JPEG

Seminar: Kompressionsalgorithmen

Ruslan Ragimov

5. September 2012

Zusammenfassung

Die allgemeinen verlustfreien Verfahren zur Datenkompression können gute Kompressionsraten für verschiedene Dateitypen liefern. Allerdings ist die verlustfreie Komprimierung nicht immer besonders gut für die digitalen Bilder geeignet, denn die Farbinformation bleibt vom Pixel zum Pixel in seltenen Fällen gleich. Es können jedoch auch nicht beliebige Informationen eines Bildes bei verlustbehafteter Kompression weggelassen werden. In dieser Arbeit wird die Komprimierungsstrategie JPEG, die ausschließlich für digitale Bilder entwickelt wurde und sowohl eine verlustfreie als auch verlustbehaftete Methode zur Bilderkompression unterstützt, betrachtet.

1 Einleitung

Damit die Kompression von digitalen Bildern betrachtet werden kann, sollen zunächst einige Grundlagen über Bilder erläutert werden. In diesem Kapitel wird auf Bilder, deren Datentyp und Farbräume eingegangen.

1.1 Digitalbild

Ein Digitalbild besteht aus einer Matrix von Pixeln, wobei ein Pixel die kleinste Einheit in dem Bild darstellt. Die Größe dieser Matrix wird als Auflösung bezeichnet. Dabei wird meistens die Pixeldichte durch die Anzahl von Pixeln pro Zoll (engl.: *dots per inch*, dpi) angegeben. In der Abbildung 1 ist ein Bild mit der Auflösung 512×512 Pixel dargestellt und immer weiter vergrößert, bis man die einzelnen Pixel erkennen kann.

In Abhängigkeit davon, wie viele Bits man für die Kodierung eines Pixels vergibt, ändert sich die Dateigröße und ggf. die Bildart. Bespielsweise kann man mit einem Bit pro Pixel nur Schwarz-Weiß-Bilder speichern. Die Farbbilder brauchen dagegen wesentlich mehr Informationen und werden meistens mit 24 Bits kodiert. Dabei werden jeweils 8 Bits für die Kodierung der Farben Rot, Grün und Blau gebraucht. Diese drei Farben bilden einen

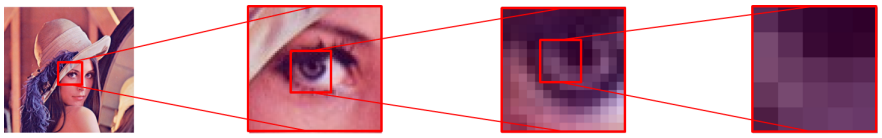


Abbildung 1: Ein digitales Bild und seine Pixel

additiven Farbraum und können durch das Zusammenmischen in verschieden Proportionen unterschiedliche Farbtöne darstellen.

Die verlustfreie Bildkompression wäre dann nichts anderes als die Redundanzen in den Pixelkodierungen von einzelnen Farbkomponenten zu suchen und diese zusammenzufassen. Bei der verlustbehafteten Bildkompression muss zuerst festgestellet werden, welche Informationen für ein digitales Bild weniger wichtig sind, um diese weglassen zu können. Dafür werden die Eigenschaften des menschlichen Auges betrachtet. Eine wichtige Eigenschaft, die für die verlustbehaftete Komprimierung benutzt werden kann, ist die Tatsache, dass das Auge mehr gegenüber Helligkeit als gegenüber Chrominanz empfindlich ist.

Bei den Bildern, die in dem Rot-Grün-Blau-Farbraum (kurz: RGB-Farbraum) gespeichert sind, ist es umständlich die Farbinformationen getrennt von Helligkeitsinformationen zu bearbeiten, denn die Helligkeitsinformation ist in jeder Farbkomponente gespeichert. Um dieses Problem zu beheben, wird anstatt des RGB-Farbraumes der YUV-Farbraum für die Bilddarstellung eingesetzt.

1.2 YUV-Farbraum

Die Bilder, die in dem YUV-Farbraum gespeichert werden, unterscheiden sich nicht von den Bildern, die in dem RGB-Farbraum gespeichert sind. Sie können von einem in einen anderen Farbraum mit Hilfe der folgenden Formeln umgewandelt werden:

$$
\begin{aligned}
Y &= 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B & \quad B &= Y + U/0.493 \\
U &= 0.493 \cdot (B - Y) & \quad R &= Y + V/0.877 \\
V &= 0.877 \cdot (R - Y) & \quad G &= 1.7 \cdot Y - 0.509 \cdot R - 0.194 \cdot B
\end{aligned}
$$

Dabei beinhaltet die Y-Komponente nur die Helligkeitsinformation und die U- und V-Komponenten speichern die Rot-Grün- bzw. Gelb-Blau-Balance. Somit kann die Chrominanz beeinflusst werden ohne den Einfluss auf die Helligkeit auszuüben. Beispielsweise kann ein Bild mit 24 Bits pro Pixel ohne große Unterschiede für das menschliche Auge auf 16 Bits pro Pixel reduziert werden, indem 8 Bits für die Y-Komponente und jeweils 4 Bits für die U- und V-Komponente gelassen werden. Diese Eigenschaft wird auch

in der JPEG-Kodierung verwendet, um die hohen Kompressionsraten ohne große Qualitätsverlust zu erreichen. Die genauere Vorgehensweise wird in dem nächsten Kapitel betrachtet.

