Datenkompression:
Mehr Transformationen und JPEG-Standard

H. Fernau

email: fernau@uni-trier.de

WiSe 2008/09
Universität Trier

Mehr Transformationen und Filter

- Wavelets als moderner Filtererzeuger
- Matrixbasierte Bildtransformationen—Wiederholung
- Diskrete Kosinustransformation
- Anwendung dieser Verfahren bei JPEG

Wavelets

Ein flexibleres und modernes Instrument zur Signalzerlegung liefern *Wavelets*. Bei ihnen steht sowohl der Zeit- als auch der Frequenzbereich parametrisiert zur Verfügung.

Sehr beliebt sind die sog. *Haar-Wavelets*. Aus der einfachen *Urfunktion* (engl.: mother wavelet)

$$\psi_{0,0}(x) = \begin{cases} 1 & 0 \leq x < \frac{1}{2} \\ -1 & \frac{1}{2} \leq x < 1 \\ 0 & \text{sonst} \end{cases} \quad \text{wird durch}$$

$$\begin{aligned} \psi_{j,k}(x) &= \psi_{0,0}(2^j x - k) \\ &= \begin{cases} 1 & k2^{-j} \leq x < (k+\frac{1}{2})2^{-j} \\ -1 & (k+\frac{1}{2})2^{-j} \leq x < (k+1)2^{-j} \\ 0 & \text{sonst} \end{cases} \end{aligned}$$

eine Schar von Funktionen. j kann als *Streckparameter* (*Frequenzparameter*) und k als *Verschiebeparameter* (*Zeitparameter*) gedeutet werden.

Wavelets — ein einfaches Beispiel

Um eine Idee von der Arbeitsweise von Wavelets zu erhalten, betrachten wir exemplarisch nun als Urfunktion

$$\phi_{0,0}(x) \;=\; \begin{cases} 1 & 0 \le x < 1 \\ 0 & \text{sonst} \end{cases}$$

sowie die erzeugte Schar

$$\phi_{j,k}(x) \;=\; \phi_{0,0}(2^j x - k)\,.$$

Angenommen, wir wollten eine Funktion $f : [0, N] \to \mathbb{R}_+$ approximieren. In erster Näherung setzen wir

$$\begin{aligned} \phi_f^0(t) &= \sum_{k=0}^{N-1} c_{0,k}\phi_{0,k} \quad \text{mit} \\ c_{0,k} &= \int_k^{k+1} f(t)[\phi_{0,k}(t)]dt \end{aligned}$$

Durch Wahl eines höheren Frequenzparameters können wir die Genauigkeit der Approximation steigern:

$$\begin{aligned} \phi_f^j(t) &= \sum_{k=0}^{2^j N-1} c_{j,k}\phi_{j,k} \quad \text{mit} \\ c_{j,k} &= 2^j \int_{k/2^j}^{(k+1)/2^j} f(t)dt\,. \end{aligned}$$

Offensichtlich gilt

$$c_{j-1,k} = 1/2(c_{j,2k} + c_{j,2k+1})\,. \qquad (1)$$

Unser Ziel ist es, Funktionen in Bestandteile (mit evtl. unterschiedlichen Charakteristika) zu zerlegen. Wenn nun ϕ_f^1 die Funktion f genügend approximiert, kann man ϕ_f^0 als niederfrequenten Anteil betrachten und muss dann die Differenz $\phi_f^1 - \phi_f^0$ diskutieren. Es gilt wegen Gleichung (1):

$$\begin{aligned}&\phi_f^1(t) - \phi_f^0(t)\\ &= \begin{cases} c_{0,k} - c_{1,2k} = -1/2c_{1,2k} + 1/2c_{1,2k+1} & k \le t < k + 1/2 \\ c_{0,k} - c_{1,2k+1} = 1/2c_{1,2k} - 1/2c_{1,2k+1} & k + 1/2 \le t < k + 1 \end{cases}\end{aligned}$$

M. a. W., die Differenz lässt sich leicht mit Haar-Wavelets ausdrücken, nämlich:

$$\phi_f^1(t) - \phi_f^0(t) = \underbrace{(-c_{1,2k} + c_{1,2k+1})}_{b_{0,k}} \psi_{0,k}(t)\,.$$

Die $2N$-Punkte Folge $c_{1,k}$ kann so in zwei N-Punkt-Folgen $c_{0,k}$ und $b_{0,k}$ zerlegt werden;
die zweite Folge kann als Faktoren von Wavelets interpretiert werden.

