel zur Kompression und Speicherung von Bildern. [3, 14]

Abbildung 3 zeigt eine Übersicht der *Kompression* bei JPEG, Abbildung 4 zeigt entsprechend die Dekompression. Die folgenden Abschnitte werden genauer auf die einzelnen Schritte eingehen.

3.1 Vorverarbeitung

In der Vorverarbeitung (engl. preprocessing) wird eine Farbtransformation durchgeführt, um bessere Kompression zu erreichen. Danach wird jede Komponente einzeln betrachtet und in 8x8 Blöcke unterteilt (engl. tiling), da die nachfolgende Kosinus-Transformation auf diesen Blöcken arbeitet.

3.1.1 Farbtransformation

Ein weit verbreitetes optisches Modell, das z.B. bei Bildschirmdarstellungen und Projektoren genutzt wird, ist das Rot-Grün-Blau-Modell (RGB). Dieses Farbmodell basiert auf additiver Farbmi-

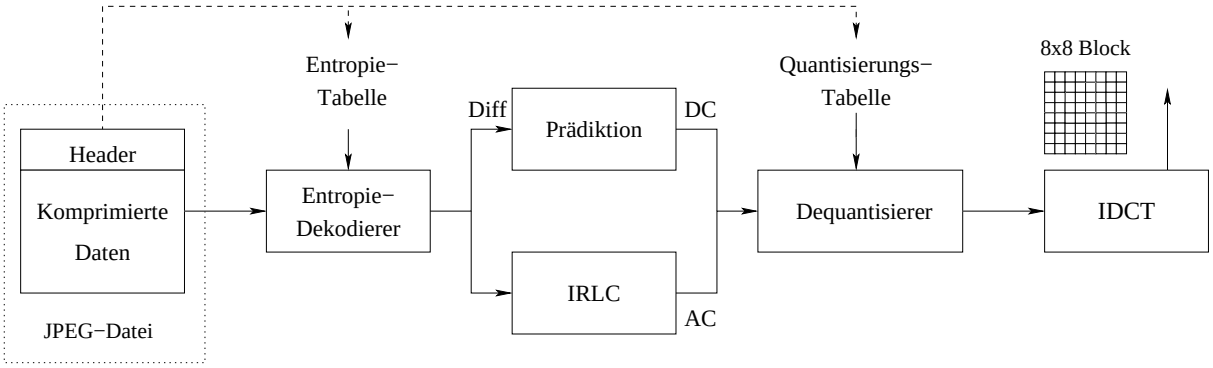


Abbildung 4: Dekompression bei JPEG

schung. Ein Farbbild in dieser Darstellung besteht also aus drei Farbwerten pro Pixel.

Beim JPEG-Standard wechselt man die Basis, in der die Farben dargestellt werden und benutzt statt Rot, Grün, Blau das sogenannte YCbCr-Farbmodell. In diesem Farbmodell stellen die einzelnen Komponenten die Helligkeit (Luminanz Y), sowie die Farbigkeit (Chrominanz), eine Richtung Blau zu Gelb (C_b), die zweite Richtung Rot zu Türkis (C_r), dar. Man wählt dieses Modell, das sich empirisch ergeben hat, weil man Helligkeitsunterschiede eher wahrnimmt als Farbtonunterschiede. Da man somit bei den beiden Chrominanz-Komponenten größere Verluste hinnehmen kann, als in der Luminanz-Komponente, weswegen die Chrominanz-Komponenten oft gröber quantisiert werden als die Luminanz-Komponente.

Außerdem unterstützt JPEG ein sogenanntes Unterabtasten der Chrominanzen (C_b und C_r) und benutzt dadurch das begrenzte Farbsehen des Menschen, um die Datenmenge zu verringern.

Der beschriebene Basiswechsel bei typischer $3 \cdot 8$ Bit Farbtiefe berechnet sich über [3]

$$\begin{pmatrix} Y \\ C_b \\ C_r \end{pmatrix} \approx \begin{pmatrix} 0 \\ 128 \\ 128 \end{pmatrix} + \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.168736 & -0.331264 & 0.5 \\ 0.5 & -0.418688 & -0.081312 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}.$$

Die Daten, die im Bereich $0 \ldots 255$ liegen, werden nun in den Wertebereich $-128 \ldots 127$ verschoben, so dass wir eine Zentrierung um die Null erreichen. [14]

Der weitere Kompressionsprozess verläuft komponentenweise, wir beschränken uns also auf die Betrachtung einer einzelnen Komponente, die auch als Graubild (engl. greymap) interpretiert werden kann.

3.1.2 Tiling

Beim sogenannten Tiling wird das Bild in 8x8 Pixel große Blöcke unterteilt. Da im Allgemeinen die Höhe bzw. Breite des Bildes nicht durch 8 teilbar ist, werden die Blöcke an den Rändern des Bildes aufgefüllt (siehe Abbildung 5), wobei es im JPEG-Standard keine feste Vorschrift gibt, wie dies geschehen soll. Eine gängige Praxis ist jedoch, die letzte Zeile bzw. Spalte des Bildes entsprechend oft an den Rändern zu wiederholen. Die darauf folgende diskrete Kosinus-Transformation wird nun separat auf den einzelnen Blöcken durchgeführt.

3.2 Diskrete Kosinus-Transformation

Bei der zweidimensionalen diskreten Kosinus-Transformation (engl. discret cosine transformation, DCT) wird jeder 8x8-Block als Linearkombination von 2D-Kosinusfunktionen, den Basisfunktionen, dargestellt. Die Koeffizienten $(X(k,l))_{0 \le k,l \le 7}$ der DCT berechnen sich über

$$X(k,l) = \frac{C_k \cdot C_l}{4} \cdot \sum_{n=0}^{7} \sum_{m=0}^{7} x(n,m) \cdot \cos\left(\frac{(2n+1) \cdot k \cdot \pi}{16}\right) \cdot \cos\left(\frac{(2m+1) \cdot l \cdot \pi}{16}\right) \tag{1}$$

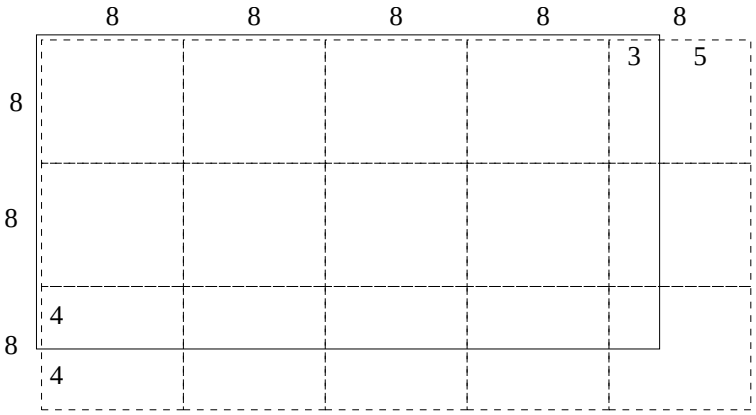


Abbildung 5: Unterteilung eines Bildes in 8x8 Blöcke mit Ergänzung am Rand

mit

$$C_i = \begin{cases} \frac{1}{\sqrt{2}}, & \text{für} \quad i = 0, \\ 1, & \text{für} \quad i \neq 0, \end{cases}$$

wobei x(k,l) der Wert des jeweiligen Pixels ist. [14] Abbildung 6 zeigt, als Graustufen dargestellt, die 64 Basisfunktionen. Dabei ist die erste Basisfunktion (oben links) konstant, nach rechts bzw. unten erhöht sich die Frequenz der Funktion in die jeweilige Richtung. Man unterscheidet die Koeffizienten

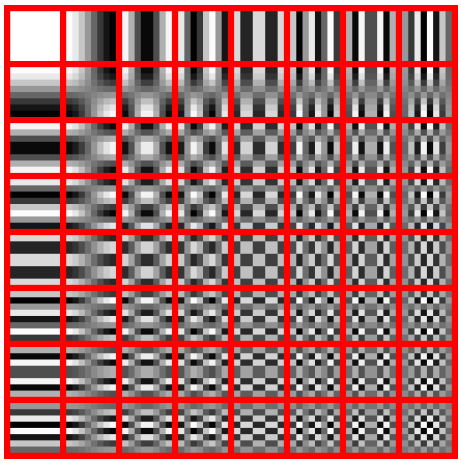


Abbildung 6: Das ursprüngliche Bild ist eine Linearkombination dieser 64 Blöcke [3]

in Gleichstromanteil (engl. direct current, DC) und Wechselstromanteile (engl. alternating current, AC), wobei DC die Koeffizienten der konstanten Basisfunktion darstellen. Die AC sind die restlichen Koeffizienten, sie werden nach aufsteigenden Frequenzen in einer Zick-Zack-Abtastung sortiert, siehe Abbildung 7. Die Tabellen 1 und 2 zeigen ein Beispiel für die Kosinus-Transformation eines solchen Blockes.