2 JPEG

Der Name JPEG steht als Abkürzung für *Joint Photographic Experts Group*. Diese Gruppe wurde im Juni 1987 von CCITT (fr.: *Comité Consultatif International Téléphonique et Télégraphique*) und ISO (engl.: *International Standards Organisation*) gegründet und hat den ersten Entwurf zur Bildkomprimierung bereits im Jahre 1991 erstellt. Das entwickelte Verfahren wird auch JPEG genannt und eignet sich am besten für die Komprimierung von durchgängig getönten Bildern.

In diesem Kapitel werden die vier JPEG-Komprimierungsstrategien und deren Stufen vorgestellt, und ferner einige von den Stufen genauer betrachtet.

2.1 Überblick über die Komprimierungsstufen

In JPEG wird ein Bild über mehrere Bearbeitungsstufen komprimiert. In Abhängigkeit von der ausgewählten Strategie werden einige Stufen ausgelassen. Die Dekomprimierung eines Bildes verläuft über die gleichen Stufen, die für die Komprimierung des Bildes eingesetzt wurden, nur in die entgegengesetzte Richtung. Somit ist JPEG ein symmetrisches Verfahren.

Grundsätzlich werden in JPEG vier verschiedene Komprimierungsstrategien unterschieden:

1. *Sequentielle Strategie.* In dieser Strategie werden die Bildkomponenten von links nach rechts und von oben nach unten abgearbeitet.

2. *Progressive Strategie.* Diese Strategie ist eine zusätzliche Option zur sequentiellen Strategie, in der die komprimierten Daten in bestimmte Blöcke (sogenannte "scans") angeordnet werden, sodass der Dekomprimierer Schritt für Schritt die Daten von jedem Block einliest und dekodiert. Als Folgeeffekt wird bei der Dekodierung das Bild direkt grob dargestellt und bekommt die detaillierte Darstellung mit weiteren dekodierten Daten. Allerdings werden Bilder in progressiver Strategie langsamer als in sequentieller Strategie kodiert, denn der Kodierer muss beim Kodieren alle Koeffizienten im Puffer halten, bis diese am Ende in den verschiedenen Blöcken gespeichert werden.

3. *Hierarchische Strategie.* Bei diesem Verfahren wird das zu komprimierende Bild zuerst in verschiedene Auflösungen geteilt und danach jeweils jede Auflösung komprimiert. Dabei wird die Kodierung von

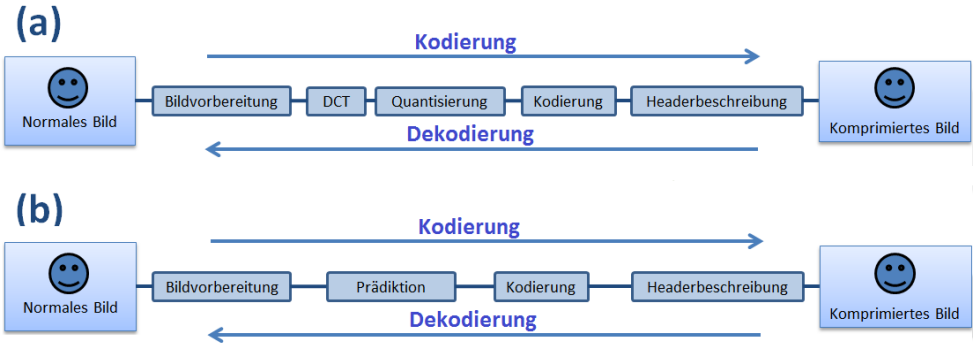


Abbildung 2: **(a)** Komprimierungsstufen bei grundlegenden Strategien. **(b)** Komprimierungsstufen bei der verlustfreien Strategie.

niedriger Auflösung für die Kodierung von höherer Auflösung wiederverwendet, um das komprimierte Bild möglichst klein zu halten. Für die Kodierung verschiedener Auflösungen kann wiederum progressive Strategie verwendet werden. Somit wird ähnlich zur progressiven Strategie ein dekodiertes Bild zuerst grob und zum Schluss detailliert dargestellt. Die Dekodierung kann jedoch bei bestimmter Auflösung gestoppt werden.

4. *Verlustfreie Strategie.* Das mit der verlustfreien Strategie komprimierte Bild kann vollständig zum Originalbild dekomprimiert werden.