Allgemein lässt sich jede Funktionenschar $\phi_{j,k}$, die gewissen Skalier-, Darstellungs- und Integrabilitätsbedingungen genügt, dafür benutzen, eine zugehörige Wavelet-Familie zu definieren. Aus der Darstellungsbeziehung

$$\phi_{0,0}(t) = \sum h_n \phi_{1,n}(t)$$

gewinnt man die Impulsantwort h_n des Glättungsfilters, denn $\phi_{0,0}$ kann als stetige Form des Impulses gesehen werden, und die Darstellung

$$\psi_{0,0}(t) = \sum (-1)^{N-n-1} h_n \phi_{1,n}(t)$$

des zugehörigen Ur-Wavelets liefert die Impulsantwort des Differenzierfilters. Beliebt sind insbesondere die so zu erhaltenen *Daubechies- und Coiflet-Filter*. Wavelets liefern Spiegelfilter.

Wichtig: die Aufspaltung des Ursignals in Glätte- und Differenzanteil kann man weitertreiben, indem der Glätteanteil rekursiv weiter aufgespalten wird.
Die Differenzanteile m-ter Stufe lassen sich dann durch die Wavelets $\psi_{m,k}$ darstellen.
Die so mögliche rekursive *Multiresolutionsanalyse* ist einer der wesentlichen Vorteile Wavelet-basierter Zeit/Frequenz-Analyse gegenüber dem (älteren) Fourier-Ansatz.

Matrixbasierte Bildtransformation — Übersicht

Wir definieren solche Transformationen für einen gegebenen $N \times N$ Block

$$X = \begin{bmatrix} x_{0,0} & x_{0,1} & \cdots & x_{0,N-1} \\ x_{1,0} & x_{1,1} & \cdots & x_{1,N-1} \\ \vdots & & & \vdots \\ x_{N-1,0} & x_{N-1,1} & \cdots & x_{N-1,N-1} \end{bmatrix}$$

folgendermaßen:

$$\Theta = AXA^T$$

und die inverse Transformation: $X = B\Theta B^T$. Alle Transformationen, die wir hier betrachten werden, sind orthonormal. Daher ist es einfach, die inverse Transformation zu bekommen:

$$X = A^T\Theta A$$

Diskrete Fouriertransformation mit (komplexer) Transformationsmatrix:

$$a_{\ell,k} = \sqrt{1/N}(\cos\frac{2\pi\ell k}{N} - i\sin\frac{2\pi\ell k}{N}).$$

Die Basisvektoren für $N = 8$ sehen folgendermaßen aus: Die reelle Basis:

$$\begin{bmatrix}
0,35 & 0,35 & 0,35 & 0,35 & 0,35 & 0,35 & 0,35 & 0,35 \\
0,35 & 0,25 & 0,00 & -0,25 & -0,35 & -0,25 & 0,00 & 0,25 \\
0,35 & 0,00 & -0,35 & 0,00 & 0,35 & 0,00 & -0,35 & 0,00 \\
0,35 & -0,25 & 0,00 & 0,25 & -0,35 & 0,25 & 0,00 & -0,25 \\
0,35 & -0,35 & 0,35 & -0,35 & 0,35 & -0,35 & 0,35 & -0,35 \\
0,35 & -0,25 & 0,00 & 0,25 & -0,35 & 0,25 & 0,00 & -0,25 \\
0,35 & 0,00 & -0,35 & 0,00 & 0,35 & 0,00 & -0,35 & 0,00 \\
0,35 & 0,25 & 0,00 & -0,25 & -0,35 & -0,25 & 0,00 & 0,25
\end{bmatrix}$$