Die Berechnungsvorschrift der inversen diskreten Kosinus-Transformation (engl. inverse discrete cosine transformation, IDCT) lautet entsprechend

$$x(n,m) = \sum_{k=0}^{7} \sum_{l=0}^{7} \frac{C_k \cdot C_l}{4} \cdot X(k,l) \cdot \cos(\frac{(2n+1) \cdot k \cdot \pi}{16}) \cdot \cos(\frac{(2m+1) \cdot l \cdot \pi}{16}). \tag{2}$$

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

Tabelle 1: Beispiel: Originale Grauwerte eines 8x8-Blocks [14]

3.3 Quantisierung

Die Koeffizienten, die man bei der Kosinus-Transformation erhalten hat, sind reelle Zahlen und auf dem Rechner im Allgemeinen nicht exakt darstellbar, da ihre Bitfolge nicht abbricht. Ab einer bestimmten Genauigkeit ist es nicht sinnvoll, mehr Stellen eines Wertes zu speichern, da es keinen sichtbaren oder nur einen hinnehmbaren Fehler produziert, wenn man diesen Speicherplatz spart. Man verzichtet also auf die exakte Rekonstruierbarkeit der Daten zugunsten einer Näherung, die weniger Speicherplatz benötigt. Wie bereits in [11] diskutiert wird, verwendet man eine Quantisierung, die reelle Zahlen eines kleinen Intervalls durch ein Symbol aus einem endlichen Alphabet repräsentiert. Für die Quantisierung legt der JPEG-Standard

$$q(k,l) = \left\lfloor \frac{X(k,l)}{Q_{k,l}} + 0.5 \cdot \mathrm{sgn}(X(k,l)) \right\rfloor \tag{3}$$

fest, es handelt sich also um eine gleichmäßige Quantisierung, da alle Intervalle dieselbe Größe haben.

Die Wahl der Quantisierungswerte $Q_{k,l}$ bleibt dabei dem Anwender überlassen, daher müssen die entsprechenden Informationen im Header der Datei übermittelt werden. Auch wenn der Standard keine Quantisierungs-Tabellen vorschreibt, so macht er doch einen Vorschlag, der sich empirisch bewährt hat. In diesem Vorschlag finden sich zwei verschiedene Quantisierungs-Tabellen für Luminanz und Chrominanz, siehe Tabelle 3.

Der Anwender von JPEG gibt eine gewünschte Qualität im Bereich von 0 bis 100 vor, dieser Wert skaliert die Quantisierungstabelle.

3.4 Entropiekodierung

Bei der Codierung der quantisierten Koeffizienten werden Gleich- (DC) und Wechselanteile (AC) getrennt behandelt. Der Grund für diese Vorgehensweise liegt darin, dass es bei vielen Bildern größere, nahezu einfarbige Flächen gibt. Daher ist der Gleichanteil, also der Mittelwert eines Blockes, von

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	−0.6
−7.1	−1.9	0.2	1.5	0.9	−0.1	0.0	0.3
−0.6	−0.8	1.5	0.9	−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

Tabelle 2: Beispiel: zugehörige DCT-Koeffizienten [14]

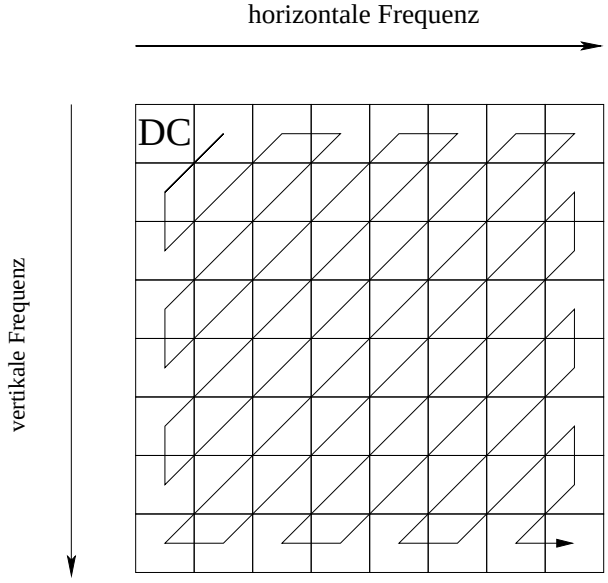


Abbildung 7: Zick-Zack-Abtastung der AC-Koeffizienten

benachbarten Blöcken oft ähnlich groß. Deswegen benutzt man zur Codierung der Gleichanteile die sogenannte Prädiktion. Diese betrachtet statt der einzelnen Gleichanteile der Blöcke die Differenzen (engl. difference, diff) zwischen diesen, also für den i-ten Code-Block

$$DIFF = \begin{cases} DC_i - DC_{i-1}, & \text{für} \quad i \neq 0, \\ DC, & \text{für} \quad i = 0. \end{cases}$$

Die so erhaltenen Differenzen sind im Allgemeinen mit hoher Wahrscheinlichkeit nahe bei Null, haben somit also eine Wahrscheinlichkeitsverteilung, die für Entropiekodierung günstiger ist als die der ursprüngliche Koeffizienten. Die Kodierung dieser Differenzen erfolgt mit einem modifizierten Huffman-Code. Dieser unterteilt den Wertebereich in Kategorien. Für jeden zu kodierenden Wert wird zunächst die Kategorie selektiert und die entsprechende Kennung ausgegeben, welche die Länge des Codes für den speziellen Wert festlegt. Anschließend folgt der Code für den speziellen Wert innerhalb der Kategorie.

Wiederum schreibt der JPEG-Standard keine feste Huffman-Tabelle vor, sondern überlässt dem Benutzer die Wahl. Deswegen müssen die Informationen der Huffman-Tabelle im Header des Bildes übermittelt werden. Tabelle 4 zeigt mögliche Codes für die Kategorien. Für die voraussichtlich öfter auftretenden Kategorien der betragsmäßig kleinen Werte, werden hierbei möglichst kurze Codes benutzt. Da die Chrominanzen stärker quantisiert wurden als die Luminanzen, sind bei diesen kleine Kategorien noch wahrscheinlicher, deswegen ist es sinnvoll, für die Chrominanzen eine andere Code-Tabelle zu benutzten.

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	72	95	98	112	100	103	99

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

Tabelle 3: Beispiel für Quantisierungstabellen für Luminanz und Chrominanz [14, Seite 147]

Kategorie	DIFF-Wert	Code (Lum)	Code (Chrom)
0	0	00	00
1	-1,1	010	01
2	-3,-2,2,3	011	10
3	-7...-4,4...7	100	110
4	-15...-8,8...15	101	1110
5	-31...-16,16...31	110	11110
6	-63...-32,32...63	1110	111110
7	-127...-64,64...127	11110	1111110
8	-255...-128,128...255	111110	11111110
9	-511...-256,256...511	1111110	111111110
10	-1023...-512,512...1023	11111110	1111111110
11	-2047...-1024,1024...2047	111111110	11111111110

Tabelle 4: Beispiele für einen Huffman-Code für die Kategorien [14, Seite 149]

Bei der Kodierung der Wechselstromanteile berücksichtigt man die Korrelation der AC-Koeffizienten eines Blockes. Abhängig von der Quantisierung der Wechselstromanteile hat man viele Koeffizienten gleich Null, daher lohnt sich eine Lauflängenkodierung (engl. run length coding, RLC).

Vor der eigentlichen Kodierung der Wechselstromanteile sortiert man diese nach aufsteigenden Frequenzen mit der Zick-Zack-Abtastung, siehe Abbildung 7. Diese Reihenfolge ist vorteilhaft, um die Korrelation der Koeffizienten, die zu ähnlichen Frequenzen gehören, optimal auszunutzen. Außerdem sind die hochfrequenten Anteile mit höherer Wahrscheinlichkeit Null als die niedrigfrequenten Anteile, daher hat man bei dieser Sortierung möglichst viele Nullen am Ende der zu kodierenden Koeffizienten.

Die AC-Koeffizienten ungleich Null werden wie die DC-Koeffizienten in Kategorien unterteilt und es wird der Abstand zum Vorgänger ungleich Null ermittelt, dies entspricht der Anzahl der Nullen zwischen diesen beiden Koeffizienten und wird Lauflänge genannt. Die Lauflänge hat immer einen Wert von 0 bis 15, wobei es einen speziellen Code für 15 Nullen auf die noch eine Null folgt gibt. Für jede Kombination von Kategorie und Lauflänge wird nun ein Code festgelegt, dabei wird die Korrelation zwischen Kategorie und Lauflänge berücksichtigt, denn ein großer Koeffizient nach mehreren Nullen ist z.B. sehr unwahrscheinlich.

Nun wird für jeden Koeffizienten ungleich Null der Code für Lauflänge und Kategorie, gefolgt von einem Code für den exakten Koeffizienten ausgegeben, dessen Länge abhängig von Lauflänge und Kategorie ist. Es gibt außerdem einen speziellen Code, der nach dem letzten Koeffizienten ungleich Null ausgegeben wird und das Ende des Blocks signalisiert (engl. end of block, EOB). Alle restlichen Nullen müssen also nicht mehr kodiert werden. Alle Codes können wie auch bei der Kodierung der DC-Koeffizienten vom Anwender generiert werden, die entsprechenden Informationen müssen wieder im Header der Datei enthalten sein.