Die ersten drei Strategien unterscheiden sich von der verlustfreien Strategie in manchen Komprimierungsstufen und werden als *grundlegend* bezeichnet. Die Komprimierungsstufen der grundlegenden und der verlustfreien Strategien sind auf der Abbildung 2 dargestellt und werden im Weiteren kurz erläutert:

- *Bildvorbereitung.* In dieser Stufe wird das Bild für die weitere Bearbeitung vorbereitet. Zuerst wird das Bild in YUV-Farbraum umgewandelt. Danach, im Falle einer hierarchischen Strategie, wird die Auflösung von Farbkomponenten reduziert. Zum Schluss werden die drei Komponente in die Blöcke von 8×8 Pixel geteilt. Falls ein Block nicht vollständig aus 8×8 Pixeln bestehen kann (z.B. am Rand des Bildes) werden die letzten Pixel nach rechts und nach unten kopiert.

- *DCT.* Die diskrete Kosinustransformation (engl.: *Discrete Cosine Transformation*) ist die Stufe, in welcher die Kodierungswerte der Pixeln eines 8×8-Blocks von dem Ortsbereich in den Frequenzbereich, welcher besser für die Komprimierung geeignet ist, transformiert werden. Die entstandenen Werte können mit Ausnahme von der Endlichkeit der

Rechengenauigkeit ohne Verluste zurück transformiert werden. Diese Stufe wird in dem Kapitel 2.2 genau betrachtet.

- *Quantisierung.* Vor der Kodierung der transformierten Werte müssen diese als Nächstes quantisiert werden. Hierbei werden die Werte des 8×8-Blocks durch entsprechende Werte der Quantisierungstabelle geteilt und zu einer geraden Zahl gerundet. Durch die Quantisierung bekommt man einen Datenblock, der nur aus ganzen Zahlen und mehreren Nullen besteht. Erst in dieser Stufe gehen bei der Kodierung eines Bildes einige Daten verloren. Die detaillierte Beschreibung dieser Stufe ist der Inhalt des Kapitels 2.3.

- *Prädiktion.* Diese Stufe wird im Falle einer verlustfreien Komprimierungsstrategie anstatt der Stufen DCT und Quantisierung angewendet. Dabei werden nur die Nummer des Prädiktors und die Differenz des zu kodierenden Wertes zum Wert des Prädiktors in einem Tupel gespeichert. In dem Kapitel 2.5 sind die Liste aller Prädiktoren sowie ein entsprechendes Beispiel vorgestellt.

- *Kodierung.* Nachdem die Datenblöcke transformiert und quantisiert bzw. in die Tupel (Prädiktor, Differenz) umgewandelt wurden, müssen sie kodiert werden. In JPEG wird dafür RLE (engl.: *Run-Length Encoding*) mit entweder arithmetischer Kodierung oder Huffman-Kodierung kombiniert. In dem Kapitel 2.4 wird die zweite Variante detailliert betrachtet.

- *Headerbeschreibung.* Damit der Dekomprimierer das komprimierte Bild ordnungsgemäß dekomprimieren kann, müssen die verwendete Strategie und die dazugehörigen Parameter in dem komprimierten Bild gespeichert werden. Alle diese Informationen werden in dieser Stufe in Header des komprimierten Bildes hinzugefügt.

2.2 DCT

Mit Hilfe von diskreter Kosinustransformation können die Werte von dem Ortsbereich in den Frequenzbereich und zurück transformiert werden. Für die Transformation von zweidimensionalen Datenblöcken der Größe $m \times n$ in den Frequenzbereich wird FDCT (engl.: *Forward DCT*) angewendet und für die Rücktransformation IDCT (engl.: *Inverse DCT*): [2]

$$FDCT : G_{ij} = \sqrt{\frac{2}{m}}\sqrt{\frac{2}{n}} C_i C_j \sum_{x=0}^{n-1} \sum_{y=0}^{m-1} p_{xy} cos\left[\frac{(2y+1)j\pi}{2m}\right] cos\left[\frac{(2x+1)i\pi}{2n}\right]$$

$$IDCT : p_{xy} = \sqrt{\frac{2}{m}}\sqrt{\frac{2}{n}} C_i C_j \sum_{i=0}^{n-1} \sum_{j=0}^{m-1} C_i C_j G_{ij} cos\left[\frac{(2x+1)i\pi}{2m}\right] cos\left[\frac{(2y+1)j\pi}{2n}\right]$$

$$\text{mit } 0 \leq i \leq n-1, 0 \leq j \leq m-1 \text{ und } C_f := \begin{cases} \frac{1}{\sqrt{2}} & f = 0 \\ 1 & f > 0 \end{cases}$$

Das Anwenden von FDCT auf einen Datenblock p der Größe $n \times m$ liefert einen Datenblock der gleichen Größe von transformierten Daten G.