und die imaginäre Basis ($\rightsquigarrow$ „Datenaufblähung")

$$\begin{bmatrix}
0,00 & 0,00 & 0,00 & 0,00 & 0,00 & 0,00 & 0,00 & 0,00 \\
0,00 & 0,25 & 0,35 & 0,25 & 0,00 & -0,25 & -0,35 & -0,25 \\
0,00 & 0,35 & 0,00 & -0,35 & 0,00 & 0,35 & 0,00 & -0,35 \\
0,00 & 0,25 & -0,35 & 0,25 & 0,00 & -0,25 & 0,35 & -0,25 \\
0,00 & 0,00 & 0,00 & 0,00 & 0,00 & 0,00 & 0,00 & 0,00 \\
0,00 & -0,25 & 0,35 & -0,25 & 0,00 & 0,25 & -0,35 & 0,25 \\
0,00 & -0,35 & 0,00 & 0,35 & 0,00 & -0,35 & 0,00 & 0,35 \\
0,00 & -0,25 & -0,35 & -0,25 & 0,00 & 0,25 & 0,35 & 0,25
\end{bmatrix}$$

DCT — Diskrete Cosinus-Transformation

Die (reelle) Transformationsmatrix lautet hier:

$$a_{i,j} = \begin{cases} \sqrt{1/N} & i = 0,\ j = 0, 1, \ldots, N-1 \\ \sqrt{2/N} \cos \frac{(2j+1)i\pi}{2N} & i = 1, \ldots, N-1,\ j = 0, 1, \ldots, N-1. \end{cases}$$

Für den 8×8-Fall ergeben sich als Basisvektoren:

$$A = \begin{bmatrix} 0,35 & 0,35 & 0,35 & 0,35 & 0,35 & 0,35 & 0,35 & 0,35 \\ 0,49 & 0,42 & 0,28 & 0,10 & -0,10 & -0,28 & -0,42 & -0,49 \\ 0,46 & 0,19 & -0,19 & -0,46 & -0,46 & -0,19 & 0,19 & 0,46 \\ 0,42 & -0,10 & -0,49 & -0,28 & 0,28 & 0,49 & 0,10 & -0,42 \\ 0,35 & -0,35 & -0,35 & 0,35 & 0,35 & -0,35 & -0,35 & 0,35 \\ 0,28 & -0,49 & 0,10 & 0,42 & -0,42 & -0,10 & 0,49 & -0,28 \\ 0,19 & -0,46 & 0,46 & -0,19 & -0,19 & 0,46 & -0,46 & 0,19 \\ 0,10 & -0,28 & 0,42 & -0,49 & 0,49 & -0,42 & 0,28 & -0,10 \end{bmatrix}$$

Matrizenschreibweise ausgepackt für DCT

1. Vom "Zeit-" in den "Frequenzbereich":

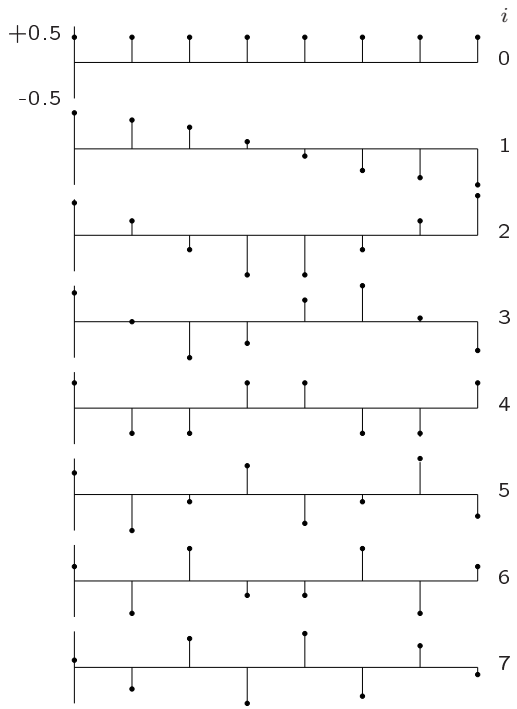
$$F_{x,y} = \frac{2 \cdot C(x) \cdot C(y)}{N} \cdot \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} f_{i,j} \cdot \cos\left(\frac{(2i+1) \cdot x \cdot \pi}{2 \cdot N}\right) \cdot \cos\left(\frac{(2j+1) \cdot y \cdot \pi}{2 \cdot N}\right)$$