3.5 Dateiformat

Der Bitstrom einer JPEG-Datei setzt sich aus Segmenten zusammen, dabei gibt es zwei Segmenttypen, Markersegmente und entropiecodierte Segmente. Die Markersegmente enthalten Zusatzinformationen z.B. die Quantsierungs- und Codetabellen. Entropiecodierte Segmente hingegen enthalten die eigentlichen Daten.

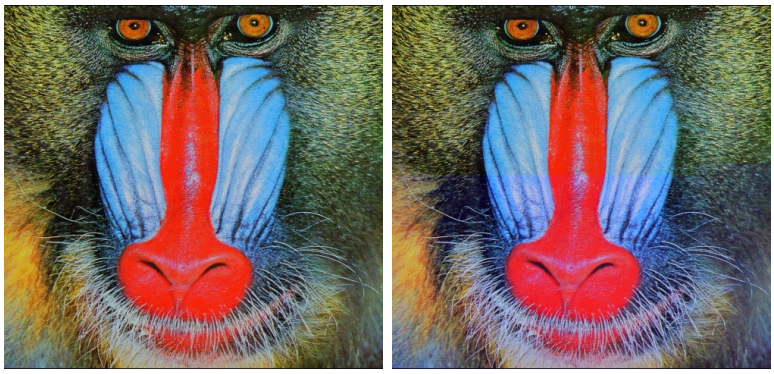
Die Datei setzt setzt sich aus Rahmen (engl. frame) zusammen. Jeder Rahmen beginnt mit einen Kopf (engl. frame-header) und besteht aus mehreren Läufen (engl. scans), wobei jeder Lauf einen Lauf-Kopf (engl. scan-header) und ein oder mehrere entropiecodierte Segmente enthält. Bestimmte Bitfolgen, sogenannte Marker, kennzeichnen Anfang und Ende von Bild, Rahmen, Läufen und Definitionen von Informationen wie Quantisierungs- und Kodierungsinformationen. Diese Marker

beginnen hexadezimal geschrieben mit '0xFF' gefolgt von einer Bitfolge, die die Art des Markers angibt, z.B 'D8' für Anfang des Bildes. Die Details zum genauen Aufbau der Datei finden sich im JPEG-Standard [7].

3.6 Vor- und Nachteile

Ein Vorteil des JPEG-Standards ist die Einfachheit des Kompressionsvorgangs, der Rechenaufwand ist also relativ gering. Außerdem ist der JPEG-Standard sehr flexibel, der Anwender kann Quantisierungs-Tabellen und Huffman-Codes vorgeben. Auf der anderen Seite hat JPEG-Standard einige Eigenschaften, die stark verbesserungswürdig sind. Wegen der Quantisierung ergeben sich Fehler, die sich als Artefakte, das sind im Bild zu erkennende Blockstrukturen, äußern. Diese Blockstrukturen entstehen aufgrund der unabhängigen Bearbeitung der 8x8-Blöcke. Abbildung 9(b) und (c) zeigen Beispiele solcher Artefakte. Der Fehler und die Bildgröße, die beim Vorgeben der Skalierung der Quantisierungs-Tabelle entstehen, sind nicht abschätzbar. Der Anwender hat also keinen direkten Einfluss auf die Qualität oder den Kompressionsfaktor des Bildes.

Ein weiterer großer Nachteil von JPEG ist die Anfälligkeit gegenüber gestörten Datenverbindungen. Da ein einzelnes gestörtes Bit im vom Huffman-Code erzeugtem Datenstrom allen folgenden Bits eine andere Bedeutung gibt, kann ein einzelnes gestörtes Bit das gesamte Bild unbrauchbar machen. Abbildung 8 zeigt ein Originalbild und die Folgen eines falschen Bits an einer ungünstigen Stelle, nämlich einen Farbumschwung nach Blau.



(a) Originalbild (b) mit gestörtem Bit

Abbildung 8: Anfälligkeit des JPEG-Formats für Übertragungsfehler

4 JPEG2000

JPEG2000 ist ein neuer Standard zur Bildkompression, der von der Joint Photographic Experts Group (JPEG) als Verbesserung des weit verbreiteten JPEG Standards entwickelt wurde.

4.1 Motivation und Geschichte

Der JPEG-Standard wurde in der 80'ger Jahren entwickelt und verbreitete sich erfolgreich, da entsprechende effiziente Software von der Independent JPEG Group (IJG) entwickelt wurde und kostenfrei allen Anwendern zur Verfügung stand. Trotz dieses Erfolges wurden in den 90'ger Jahren auch die Grenzen von JPEG erkennbar, denn JPEG ist neuen Anforderungen wie hochauflösende Bilder, großen Datenmengen in digitalen Bibliotheken und ähnlichem nur bedingt gewachsen. Eine besonders auffällige Schwäche von JPEG ist hierbei das typische Auftreten von Block-Artefakten bei hoher Kompression, siehe Abbildung 9. [9]

(a) JPEG-Kompression mit 20% Qualität, 16857 Byte, ca. 0.3 Bit/Pixel

(b) JPEG-Kompression mit 5% Qualität, 5237 Byte, ca. 0.1 Bit/Pixel

(c) JPEG-Kompression mit 3% Qualität, 3449 Byte, ca. 0.07 Bit/Pixel

(d) Originalbild, 698x594 Pixel

Abbildung 9: JPEG-Kompression des Testbildes boats

Außerdem benutzt JPEG zwei völlig verschiedene Verfahren für verlustfreie und verlustbehaftete Kompression, was zu unnötig hohem Aufwand führt, wenn man zwischen den beiden Modi wechselt.

Dies führt zum Anforderungsprofil an den neuen Standard JPEG2000:

- JPEG2000 sollte eine bessere Bildqualität besonders auch bei hohen Kompressionsraten ermöglichen.
- Verlustbehaftete und verlustfreie Kompression und Dekompression sollen denselben Code verwenden, um die Flexibilität für verschiedene Anwendungen zu erhöhen. So kann es z.B. in einem Netzwerk sinnvoll sein ein Bild mit maximaler Qualität zu speichern, aber den Zugriff mit verminderter Qualität (also verlustbehaftet) zuzulassen.
- Progressive Übertragung (engl. progressive transmission) ermöglicht beim Laden eines Bildes, dass dieses sukzessiv mit zunehmender Qualität aufgebaut wird. Man sieht zunächst die Grundstruktur eines Bildes und mit der Zeit werden immer mehr Details ergänzt. Hat man die gewünschte Zielqualität erreicht, kann man gegebenenfalls den weiteren Aufbau des Bildes abbrechen.

Außerdem verfügt JPEG2000 zusätzlich noch über einige Funktionalitäten, die in speziellen Anwendungen sehr nützlich sind:

- Region of interest: JPEG2000 erlaubt es dem Anwender, sogenannte regions of interest zu definieren, also Teile eines Bildes die „interessanter“ sind als andere. Diese Regionen werden mit weniger Verlust codiert, so dass man das gewünschte Gebiet, z.B. ein Gesicht auf einem Foto, mit gewünschter Auflösung speichern kann, ohne das ganze Bild in dieser geringeren Kompressionsrate speichern zu müssen.
- Die Syntax des Datenstroms (engl. codestream) erleichtert das Erkennen von Fehlern, die z.B. durch fehlerhafte Übertragung von Bits entstehen. Dies ist besonders wichtig bei drahtloser Übertragung von Daten, insbesondere über größere Entfernungen. [5]
- Sequentieller Bildaufbau ermöglicht das Dekodieren eines Bildes, ohne das gesamte Bild zwischenzuspeichern, dies ist wichtig für das Laufzeitverhalten auf Geräten mit geringem Arbeitsspeicher. [15]

Der JPEG2000-Standard ist in 12 Teile gegliedert, Teil 1 [8] definiert den Kernkodierungsprozess. Die restlichen Teile sind Erweiterungen, wobei noch nicht alle Teile in endgültiger Fassung vorhanden sind und auch einige Teile wieder verworfen wurden. Teil 2 definiert z.B. allgemeine lineare Farbtransformation und Wavelet-Transformationen, die Korrelation zwischen Komponenten ausnutzt. Außerdem bietet Teil 2 größere Flexibilität bezüglich Level-Offsets und der Tot-Zone beim Quantisieren, um nur ein paar Beispiele zu nennen. Teil 3 behandelt Motion-Pictures, also bewegte Bilder.

4.2 Anwendungsgebiete

Auch wenn für Standard Anwendungen JPEG zur Zeit noch sehr viel verbreiteter ist als JPEG2000, so gibt es doch einige Spezialgebiete in denen JPEG2000 anfängt sich durchzusetzen:

- In der Medizin: der DICOM-Standard für medizinische Daten,
- In der Film Branche: die Digital Cinema Initiative nutzt Motion-JPEG2000,
- Behörden: die Passbilder auf den neuen deutschen Reisepässen,
- Digitalkameras,
- Raumfahrt und Satelliten Technik für zivile und militärische Anwendungen, z.B wurden Bilder des Marsrovers in einem JPEG2000 ähnlichem Format übertragen,
- Nationale Archive legen wichtige Bilddaten im JPEG2000-Format ab [4].