Die Bedeutung der Datentransformation wird anhand eines einfachen Beispiels dargelegt. Dafür werden die Daten eines eindimensionalen Arrays mit Werten $[16, 13, 9, 8, 9, 10, 9, 8]$ in den Frequenzbereich mit FDCT für die eindimensionale Daten transformiert: [1]

$$G_i = \sqrt{\frac{2}{n}} C_i \sum_{x=0}^{n-1} p_x cos\left[\frac{(2x+1)i\pi}{2n}\right] \text{ mit } C_i = \begin{cases} \frac{1}{\sqrt{2}} & i = 0 \\ 1 & i > 0 \end{cases}$$

Auf der Abbildung 3 sind die Originalwerte und die transformierten Werte auf einer Zeitachse zu sehen. Die transformierten Daten werden als Frequenzkoeffizienten bezeichnet, deren Eigenschaften in folgenden Punkten aufgelistet werden können:

- Der Wert G_0 speichert die wichtigsten Informationen von Originaldaten und wird als DC (engl.: *Direct Current*) bezeichnet. Wenn nur dieser Wert bei IDCT genommen wird und die anderen Werte gleich Null gesetzt werden, ergibt sich der Mittelwert aller Originaldaten für jeden Wert der zurücktransformierten Daten.
- Die weiteren Werte werden als AC (engl.: *Alternating Current*) bezeichnet. Diese Werte tragen die Differenzinformationen zwischen den Werten in dem Originaldatenblock in sich.
- Je größer die Stelle x eines AC-Wertes G_x, desto detailliertere Differenzen von den Originaldaten können auf diese Werte wiederhergestellt werden.
- Wenn die Originaldaten in einer Korrelation zueinander stehen, dann nähern sich die transformierten AC-Werte der 0.

Darauffolgend können die AC-Werte, die detaillierte Informationen speichern, gleich 0 gesetzt werden, sodass die zurücktransformierten Daten keinen großen Unterschied zu den Originaldaten aufweisen werden. Beispielsweise wird das Array mit gerundeten Werten $G' : [29, 5, 4, 4, 0, 0, 0, 0]$ aus dem Beispiel von Abbildung 3 nach dem Anwenden von IDCT in das Array $p' : [16.2, 12.7, 8.9, 7.8, 9.0, 10.1, 9.3, 8.0]$ transformiert.

Im Falle eines 8×8-Datenblocks p einer Bildkomponente wird nach dem Anwenden von FDCT ein 8×8-Datenblock G von Frequenzkoeffizienten erzeugt. Die erzeugten Koeffizienten beinhalten die oben aufgelisteten Eigenschaften. Der DC-Wert von G_{00} speichert beispielsweise den Mittelfarbton

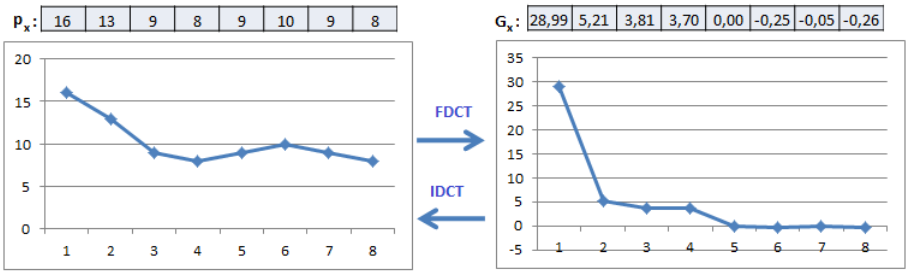


Abbildung 3: (Rück-)Transformation der Daten eines eindimensionalen Arrays mit Hilfe von DCT.

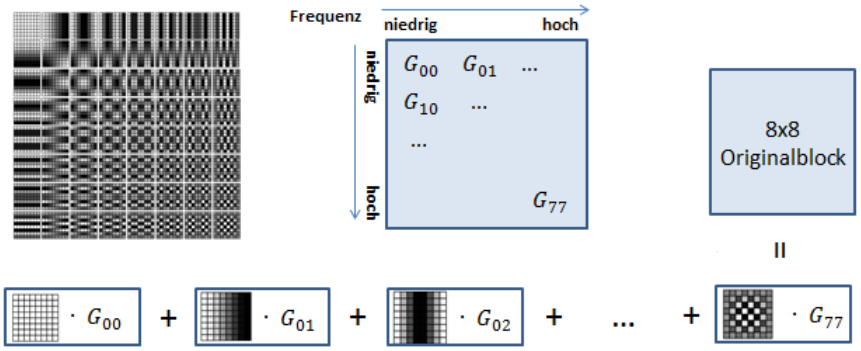


Abbildung 4: 64 Basisbilder von DCT.

und die AC-Werte die Farbübergänge von dem Originalblock in verschiedenen Frequenzen. Abbildung 4 liefert eine graphische Darstellung der 64 Basisbilder von DCT für einen 8×8-Datenblock. Die Basisbilder können mit entsprechenden Koeffizienten multipliziert und aufeinander addiert werden, sodass das Resultat dem originalen 8×8-Block entspricht.