mit $C(n) = \begin{cases} \frac{1}{\sqrt{2}}, & n = 0 \\ 1, & n \neq 0 \end{cases}$

2. und zurück:

$$f_{i,j} = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} \frac{2 \cdot C(x) \cdot C(y)}{N} \cdot F_{x,y} \cdot \cos\left(\frac{(2i+1) \cdot x \cdot \pi}{2 \cdot N}\right) \cdot \cos\left(\frac{(2j+1) \cdot y \cdot \pi}{2 \cdot N}\right)$$

Die graphische Darstellung der **Basisvektoren der DCT**



Vergleichen Sie die Werte der Basisvektoren der diskreten Cosinus-Transformation mit den angegebenen Werten für die KLT: Die Werte von DCT und KLT unterscheiden sich kaum, was die gute Qualität der DCT erklärt.

DCT: Beispiel 1: Zwei aneinanderstoßende Flächen

$$X = \begin{bmatrix} 10 & 10 & 10 & 10 & 128 & 128 & 128 & 128 \\ 10 & 10 & 10 & 10 & 128 & 128 & 128 & 128 \\ 10 & 10 & 10 & 10 & 128 & 128 & 128 & 128 \\ 10 & 10 & 10 & 10 & 128 & 128 & 128 & 128 \\ 10 & 10 & 10 & 10 & 128 & 128 & 128 & 128 \\ 10 & 10 & 10 & 10 & 128 & 128 & 128 & 128 \\ 10 & 10 & 10 & 10 & 128 & 128 & 128 & 128 \\ 10 & 10 & 10 & 10 & 128 & 128 & 128 & 128 \end{bmatrix}$$

besitzen folgende Cosinus-Transformierte:

$$\Theta = \begin{bmatrix} 552,0 & -427,7 & 0,0 & 150,2 & 0,0 & -100,4 & 0,0 & 85,1 \\ 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\ 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\ 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\ 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\ 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\ 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \\ 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 & 0,0 \end{bmatrix}$$

DCT: Beispiel 2: Ein Block des Bildes „Brücke"

$$X = \begin{bmatrix} 94 & 96 & 113 & 135 & 196 & 116 & 110 & 106 \\ 91 & 125 & 120 & 153 & 192 & 135 & 108 & 124 \\ 87 & 119 & 144 & 122 & 190 & 131 & 115 & 132 \\ 88 & 93 & 141 & 100 & 168 & 149 & 128 & 122 \\ 95 & 93 & 139 & 169 & 172 & 147 & 154 & 135 \\ 87 & 100 & 124 & 173 & 180 & 132 & 173 & 178 \\ 104 & 85 & 112 & 120 & 167 & 158 & 132 & 166 \\ 112 & 83 & 124 & 123 & 154 & 152 & 148 & 165 \end{bmatrix}$$

wird transformiert in:

$$\Theta = \begin{bmatrix} 1049,9 & -125,7 & -121,2 & 4,7 & 50,1 & -40,0 & 2,4 & 55,2 \\ -30,4 & 59,0 & -48,7 & 6,0 & -0,2 & -35,8 & -22,7 & 13,3 \\ -17,2 & 5,7 & 12,8 & 16,4 & 24,6 & -11,0 & 0,6 & -5,4 \\ 15,7 & -14,8 & -17,5 & -9,4 & 25,7 & 26,1 & -30,5 & -10,9 \\ -19,9 & 5,6 & -0,6 & 14,8 & -13,1 & 22,2 & 14,5 & 12,3 \\ -30,6 & -3,6 & 13,5 & 17,2 & -8,3 & -18,8 & 20,3 & 19,7 \\ 11,1 & -6,8 & 2,1 & -19,6 & 4,3 & 5,6 & -16,0 & 13,4 \\ 0,5 & 9,5 & -7,4 & 2,8 & 4,0 & 1,0 & 5,5 & -0,2 \end{bmatrix}$$