4.3 Übersicht

Das Kodieren eines Bildes geschieht in mehreren Schritten, siehe Abbildung 10. In der Vorverarbeitung wird das Bild in Blöcke unterteilt und unter Umständen eine Zentrierung um die Null (engl. DC level shifting) und eine Farbtransformation durchgeführt. Dann wird eine diskrete Wavelet-Transformation durchgeführt. Bei verlustbehafteter Kompression folgt nun ein Quantisierungsschritt und als letztes wird in zwei Schritten codiert, wobei der erste Schritt das arithmetische Codieren ist und der zweite der Organisation des Bitstromes dient.

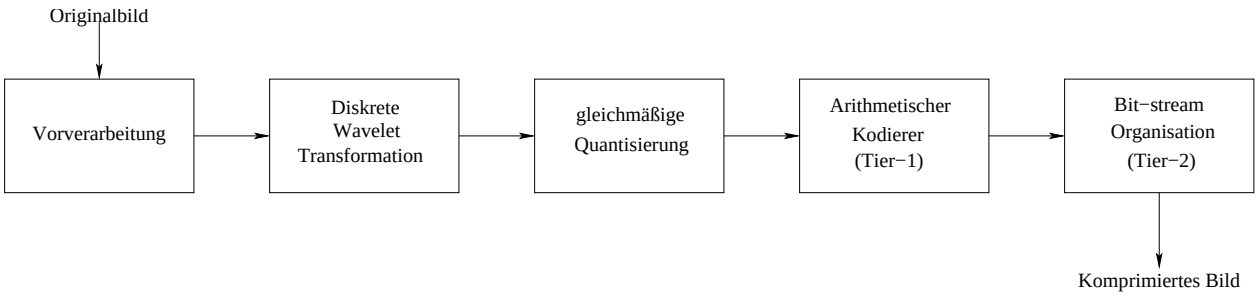


Abbildung 10: Ablaufsübersicht der Kodierung mit JPEG2000

4.4 Vorverarbeitung

Um auch mit begrenzten Ressourcen großformatige Bilder komprimieren zu können, wird das Bild zunächst in Teilbilder aufgeteilt. Diese werden wie einzelne Bilder behandelt. Definiert man eine region of interest, wird diese als unabhängiges Teilbild behandelt und kann somit mit besserer Qualität komprimiert werden. [14]

In dem Fall, dass das ursprüngliche Bild ein Vorzeichen freies Bild ist, wird wie beim JPEG-Standard eine Verschiebung (level shifting) durchgeführt, so dass die Mitte des Bildtiefe bei Null liegt.

Bei Bildern mit drei Farbkomponenten, bei denen jede Komponente dieselbe Bittiefe hat, kann nun eine Farbtransformation durchgeführt werden, um die Redundanz zwischen den Komponenten auszunutzen. JPEG2000 erlaubt zwei Arten von Farbtransformationen, eine irreversible (engl. irreversible colour transformation, ICT) und eine reversible Variante (engl. reversible colour transformation, RCT).

4.4.1 Nicht invertierbare Farbtransformation

Bei der ICT werden die drei Komponenten R,G und B abgebildet auf drei neue Komponenten Y, C_b und C_r [15], wie bei JPEG wird also mit Luminanz und Chrominanz gearbeitet. Die Transformations berechnet sich über

$$\begin{aligned} Y &= \alpha_R \cdot R + \alpha_G \cdot G + \alpha_B \cdot B, \\ C_b &= \frac{0.5}{1-\alpha_B}(B-Y), \\ C_r &= \frac{0.5}{1-\alpha_R}(R-Y) \end{aligned}$$

mit Gewichten

$$\alpha_R = 0.299, \quad \alpha_G = 0.587, \quad \alpha_B = 0.114.$$

4.4.2 Invertierbare Farbtransformation

Die reversible Variante der Farbtransformation ist für uns weniger interessant, da sie nur bei der verlustfreien Kompression verwendet wird. Im Gegensatz zu der ICT arbeitet die RCT mit ganzen Zahlen. Sie ist laut [15] definiert über

$$\begin{aligned} Y &= \left\lfloor \frac{R + 2G + B}{4} \right\rfloor, \\ C_b &= B - G, \\ C_r &= R - G. \end{aligned}$$

Das Inverse dieser Operation, also die Rücktransformation vom Farbraum YCbCr zum Farbraum RGB, ergibt sich über

$$\begin{aligned} G &= Y - \left\lfloor \frac{C_r + C_b}{4} \right\rfloor, \\ B &= C_b + G, \\ R &= C_r + G. \end{aligned}$$

4.5 Wavelet-Transformation

Die Kosinus-Transformation, die JPEG2000 benutzt, ist keine optimale Wahl zu Kompression von Bildern, denn eine Kosinus-Transformation interpretiert ein Signal als Überlagerung periodischer Signale verschiedener Frequenzen, ein Bild ist im Allgemeinen jedoch nicht periodisch. Daher ist die Kosinus-Transformation in JPEG auf kleine Teile eines Bildes beschränkt, was zu oben genannten Block-Artefakten führt. JPEG2000 benutzt aus diesem Grund keine Kosinus-Transformation, sondern eine auf Wavelets basierende Transformation, da diese die nicht-periodischen Signale, aus denen ein Bild besteht, besser annähern können.

4.5.1 Wahl der Wavelets

Die Wahl der Wavelets bleibt dabei nicht dem Zufall überlassen. Bei der Entscheidung welche Wavelets günstig für diese Anwendung sind, spielen hauptsächlich drei Faktoren eine Rolle: die Anzahl der verschwindenden Momente, die Größe des Trägers und die Regularität. Große Waveletkoeffizienten entstehen bei plötzlichen Unterschieden im Bild, also bei Kanten. Je mehr Wavelets sich an solchen Stellen überlappen, desto mehr Wavelets mit großen Koeffizienten erhält man, deswegen ist eine geringe Träger Größe wünschenswert.

Auf der anderen Seite möchte man möglichst viele verschwindende Momente haben, da diese Regelmäßigkeiten bei sich wenig verändernden Teilen des Bildes ausnutzen können, also kleine Koeffizienten zur Folge haben. Da jedoch die Anzahl der verschwindenden Momente proportional zur Trägergröße ist, wählt man einen Kompromiss zwischen diesen Eigenschaften.

Die Regularität der Wavelets ist wichtig, um die Artefakte zu verhindern, die bei JPEG so negativ auffallen. Bei irregulären Wavelets (z.B. Haar-Wavelets) sehen die Fehler, die man bei der Quantisierung macht, wie Kanten aus, wodurch die Artefakte deutlich sichtbarer sind als bei regulären Wavelets. Es hat sich jedoch gezeigt, dass mehr Regularität als stetige Differenzierbarkeit kaum noch sichtbaren Unterschied macht. [10]

Um die Ränder eines Bildes sinnvoll ergänzen zu können, ist es von Vorteil symmetrische oder antisymmetrische Wavelets zu wählen. Man hätte also gerne stetig differenzierbare symmetrische oder antisymmetrische Wavelets mit kompakten Träger und möglichst vielen verschwindenden Momenten. Diesem Ziel kommt man am nächsten, indem man auf die Orthogonalität der konstruierten Basis verzichtet und stattdessen biorthogonale Wavelets benutzt. [10]
Der JPEG2000 Standard entscheidet sich bei der verlustbehafteten Kompression für die Cohen-Daubechies-Feauveau 9/7-Wavelets (CDF 9/7) [6]. Diese sind biorthogonale symmetrische Wavelets mit vier verschwindenden Momenten und haben sich in der Praxis als gelungener Kompromiss

n	h[n]	g[n]
0	0.602949018236	0.557543526229
1,-1	0.266864118443	-0.295635881557
2,-2	-0.078223266529	-0.028771763114
3,-3	-0.016864118443	0.045635881557
4,-4	0.026748757411	

Tabelle 5: Highpass-Filter g, Lowpass-Filter h zu CDF 9/7-Wavelet [15, Seite 433]

zwischen den gewünschten Eigenschaften herausgestellt. Die Bezeichnung 9/7 bezieht sich auf die Länge der dazugehörigen Filter, siehe Tabelle 5.

Bei der verlustfreien Kompression hingegen wird eine modifizierte Variante der Wavelet-Transformation mit dem Cohen-Daubechies-Feauveau 5/3-Wavelet genutzt. Die Implementierung geschieht mit Hilfe eines Lifting-Schemas, das die ganzzahligen Signalwerte auf ganzzahlige Detail- und Approximationskoeffizienten abbildet. Auf dieses Schema wird an hier nicht näher eingegangen, Näheres findet sich in [8, 14].