Ein 8×8-Block kann auch hohe Werte bei hohen Frequenzkoeffizienten besitzen. In diesem Fall stehen die Werte des Blocks in keiner Korrelation zueinander. Das kann beispielsweise in einem 8×8-Block, der einen harten Übergang hat, passieren. Das Entfernen von hohen Frequenzen kann in diesem Fall die Originaldaten stark beschädigen. Als Beispiel ist auf der Abbildung 5 ein 8×8-Block vor und nach dem Entfernen von hohen Frequenzkoeffizienten dargestellt, der einen harten Übergang hat.

Nachdem alle Daten transformiert wurden, müssen diese für die Kodierung entsprechend vorbereitet werden. Das passiert in der nächsten Stufe.

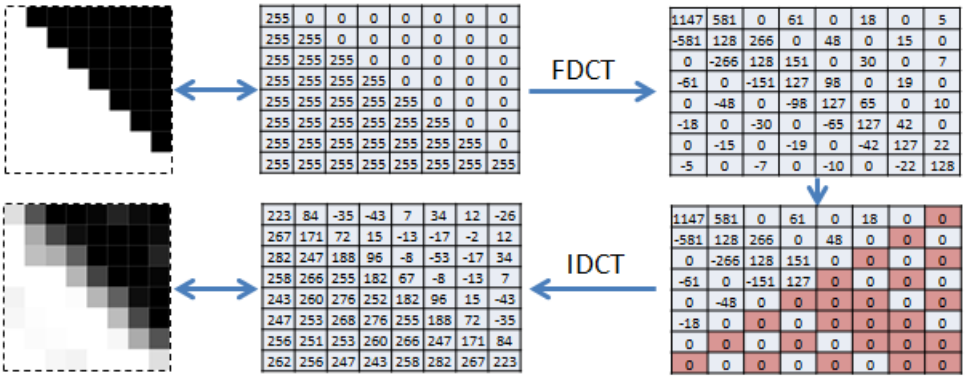


Abbildung 5: Der Effekt, welcher beim Entfernen von hohen Frequenzkoeffizienten eines 8×8-Blocks mit hartem Übergang entsteht.

2.3 Quantisierung

Die Werte, die man nach dem Anwenden von FDCT bekommt, sind meistens keine ganzen Zahlen und eignen sich daher schlecht für die Komprimierung. In dieser Stufe werden die Frequenzkoeffizienten eines 8×8-Blocks durch die entsprechenden Werte von Quantisierungstabellen geteilt und zu einer geraden Zahl gerundet.

In der JPEG-Software werden allgemein zwei Arten von Quantisierungstabellen unterschieden: [3]

1. *Standard-Tabellen.* In diesem Fall sind eine Quantisierungstabelle für die Frequenzkoeffizienten der Y-Komponente und eine Quantisierungstabelle für die Frequenzkoeffizienten der U- und V-Komponente vorgegeben. Diese zwei Tabellen sind das Resultat, welches nach vielen von JPEG durchgeführten Experimenten entstanden ist und zum JPEG-Standard erklärt wurde. Die Werte dieser Tabellen sind in der Tabelle 1 dargestellt.

2. *Benutzerdefinierte Tabelle.* Dabei werden die Frequenzkoeffizienten aller drei Komponente durch die Werte einer einfachen Quantisierungstabelle, die durch die Eingabe des Parameters R mit der Formel: $Q_{xy} = 1 + (x + y) \cdot R$ berechnet wird, geteilt. Als Beispiel sind die Werte zweier solchen Tabellen für $R = 2$ und für $R = 8$ in der Tabelle 2 dargestellt.

In beiden Fällen haben die Quantisierungstabellen einen kleinen Wert Q_{xy} an der Stelle $(0, 0)$, welcher mit zunehmender x- und y-Stelle wächst. Nach der Quantisierung werden dadurch hohe Frequenzen gleich 0 gesetzt. Der DC-Wert und die AC-Werte dagegen, die eine niedrige Frequenz haben,

Q_{xy} für Y-Komponente:

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

Q_{xy} für U-, V-Komponente:

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 1: Standard-Quantisierungstabellen in JPEG Software.