4.5.2 Diskrete Wavelet-Transformation

In einem vorhergegangenem Vortrag [17] haben wir bereits die schnelle biorthogonale Wavelet-Transformation kennengelernt. Wir wiederholen kurz die Grundlagen: Sei das Eingangssignal $a_L[n] = \langle f, \phi_{L,n} \rangle \in V_L$, dann werden $a_j[n] = \langle f, \phi_{j,n} \rangle$ und $d_j[n] = \langle f, \psi_{j,n} \rangle$ sukzessiv berechnet über

$$a_{j+1}[n] = (a_j * \bar{h})[2n] \in V_{j+1},$$
$$d_{j+1}[n] = (a_j * \bar{g})[2n] \in W_{j+1}.$$

Dabei sind $\bar{h}$ und $\bar{g}$ die zugehörigen Filter: $\bar{h}$ ist der Lowpass-Filter, deswegen gibt a_{j+1} die Grundstruktur des Signals wieder, während $\bar{g}$ ein Highpass-Filter ist und d_{j+1} somit die Details enthält. Mit Hilfe der dualen Filter $\tilde{h}$ und $\tilde{g}$ erhält man auch die Rekonstruktion

$$a_j[n] = (\tilde{a}_{j+1} * \tilde{h})[n] + (\tilde{d}_{j+1} * \tilde{g})[n].$$

$\tilde{a}$ bzw. $\tilde{d}$ bezeichnen a bzw. d mit aufgefüllten Nullen [17]. Dieses Wissen benutzen wie nun als Grundlage für die 2-dimensionale diskrete Wavelet-Transformation (2D-DWT), die in JPEG2000 zur Anwendung kommt.

4.5.3 2D-Wavelet-Transformation

Eine orthonormale Wavelet-Basis von $L^2(\mathbb{R}^2)$ wird als separables Produkt der Skalierungsfunktion ϕ und dem Wavelet ψ konstruiert. Die Skalierungsfunktion ϕ gehört zu der eindimensionalen multiresulution Analysis $\{V_j\}_{j\in\mathbb{Z}}$. Sei $\left\{V_j^2\right\}_{j\in\mathbb{Z}}$ die separable zweidimensionale multiresolution Analysis und definiert als $V_j^2 = V_j \otimes V_j$. Außerdem sei W_j^2 der Detailraum, der das Komplement des Raums der geringeren Auflösung V_j^2 in V_{j-1}^2 ist

$$V_{j-1}^2 = V_j^2 \oplus W_j^2. \tag{4}$$

Der folgende Satz konstruiert eine Wavelet-Basis des Detailraums W_j^2, um eine orthonormale Wavelet-Basis von $L^2(\mathbb{R}^2)$ zu konstruieren.

Satz 1. *Sei ϕ eine Skalierungsfunktion und ψ das dazugehörige Wavelet, das eine orthonormale Basis von $L^2(\mathbb{R})$ generiert. Wir definieren drei Wavelets*

$$\psi^1(x) = \phi(x_1)\psi(x_2), \quad \psi^2(x) = \psi(x_1)\phi(x_2), \quad \psi^3(x) = \psi(x_1)\psi(x_2)$$

und für $1 \leq k \leq 3$ *die Skalierungen*

$$\psi_{j,n}^k(x) = \frac{1}{2^j}\psi^k(\frac{x_1 - 2^j n_1}{2^j}, \frac{x_2 - 2^j n_2}{2^j}).$$

Die Wavelet-Familie

$$\left\{\psi_{j,n}^1, \psi_{j,n}^2, \psi_{j,n}^3\right\}_{n\in\mathbb{Z}^2}$$

ist eine orthonormale Basis von W_j^2 *und*

$$\left\{\psi_{j,n}^1, \psi_{j,n}^2, \psi_{j,n}^3\right\}_{(j,n)\in\mathbb{Z}^3}$$

ist eine orthonormale Basis von $L^2(\mathbb{R}^2)$.

Beweis. [10] Man kann (4) umschreiben zu

$$V_{j-1} \otimes V_{j-1} = (V_j \otimes V_j) \oplus W_j^2. \tag{5}$$

Den eindimensionalen Raum V_{j-1} kann man zerlegen in $V_{j-1} = V_j \oplus W_j$. Setzt man dies in (5), so erhält man mit der Distributivität der direkten Summe

$$W_j^2 = (V_j \otimes W_j) \oplus (W_j \otimes V_j) \oplus (W_j \otimes W_j).$$

Da $\{\phi_{j,m}\}_{m\in\mathbb{Z}}$ und $\{\psi_{j,m}\}_{m\in\mathbb{Z}}$ orthonormale Basen von V_j und W_j sind, leiten wir ab, dass

$$\{\phi_{j,n_1}(x_1)\psi_{j,n_2}(x_2), \psi_{j,n_1}(x_1)\phi_{j,n_2}(x_2), \psi_{j,n_1}(x_1)\psi_{j,n_2}(x_2)\}_{(j,n_1,n_2)\in\mathbb{Z}^3}$$

eine orthonormale Basis von W_j^2 ist. Analog zum eindimensionalen Fall, kann man $L^2(\mathbb{R}^2)$ zerlegen in eine orthogonale Summe der Detailräume aller Auflösungen

$$L^2(\mathbb{R}^2) = \oplus_{j=-\infty}^{+\infty} W_j^2.$$

Daher ist

$$\{\phi_{j,n_1}(x_1)\psi_{j,n_2}(x_2), \psi_{j,n_1}(x_1)\phi_{j,n_2}(x_2), \psi_{j,n_1}(x_1)\psi_{j,n_2}(x_2)\}_{(j,n_1,n_2)\in\mathbb{Z}^3}$$

eine orthonormale Basis von $L^2(\mathbb{R}^2)$. □

Analog kann man aus einer eindimensionalen biorthogonalen Wavelet-Basis eine separable biorthogonale Basis von $L^2(\mathbb{R}^2)$ konstruieren. Seien dazu ϕ, ψ und $\tilde{\phi}$, $\tilde{\psi}$ zwei duale Paare von Skalierungsfunktionen und Wavelets, die eine biorthogonale Wavelet-Basis von $L^2(\mathbb{R})$ generieren. Definiert man ψ^1,ψ^2 und ψ^3 wie in Satz 1 und entsprechend

$$\tilde{\psi}^1(x) = \tilde{\phi}(x_1)\tilde{\psi}(x_2), \quad \tilde{\psi}^2(x) = \tilde{\psi}(x_1)\tilde{\phi}(x_2), \quad \tilde{\psi}^3(x) = \tilde{\psi}(x_1)\tilde{\psi}(x_2),$$

dann sind

$$\left\{\psi_{j,n}^1, \psi_{j,n}^2, \psi_{j,n}^3\right\}_{(j,n)\in\mathbb{Z}^3}$$

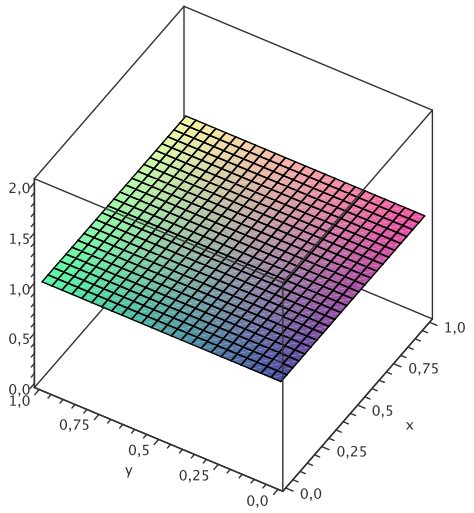
und

$$\left\{\tilde{\psi}_{j,n}^1, \tilde{\psi}_{j,n}^2, \tilde{\psi}_{j,n}^3\right\}_{(j,n)\in\mathbb{Z}^3}$$

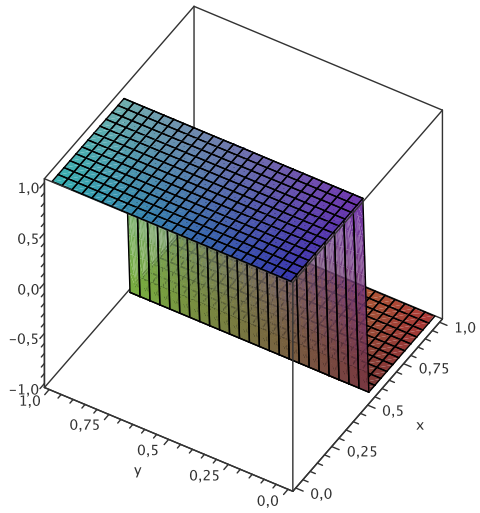
biorthogonale Riesz-Basen von $L^2(\mathbb{R})$. [10] Abbildung 11 zeigt die 2-dimensionalen Wavelets und Skalierungsfunktionen am Beispiel der Haar-Wavelets.

Die schnelle biorthogonale Wavelet-Transformation wird nun zu einer schnellen zweidimensionalen Wavelet-Transformation erweitert. Dazu seien (h, g) und $(\bar{h}, \bar{g})$ die Filter Paare zum Wavelet ψ bzw. $\tilde{\psi}$. Außerdem bezeichne

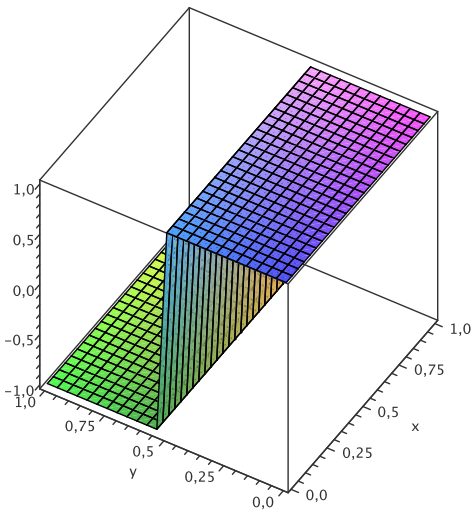
$$a_j[n] = \left\langle f, \phi_{j,n}^2 \right\rangle \text{ und } d_j^k[n] = \left\langle f, \psi_{j,n}^k \right\rangle \text{ für } 1 \leq k \leq 3 \tag{6}$$



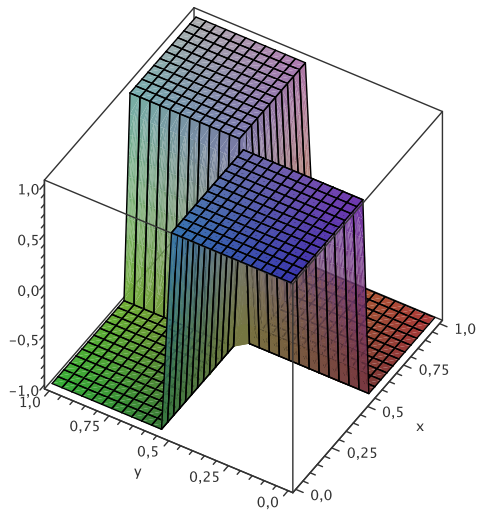
(a) Skalierungsfunktion $\phi(x)\phi(y)$



(b) Wavelet $\psi(x)\phi(y)$

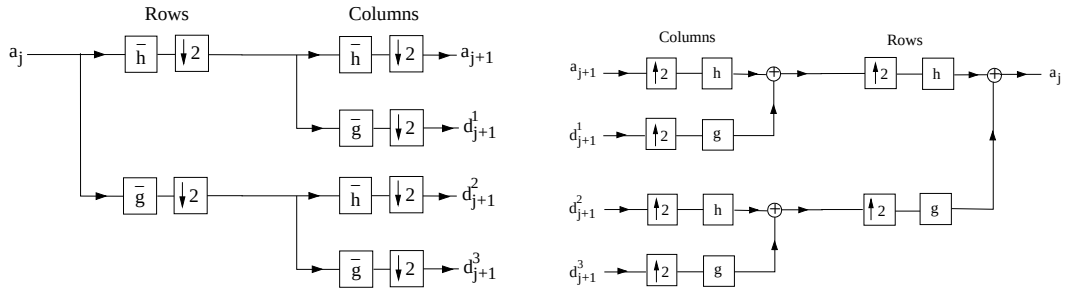


(c) Wavelet $\phi(x)\psi(y)$



(d) Wavelet $\psi(x)\psi(y)$

Abbildung 11: 2D-Tensorkonstruktion für das Haar-Wavelet



(a) diskrete (b) diskrete inverse

Abbildung 12: 2 dimensionale Wavelet-Transformation

für alle $n = (n_1, n_2)$ und jede Auflösung 2^j. Dann erhält man die Waveletkoeffizienten der Auflösung 2^{j+1} über 2-dimensionale Faltung von a_j

$$\begin{aligned} a_{j+1}[n] &= a_j * \bar{h}\bar{h}[2n], \\ d^1_{j+1}[n] &= a_j * \bar{h}\bar{g}[2n], \\ d^2_{j+1}[n] &= a_j * \bar{g}\bar{h}[2n], \\ d^3_{j+1}[n] &= a_j * \bar{g}\bar{g}[2n]. \end{aligned}$$

Die 2D-DWT kann man also in eindimensionale Faltungen entlang der Zeilen und Spalten des Bildes faktorisieren. Wie Abbildung 12(a) veranschaulicht, wird zuerst zeilenweise einmal der Lowpass-Filter $\bar{h}$ und einmal der Highpass-Filter $\bar{g}$ angewendet und jeweils jeder zweite Wert gelöscht (engl. downsampling). Anschließend werden auf beiden Teilbildern spaltenweise die Filter ausgeführt und in jeder Spalte jeder zweite Wert gelöscht.

Abbildung 13 zeigt die Struktur eines Bildes einer 2D-DWT mit Stufen. Die einzelnen Blöcke, die man sehen kann, werden Subbänder genannt (engl. subband). Der Index zeigt, zu welcher Stufe das jeweilige Subband gehört. Subband LL_4 enthält eine grobe Darstellung des Bildes. HL-Subbänder zeigen Details in horizontaler Richtung, da sie das Bild zeigen, das zeilenweise durch den Highpass-Filter und spaltenweise durch den Lowpass-Filter transformiert wurde. Entsprechend enthalten die LH-Subbänder Details in vertikaler und HH-Subbänder Details in diagonaler Richtung. Für die Rekonstruktion der Auflösung 2^j gilt

$$a_j[n] = \check{a}_{j+1} * hh[n] + \check{d}^1_{j+1} * hg[n] + \check{d}^2_{j+1} * gh[n] + \check{d}^3_{j+1} * gg[n],$$

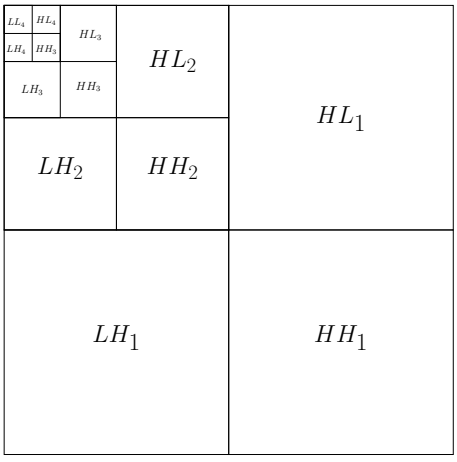
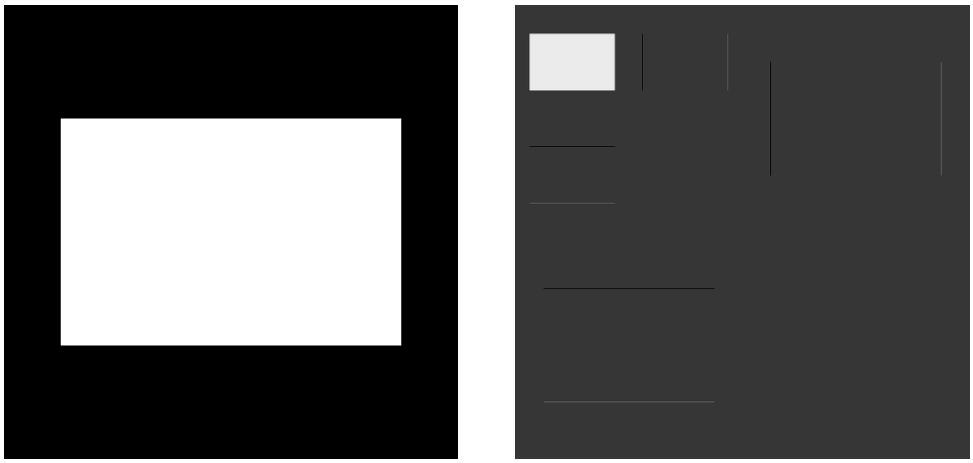


Abbildung 13: Subbänder bei 4 stufiger 2D-Wavelet-Transformation



(a) Originalbild, 512x512 Pixel (b) Wavelet-Transformation, 2 Level

Abbildung 14: Wavelet-Transformation des Testbildes box, CDF9/7

(a) Originalbild, 698x594 Pixel (b) Wavelet-Transformation, 2 Level

Abbildung 15: Wavelet-Transformation des Testbildes boats, CDF9/7

$\check{\cdot}$ bezeichne dabei, dass die Daten mit Nullen aufgefüllt werden, da diese bei der schnellen Wavelet-Transformation gesubsampelt wurden. Abbildung 12(b) veranschaulicht die Rekonstruktion. Die Abbildungen 14 und 15 zeigen die Wavelet-Zerlegung von Beispielbildern.