Q_{xy} für Y-, U- ,V-Komponente mir $R = 2$:

1	3	5	7	9	11	13	15
3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29

Q_{xy} für Y-, U-, V-Komponente mir $R = 8$:

1	9	17	25	33	41	49	57
9	17	25	33	41	49	57	65
17	25	33	41	49	57	65	73
25	33	41	49	57	65	73	81
33	41	49	57	65	73	81	89
41	49	57	65	73	81	89	97
49	57	65	73	81	89	97	105
57	65	73	81	89	97	105	113

Tabelle 2: Quantisierungstabellen, die durch die Eingabe des Parameters R mit der Formel $Q_{xy} = 1 + (x + y) \cdot R$ erzeugt werden.

bleiben mit hoher Wahrscheinlichkeit ungleich 0. Dadurch gehen die hohen Frequenzen eines 8×8-Blocks bei der Dequantisierung verloren. Hier ist zu beachten, dass die Werte der Quantisierungstabelle für die Farbkomponenten schneller als die Werte der Quantisierungstabelle für die Helligkeitskomponente wachsen. Infolgedessen wird die Information bei den Farbkomponenten im Vergleich zur Helligkeitskomponente wesentlich mehr reduziert. Das ist der Punkt, in dem JPEG die beschriebene Eigenschaft des menschlichen Auges ausnutzt.

In der Abbildung 6 ist ein möglicher Verlust von hohen Frequenzen eines 8×8-Blocks der Y-Komponente, der nach der Dequantisierung entsteht, dargestellt.

2.4 Kodierung

Im Normalfall bestehen die quantisierten AC-Werte aus mehreren Nullen, zwischen denen andere Zahlen vorkommen können. Außerdem liegen die DC-Werte eines Fotobildes im Allgemeinen sehr nah beieinander, denn der mittlere Farbton zwischen zwei benachbarten 8×8-Blöcken unterscheidet sich selten stark voneinander. Aus diesen Gründen werden DC- und AC-Werte in JPEG getrennt voneinander kodiert. Dabei werden aus einer Folge von DC-Werten der erste Wert und die Differenzen zwischen zwei benachbarten DC-Werten kodiert. Nach jeder solchen Kodierung folgt die dazugehörige Kodierung von den restlichen 63 AC-Werten des entsprechenden

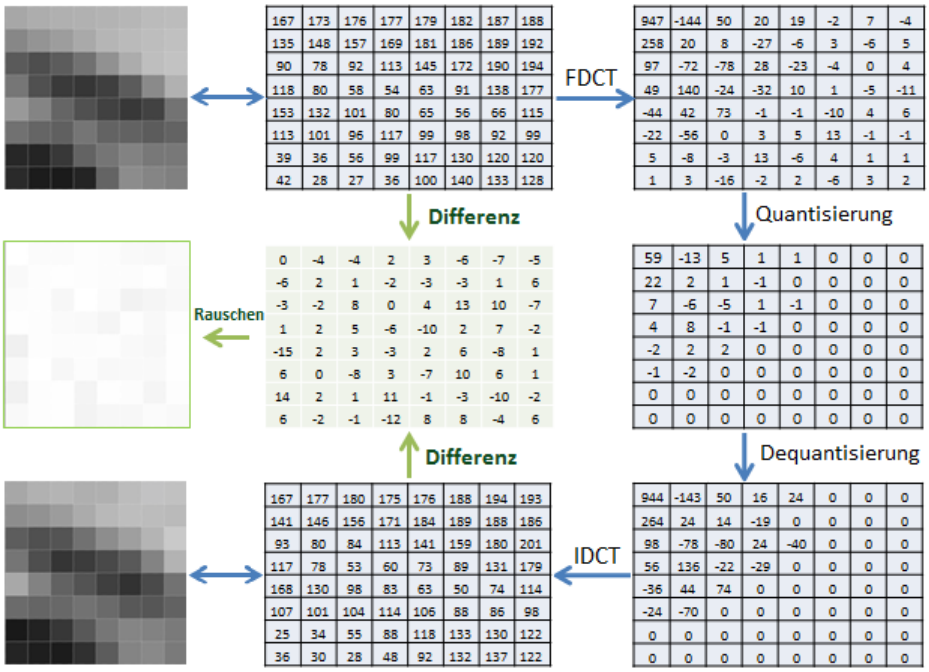


Abbildung 6: Differenz zweier 8×8-Blöcke nach der DCT und (De-) Quantisierung.

8×8-Blocks.

Die Kodierung von DC-Werten wird mit Hilfe von Kodierungswerten aus der Tabelle 3 durchgeführt. Diese Tabelle beinhaltet Huffman-Kodes, die von JPEG empfohlen werden, und besteht aus insgesamt 17 Zeilen. Jede Zeile i ist durchnummeriert von 0 bis 16 und speichert eine unäre Kodierung ihrer Zeilennummer sowie eine Menge von Werten im Bereich $[-(2^i-1), +(2^i-1)]$ ohne die Werte im Zwischenbereich $[-(2^{i-1}-1), +(2^{i-1}-1)]$. Um eine Zahl x mit Hilfe dieser Tabelle zu kodieren, wird zuerst die Zeile i, in welcher sich x befindet, gesucht und der Index j von x in der entsprechenden Zahlenfolge gemerkt. Dabei beginnt die Indexierung in jeder Zeile mit 0. Die Zahl x wird dann als unäre Kodierung von i gefolgt von binärer Kodierung von j mit i Bits gespeichert.