4.6 Quantisierung

Bei der Quantisierung wird jeder Waveletkoeffizient auf einen Quantisierungsindex abgebildet, die Abbildungsvorschrift lautet

$$q = \operatorname{sgn} \left\lfloor \frac{|y|}{\Delta_b} \right\rfloor .$$

Dabei bezeichnet y den Waveletkoeffizienten und q den resultierenden Quantisierungsindex. Δ_b bezeichnet die Quantisierungsschrittweite, die vom Benutzter gewählt werden kann und für die Quantisierung der Koeffizienten in einem Subband genutzt wird. Bei der Quantisierung wird also der Wertebereich der Waveletkoeffizienten in Intervalle unterteilt. Das zentrale Intervall um Null hat die Breite $2\Delta_b$ und wird die Tot-Zone genannt, da alle Waveletkoeffizienten, die in diesem Intervall liegen, auf den Quantisierungsindex 0 abgebildet werden. Alle übrigen Intervalle haben die Breite Δ_b, wie in Abbildung 16 zu sehen ist. Es handelt sich also um einen gleichmäßigen Kodierer mit Tot-Zone (engl. uniform quantizer with deadzone).

Da man bei der Quantisierung Waveletkoeffizienten aus einem ganzen Intervall auf denselben Index abbildet, ist es offensichtlich, dass diese Operation nicht invertierbar ist und somit auch nur bei der verlustbehafteten Bildkompression eine Rolle spielt.

Bei der Dekompression wird also nicht der Quantisierungsindex auf den ursprünglichen Koeffizienten abgebildet, statt dessen benutzt man

$$z = \begin{cases} (q + r\,\mathrm{sgn}(q))\Delta_b, & \text{für} \quad q \neq 0, \\ 0, & \text{für} \quad q = 0. \end{cases}$$

Der Parameter $r \in [0, 1]$ kann dabei frei gewählt werden. Für $r = 0.5$ wird z.B. jeder Index auf die Mitte des ursprünglichen Intervalls abgebildet. [12]

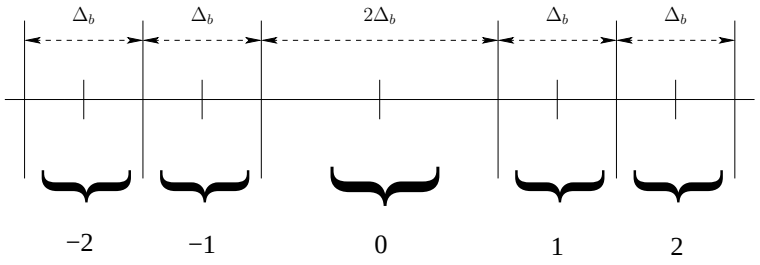


Abbildung 16: Gleichmäßige Kodierung mit Quantisierungskoeffizient Δ_b

4.7 Kodierung

4.7.1 Tier-1

Vor dem eigentlichen Kodieren werden die Subbänder in Code-Blöcke unterteilt, die unabhängig voneinander kodiert werden. Die Höhe und Breite dieser Code-Blöcke sind Zweierpotenzen, bis auf die Code-Blöcke an den Rändern der Subbänder, da diese eventuell etwas kleiner sind, als die restlichen Code-Blöcke eines Subbandes.

Im Gegensatz zu der Kodierung in JPEG wird ein quantisierter Koeffizient in JPEG2000 nicht als ein einzelnes Symbol kodiert. Stattdessen arbeitet der Kodierer in JPEG2000 bitweise. Das bedeutet, dass der Kodierer die Code-Blöcke in Bitebenen (engl. bit-plane, BP) aufteilt. Jeder zu kodierende Quantisierungsindex wird mit derselben Anzahl von Bits dargestellt. Diese Anzahl ist die Anzahl der Bitebenen. Der Kodierer kodiert zuerst die Bitebene, die die höchstwertigen Bits (engl. most significant bit, MSB) enthält und arbeitet sich durch die Bitebenen zu der BP, die die niederwertigsten Bits enthält (least significant bit, LSB).

Jede Bitebene wird in einer festgelegten Reihenfolge durchlaufen. Der Kodierer startet mit den ersten vier Bits der ersten Spalte, gefolgt von den ersten vier Bits der zweiten Spalte bis die Zeile zu Ende ist. Dann folgen die nächsten vier Bits der ersten Spalte usw. bis zum Ende des Code-Blockes. Ist die Anzahl der Zeilen kein Vielfaches von vier, werden beim Durchlaufen der untersten Zeilen nur weniger als vier Bits genommen.
Das Vorzeichen eines Quantisierungsindex wird nicht als eigene Bitebene behandelt, sondern wird kodiert, wenn der erste Eintrag eines Quantisierungsindex, der ungleich Null ist, kodiert wird. Jeder Koeffizient wird zunächst als insignifikant bezeichnet. Wird das erste Bit ungleich Null eines Index kodiert, so wird dieser signifikant.

Das Kodieren einer Bitebene geschieht in drei Phasen, der Signifikanz-Phase (engl. significance propagation pass), der Verfeinerungs-Phase (engl. magnitude refinement pass) und der Cleanup-Phase (engl. cleanup pass). Die Signifikanz-Phase ist der erste Durchlauf, in dieser Phase werden die Bits kodiert, die selbst insignifikant sind, jedoch mindestens einen signifikanten Nachbarn haben, da bei diesen Symbolen, die Wahrscheinlichkeit hoch ist, dass sie in diesem Schritt signifikant werden. Nachbarn heißen in diesem Zusammenhang die acht vertikal, horizontal bzw. diagonal angrenzenden

Bits innerhalb der jeweiligen Bitebene. Es gibt 256 mögliche Kombinationen der Signifikanzen dieser acht Nachbarn. Diese Möglichkeiten werden in 9 verschiedene Kategorien eingeteilt, die jeweilige Kategorie nennt sich Kontextkennung des Bits. Generell hängt die Einteilung der Kontextkennungen von der Implementierung ab, ein Vorschlag findet sich in [8, Seite 86]. Im Allgemeinen bedeutet eine Kontextkennung von „0“, dass keiner der Nachbarn signifikant ist und somit das Bit für die Cleanup-Phase übrig bleibt.

Das aktuelle Bit des Koeffizienten wird in Abhängigkeit der Kontextkennung mit einem arithmetischen Kodierer, dem MQ-Coder, verarbeitet. Eine genaue Beschreibung des MQ-Codes findet sich in [15]. Ist das zu kodierende Bit '1', so wird das Vorzeichen des Koeffizienten in Abhängigkeit der Vorzeichen der vertikal und horizontal angrenzenden Koeffizienten kodiert und der Koeffizient erhält den Status signifikant. Danach geht der Kodierer zum nächsten Bit über.

Nach der Signifikanz-Phase folgt die Verfeinerungs-Phase, in der alle signifikanten Bits kodiert werden. In dieser Phase müssen keine Vorzeichen mehr kodiert werden. Der Signifikanz-Status der vertikalen und horizontalen Nachbarn wird nur für das Kodieren des ersten Verfeinerungsbits berücksichtigt, also dem ersten Bit eines Koeffizienten, das in der Verfeinerungs-Phase kodiert wird.

Als letztes folgt die Cleanup-Phase, die alle übrigen Bits kodiert. Dabei kommt auch eine Lauflängenkodierung zum Einsatz, da die Bits, die in dieser Phase kodiert werden mit relativ hoher Wahrscheinlichkeit gleich Null sind. Bei der Lauflängenkodierung betrachtet der Kodierer immer die vier Bits einer Spalte. Sind diese noch alle zu kodieren und haben alle den Kontext Null, was nicht zwangsweise eintreten muss, da sich der Status von Koeffizienten in der Signifikanz-Phase geändert haben kann, so steht ein 0-Bit, dafür, dass alle vier Bits insignifikant bleiben. Ein 1-Bit bedeutet, dass sich mindestens der Status eines Bits ändert. In diesem Fall benötigt der Kodierer zwei weitere Bits, um die Spaltenposition des ersten Koeffizienten, dessen Status sich ändert, zu kodieren. Anschließend wird das Vorzeichen des Koeffizienten kodiert. Die restlichen Koeffizienten werden wie in der Signifikanz-Phase kodiert.

Jedes Bit wird also in genau einer der drei Phasen kodiert. Bei der ersten Bitebene eines Code-Blocks wird nur eine Cleanup-Phase durchlaufen, da in dieser Bitebene noch alle Bits als insignifikant markiert sind und somit weder die Signifikanz-Phase noch die Verfeinerungs-Phase Bits zu kodieren hat.[8]

4.7.2 Tier-2

Das Tier-1-Kodieren liefert eine Ansammlung von Bitstreams, ein Bitstream für jede Bitebene jedes Code-Blocks. Das Tier-2-Kodieren fügt diese einzelnen Bitstreams zu einem gesamten Codestream zusammen. JPEG2000 unterstützt das progressive Dekomprimieren, dabei wird mit den ersten Daten im Codestream ein Bild mit zunächst schlechter Qualität erzeugt und im Laufe der Dekompression wird die Auflösung das Bildes besser, da immer mehr Details hinzukommen. Dabei kennt der Standard mehrere Wege, ein Bild progressiv aufzubauen, abhängig davon in welcher Reihenfolge die Bitstreams im Codestream enthalten sind. Welche Reihenfolge jeweils vorliegt, wird in Header das Bildes abgespeichert. Grundsätzlich unterscheidet man fünf Modi:

- Ebene-Auflösung-Komponente-Position: Die Bitebenen werden nacheinander durchlaufen, innerhalb der Bitebenen alle Auflösungen (Subbänder der Wavelet-Transformation) und innerhalb der Subbänder alle Komponenten (entfällt bei einem ein komponentigem Graubild).