Als Beispiel wird weiter die Folge $[59, 48, 52, 59]$ von quantisierten DC-Werten betrachtet. Die Zahl 59 befindet sich in der sechsten Zeile und hat den Index 59. Als Kodierung für die Zahl 59 wird also 1111110 111011 gespeichert. Weiterhin werden nur die Differenzen betrachtet. Die Differenz zu der nächsten Zahl ist -11. Diese befindet sich in der vierten Zeile und hat den Index 4. Die Kodierung der Zahl 48 entspricht somit der 11110 0100. Nachdem die Differenzen zu den zwei nächsten Werten kodiert werden,

0:	0	0
1:	10	-1,1
2:	110	-3,-2,2,3
3:	1110	-7,-6,-5,-4,4,5,6,7
4:	11110	-15,-14,-13,-12,-11,-10,-9,-8,8,9,10,11,12,13,14,15
5:	111110	-31,-30,-29,-28,-27,. . .,-17,-16,16,17,. . .,27,28,29,30,31
6:	1111110	-63,-62,-61,-60,-59,. . .,-33,-32,32,33,. . .,59,60,61,62,63
7:	11111110	-127,-126,-125,-124,. . .,-65,-64,64,65,. . .,124,125,126,127
⋮	⋮	⋮
14:	111111111111110	-16383,-16382,. . .,-8193,-8192,8192,8193,. . .,16382,16383
15:	1111111111111110	-32767,-32766,. . .,-16385,-16384,16384,16385,. . .,32766,32767
16:	11111111111111110	32768

Tabelle 3: Kodierungstabelle für die DC-Werte.

wird das Resultat wie folgt aussehen: [1111110111011, 111100100, 1110100, 1110111]. Nach jedem von diesen Werten folgt eine zusätzliche Kodierung von den entsprechenden 63 AC-Werten.

Als Nächstes wird die Kodierung von AC-Werten eines Blocks mit Kombination von Huffman und RLE betrachtet. Dabei geht die Kodierung folgende Schritte durch:

1. Die 63 AC-Werte des 8×8-Blocks werden im Zick-Zack angeordnet. Dadurch werden die quantisierten Koeffizienten von hohen Frequenzen an das Ende der Folge geschoben, sodass ab einer bestimmten Stelle nur die Nullen folgen. Eine solche Restfolge von Nullen wird somit als Blockende mit 1010 kodiert. Für die weiteren Werte, die sich vor dem Blockende befinden, wird von links nach rechts mit dem nächsten Schritt weitergemacht.

2. Solange das Blockende nicht erreicht wurde, wird die nicht kodierte Zahl $x \neq 0$ gesucht.

3. In Z wird die Anzahl von aufeinanderfolgenden Nullen, die sich unmittelbar vor x befinden, gespeichert.

4. In der Tabelle 3 wird nach x gesucht und die entsprechende Zeilennummer R und der Index I gemerkt.

5. Für diesen Schritt wird die Tabelle 4, die die von JPEG empfohlenen Kodierungen für die AC-Werte beinhaltet, benötigt. In K wird der Wert, der sich in dieser Tabelle in der Zeile R und Spalte Z befindet, gespeichert.

6. Die Zahl x und die unmittelbar davor aufeinanderfolgenden Nullen werden als der Kode K kodiert, gefolgt von dem mit R Bits kodiertem Index I. Es wird mit Schritt 2 weitergemacht.

↓R, Z →	0	1	···	15
0:	1010(EOB)			11111111001(ZRL)
1:	00	1100	···	1111111111110101
2:	01	11011	···	1111111111110110
3:	100	1111001	···	1111111111110111
4:	1011	111110110	···	1111111111111000
5:	11010	11111110110	···	1111111111111001
⋮	⋮	⋮	...	⋮

Tabelle 4: Kodierungstabelle für die AC-Werte.

Die Huffman-Kodes in der Tabelle 4 sind nicht die einzigen Kodierungen, die von JPEG-Standard empfohlen werden. Ein JPEG-Codec kann bis zu vier verschiedenen Tabellen mit Huffman-Kodes benutzen, wobei in den grundlegenden Strategien nur zwei solche Tabellen benutzt werden können.

Zwei wichtige Werte aus Tabelle 4 befinden sich in der Zeile $R = 0$: Diese Zeile kann mit dem oben beschriebenen Algorithmus nie erreicht werden, denn es wird immer nach einem $x \neq 0$ gesucht. Mit dem Kode an der Stelle $(R, Z) = (0, 0)$ wird das Blockende bezeichnet. Für die 15 aufeinanderfolgenden Nullen wird der Kode an der Stelle $(R, Z) = (0, 15)$ benutzt.

Die Kodierung der AC-Werte wird an dem Beispiel dieser Folge gezeigt:

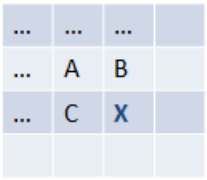
$$59, 2, 0, -2, 0, -2, \underbrace{0, \ldots, 0}_{13}, -1, \underbrace{0, \ldots, 0}_{44}$$

Der erste AC-Koeffizient $x = 2$ hat keine davorstehenden Nullen, sodass $Z = 0$ ist. In der Tabelle 3 befindet sich die Zahl x in der Zeile $R = 2$ und hat den Index $I = 2$. Der Kode K an der Stelle $(R, Z) = (2, 0)$ aus der Tabelle 4 ist 01. Somit wird x als 01 10 kodiert. Der nächste AC-Koeffizient, der ungleich Null ist, hat den Wert $x = -2$. Vor diesem Wert steht eine Null, deswegen ist $Z = 1$. Die entsprechende Zeile R und Index I aus der Tabelle 3 sind 2 bzw. 1. In der Tabelle 4 steht an der Stelle $(R, Z) = (2, 1)$ der Kode 11011. Dadurch wird der Koeffizient x mit der davorstehenden Null als 11011 01 kodiert. So wird weiterverfahren bis die letzten 44 Nullen, die mit 1010 kodiert werden, erreicht sind. Die vollständige Kodierung dieser Folge (inklusiv DC-Wert) ist: 1111110111011 0110 11011 1011101010 1010.

2.5 Prädiktion

In der verlustfreien JPEG-Komprimierungsstrategie wird kein DCT und keine Quantisierung angewendet, denn durch die Quantisierung gehen die hohen Frequenzen endgültig verloren. Stattdessen wird die Prädiktion durchgeführt.

In dieser Stufe wird die Differenz zwischen den Werten von benachbarten Pixeln betrachtet. Mit einem Prädiktor aus der Abbildung 7 wird ein Wert,



Prädiktor	Wert
0	keine Vorhersage
1	A
2	B
3	C
4	A+B-C
5	A+(B-C)/2
6	B+(A-C)/2
7	(A+B)/2

Abbildung 7: Liste mit Prädiktoren und deren Werte.

der am nächsten zu dem zu kodierenden Wert X liegt, bestimmt. Der Wert dieses Prädiktors wird von dem Wert X subtrahiert. Nachfolgend werden die Nummer des benutzten Prädiktors und der neue X Wert mit Huffman oder arithmetischer Kodierung komprimiert.

Mit diesem Verfahren können die passenden Prädiktoren für die Anfangswerte, die sich an dem linken oberen Rand des Bildes befinden, nicht immer gefunden werden. In diesem Fall wird der Prädiktor 0 mit dem Originalwert X verwendet. Andererseits, wenn der Wert X durch einen Prädiktor verkleinert werden kann, dann wird die Nummer des Prädiktors mit der Differenz zwischen dem Wert X und dem Wert des Prädiktors kodiert. Beispielsweise, wenn die Werte $(A, B, C) = (10, 8, 6)$ bereits kodiert wurden und der neue Wert $X = 12$ an der Reihe ist, dann wird der Prädiktor 4 mit der Differenz 0 kodiert.

3 Vor- und Nachteile von JPEG

Das JPEG-Komprimierungsverfahren ist sehr flexibel, denn beim Bedarf können mehrere Parameter von dem Nutzer eingestellt und optimaler Datenverlust beeinflusst werden. Durch das Nutzen der Eigenschaften des menschlichen Auges in Zusammenspiel mit DCT kann die Kompressionsrate den Faktor 10 bis 20 erreichen, sodass der Qualitätsverlust nicht direkt erkennbar wird. Außerdem ist JPEG sehr weit verbreitet. Dies gilt nicht nur für die Software- sondern auch für die Hardwareprodukte.

Zu den Nachteilen werden die Qualitätsverluste, die auch bei jedem Speichern nach der Bildbearbeitung entstehen, gezählt. Einschließlich eignet sich JPEG schlecht für die Schwarz-Weiß-Bilder, denn diese können harte Übergänge zwischen zwei Pixeln haben und würden somit in einem 8×8-Block in keiner Korrelation zueinander stehen. Ein Beispiel dazu wurde bereits in der Abbildung 5 dargestellt.

4 Fazit

Heutzutage ist JPEG unbestreitbar der Gewinner aller Kompressionsverfahren, die für die Digitalfotos entwickelt wurden. Es ist nicht nur ein einfaches Verfahren, das auf beliebigen Plattformen unterstützt wird, sondern auch ein sehr starkes und flexibles Verfahren, das bereits seit über 10 Jahren zu den meistgenutzten zählt.

Literatur

[1] J. Miano. *Compressed Image File Formats: Jpeg, Png, Gif, Xbm, Bmp.* Siggraph Series. Addison Wesley, 1999.

[2] D. Salomon. *Data Compression: The Complete Reference.* Number Bd. 10 in Data compression: the complete reference. Springer, 2007.

[3] D. Salomon, G. Motta, and D. Bryant. *Handbook of Data Compression.* Springer, 2009.