- Auflösung-Ebene-Komponente-Position: Bei diesem Modus werden alle Auflösungen nacheinander abgearbeitet, darin alle Bitebenen, darin wiederum nach Komponenten geordnet und zuletzt nach der Position.

Die weiteren Modi Auflösung-Position-Komponente-Ebene, Position-Komponente-Auflösung-Ebene und Komponente-Position-Auflösung-Ebene funktionieren analog.

Der Codestream alleine enthält im Prinzip alle Informationen, die das Bild selbst enthält. Damit ein Programm das Bild auch dekomprimieren kann, wird der Codestream in ein Dateiformat eingebettet.

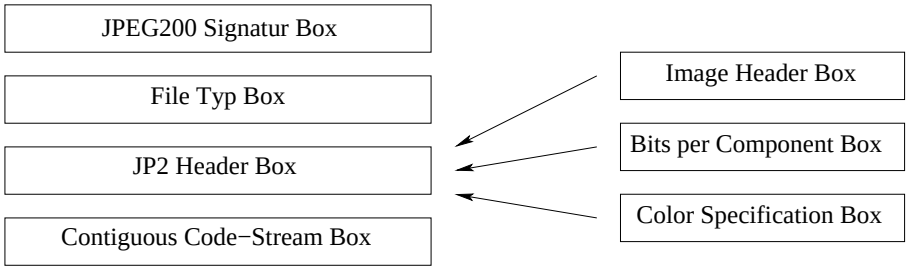


Abbildung 17: Aufbau einer jp2 Datei

4.8 Dateiformat

Ein Dateiformat, das typischerweise für mit JPEG2000 komprimierte Dateien genutzt wird, ist jp2. Es besteht aus mehreren sogenannten Boxen, die hintereinander in der Datei stehen, teilweise jedoch auch ineinander geschachtelt sind. Dabei gibt es Boxen, die in jeder jp2-Datei vorhanden sein müssen und optionale Boxen. Wir beschränken uns hier auf die benötigten Boxen. Wie in Abbildung 17 ersichtlich ist, ist die erste Box die JPEG2000 Signature Box, direkt gefolgt von der File Type Box. Danach können zwischen den benötigten Boxen auch optionale Boxen auftauchen, es folgen aber auf jeden Fall noch die JP2 Header Box und die Codestream Box in dieser Reihenfolge.

Der Grundaufbau aller Boxen ist derselbe. Sie besteht aus den Parametern L, T, XL und C, vergleiche Abbildung 18. L ist vier Byte groß und gibt die Gesamtlänge der Box an. Ist L=1, so gibt L nicht die Länge an, da diese zu groß ist für vier Byte. Stattdessen gibt es in diesem Fall den Parameter XL, der diese Aufgabe übernimmt und acht Byte zur Verfügung hat. Für die letzte Box einer Datei ist außerdem L=0 erlaubt. Das bedeutet, die Länge war zur Laufzeit nicht bekannt, der Rest der Datei wird in diesem Fall zu dieser Box gewertet. Der Parameter T gibt den Typ der Box an. T ist immer ein fest vorgeschriebenes vier Byte großes Zeichen, die JPEG2000 Signature Box hat z.B. $T =' jp' = 6A502020_h$. C kann als der Inhalt der Box bezeichnet werden, der vom Typ der Box abhängt. Bei der Signature Box enthält C z.B. fest vorgegebene vier Byte, die dem Zweck dienen, Dateiübertragungsfehler zu erkennen. Die File Type Box enthält Angaben über das genaue Dateiformat und eine Liste mit kompatiblen Formaten. Die JP2 Header Box enthält wiederum selbst Boxen, vergleiche Abbildung 17. Diese enthalten nun z.B. Informationen über Höhe und Breite, Anzahl und Tiefe der einzelnen Komponenten und den Farbraum des Bildes. [15]

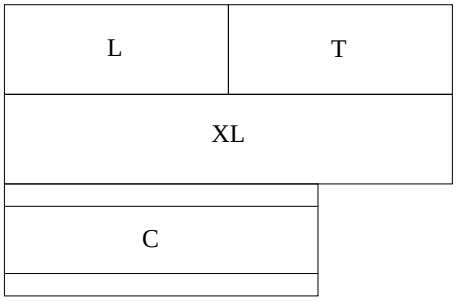


Abbildung 18: Aufbau einer Box in jp2

4.9 Ergebnisse

Abbildung 19 zeigt ein Beispielbild, das mit JPEG2000 bei verschiedenen Kompressionsraten komprimiert wurde. Vergleicht man diese mit demselben Beispielbild in Abbildung 9, das mit JPEG zu ähnlichen Dateigrößen komprimiert wurde, so erkennt man, dass JPEG2000 eine deutlich bessere

(a) JPEG2000-Kompression mit 0.3 Bit/Pixel

(b) JPEG2000-Kompression mit 0.1 Bit/Pixel

(c) JPEG2000-Kompression mit 0.07 Bit/Pixel

(d) Originalbild, 698x594 Pixel

Abbildung 19: JPEG2000-Kompression des Testbildes boats

Qualität liefert. Abbildung 20 zeigt ein Beispiel, bei dem JPEG2000 wegen der scharfen Kanten keine optimales Ergebnis liefert.

Abbildung 21 zeigt ein Beispiel für Kodierung mit region of interest, man erkennt deutlich die bessere Qualität in diesem Bereich des Bildes.

(a) JPEG2000-Kompression mit 0.7 Bit/Pixel (b) Originalbild, 64x64 Pixel

Abbildung 20: JPEG2000-Kompression des Testbildes schach

(a) ohne region of interest mit 0.375 Bit/Pixel

(b) mit region of interest mit 0.375 Bit/Pixel

(c) Originalbild, 698x594 Pixel

Abbildung 21: JPEG2000-Kompression des Testbildes boats

Literatur

[1] *JJ2000: An Implementation of the JPEG2000 standard in Java, cooperative project of Canon Research Centre France (CRF), the Swiss Federal Institute of Technology (EPFL) and ERICSSON Research.* `http://jj2000.epfl.ch`.

[2] *Set of classical test still images.* `http://www.hlevkin.com/TestImages/classic.htm`.

[3] *Wikipedia-Artikel zum Thema JPEG.* `http://de.wikipedia.org/wiki/JPEG`.

[4] *Wikipedia-Artikel zum Thema JPEG2000.* `http://de.wikipedia.org/wiki/JPEG2000`.

[5] Christopoulos, Charilaos, Athanassios Skodras und Touradj Ebrahimi: *The JPEG2000 Still Image Coding System: An Overview.* In: *IEEE Transactions on Consumer Electronics*, November 2000.

[6] Cohen, Albert, Ingrid Daubechies und Jean-Christophe Feauveau: *Biorthogonal bases of compactly supported wavelets.* Commun. Pure Appl. Math., 1992.

[7] ISO/IEC: *International Standard ISO/IEC 10918-1, Digital Compression and Coding of Continuous-Tone Still Images: Requirements and guidelines.*

[8] ISO/IEC: *International Standard ISO/IEC 15444-1, Information Technology-JPEG 2000 image coding system: Core coding system*, 2004.

[9] Lee, Daniel T.: *JPEG2000: Retrospective and New Developments.* In: *Proceedings of the IEEE*, Band 93, Seiten 32–41, Januar 2005.

[10] Mallat, Stéphane: *A Wavelet Tour of Signal Processing.* Academic Press, New York, 2. Auflage, 1999.

[11] Pawlitta, Nadine: *Datenkompression. Seminar zur Signalanalyse und Wavelets. Institut für Geometrie und Praktische Mathematik, Wintersemester 08/09, RWTH Aachen.*

[12] Rabbani, Majid und Diego Santa Cruz: *The JPEG2000 Still-Image Compression Standard.* `http://jj2000.epfl.ch/jj-puplications/papers/011.pdf`.

[13] Sayood, Khalid: *Introduction to Data Compression.* Morgan Kaufmann, San Diego, 2. Auflage, 2000.

[14] Strutz, Tilo: *Bilddatenkompression Grundlagen, Codierung, MPEG, JPEG.* Vieweg, 2000.

[15] Taubman, David S. und Michael W. Marcellin: *JPEG2000 Image Compession Fundamentals, Standards and Practice.* Kluwer Academic Publishers, Norwell, 2002.

[16] Tianhui, Wang: *CDF 9/7 Wavelet Transform, MATLAB code.* `http://www.mathworks.com/matlabcentral/fileexchange/11846`.

[17] Tieves, Martin: *Multi-Skalen-Analyse. Seminar zur Signalanalyse und Wavelets. Institut für Geometrie und Praktische Mathematik, Wintersemester 08/09, RWTH Aachen.*