DCT: Beispiel 3: Ein einzelner Punkt

$$X = \begin{bmatrix} 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 128 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \\ 10 & 10 & 10 & 10 & 10 & 10 & 10 & 10 \end{bmatrix}$$

ergibt:

$$\Theta = \begin{bmatrix} 94,8 & 4,1 & -19,3 & -11,6 & 14,8 & 17,3 & -8,0 & -20,5 \\ 4,1 & 1,1 & -5,3 & -3,2 & 4,1 & 4,8 & -2,2 & -5,6 \\ -19,3 & -5,3 & 25,2 & 15,1 & -19,3 & -22,7 & 10,4 & 26,7 \\ -11,6 & -3,2 & 15,1 & 9,1 & -11,6 & -13,6 & 6,3 & 16,1 \\ 14,8 & 4,1 & -19,3 & -11,6 & 14,8 & 17,3 & -8,0 & -20,5 \\ 17,3 & 4,8 & -22,7 & -13,6 & 17,3 & 20,4 & -9,4 & -24,1 \\ -8,0 & -2,2 & 10,4 & 6,3 & -8,0 & -9,4 & 4,3 & 11,1 \\ -20,5 & -5,6 & 26,7 & 16,1 & -20,5 & -24,1 & 11,1 & 28,4 \end{bmatrix}$$

Frage: Warum DCT und nicht DFT?

Diskrete Cosinus-Transformation und diskrete Fourier-Transformation hängen eng zusammen: Spiegelt man die N-Punkt-Folge einer DFT am rechten Rand, so erhält man die „zugehörige" $2N$-Punkt-Folge der DCT.

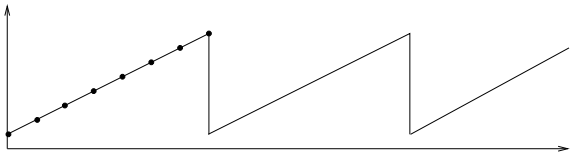
Die wichtigsten Unterschiede zwischen DCT und DFT sind folgende:

1. DFT generiert komplexe Zahlen, während DCT nur reelle Zahlen ausgibt.

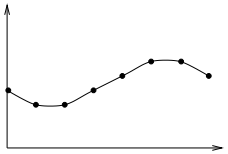
2. DFT nimmt an, dass die Funktion, die aus $x_0, x_1, \ldots, x_{N-1}$ rekonstruiert wird, periodisch ist (mit Periode N). So betrachtet DFT die Folge

   $$8, 16, 24, 32, 40, 48, 56, 64$$

   als die Werte der folgenden periodischen Funktion:

   Die inverse Transformation für die „bearbeiteten Werte" des Vektores Ax (z. B. nach Quantisierung) gibt daher die Werte der folgenden Funktion:

DCT hat solche Eigenschaft nicht. Wir betrachten DCT als die ersten N reellen Werte der DFT für die Folge:

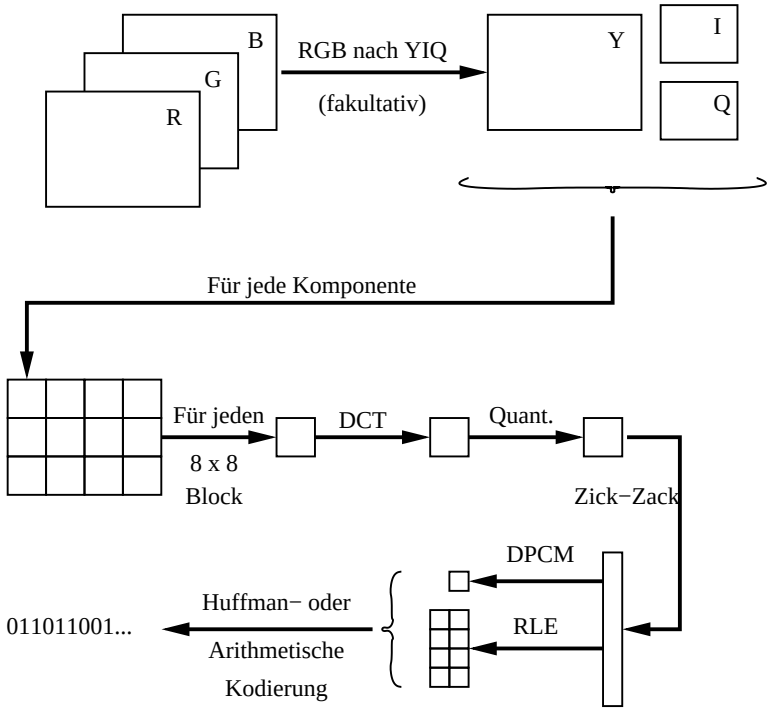
$$x_0, x_1, \ldots, x_{N-1}, x_{N-1}, \ldots, x_1, x_0$$

und deshalb entfernen wir diese Nichtstetigkeiten.

3. Die Komplexität der DCT ist höher als die der DFT.

In der Praxis benutzt Software zur Berechnung der DCT Festpunktarithmetik. Der „Weltrekord" für die schnellste Ausführung der DCT liegt bei 11 Multiplikationen und 29 Additionen [C. Loeffler, A. Ligtenberg and G. Moschytz, *Practical Fast 1-D DCT Algorithms with 11 Multiplications*, Proc. Int'l. Conf. on Acoustics, Speech, and Signal Processing 1989 (ICASSP '89), pp. 988-991].

JPEG-Standard schematisch



Die **wichtigsten Bestandteile von JPEG** sind folgende:

- diskrete Cosinus-Transformation DCT,
- Quantisierung,
- Differentialcodierung auf DC-Komponenten,
- Zickzack-Abtastung der AC-Komponenten,
- Lauflängencodierung (RLE) auf AC-Komponenten sowie
- Huffman-Codierung oder arithmetische Codierung.

JPEG: Die Schritte im Einzelnen

optional: **Umwandlung von RGB nach YIQ**.

RGB und YIQ sind dreidimensionale Räume, die die *Farben* und die *Helligkeit* beschreiben.

RGB: Format für die Hardware,
YIQ: entspricht der menschlichen Wahrnehmung

Erklärung der Komponenten:

R: Rotanteil	I: Ausgleich zwischen Orange & Cyan
G: Grünanteil	Q: Ausgleich zwischen Grün & Magenta.
B: Blauanteil	I und Q zusammen: *Chrominanzsignal*.
	Y: Pixel-Helligkeit: *Luminanzsignal*.

Konversion RGB $\Rightarrow$ YIQ:
Wieviele Bits kommen den YIQ-Komponenten zu ?

Im Allgemeinen: viermal soviel Bits für die Y- wie für die I- oder die Q-Komponente
(Die Augen sind empfindlicher für Änderungen der Helligkeit als für Änderungen der Farbe.)

YIQ wurde in den USA entwickelt für den Übergang vom Schwarz-Weiß zum Farbfernsehen (NTSC).

Ähnliche Formate verwenden PAL, SECAM und beim Digitalfernsehen.

Teile das Bild in Pixelblöcke der Größe 8×8:

Beispiel: Ein 8×8-Block aus dem Sena-Bild (Buch von Sayood). . .

124	125	122	120	122	119	117	118
121	121	120	119	119	120	120	118
126	124	123	122	121	121	120	120
124	124	125	125	126	125	124	124
127	127	128	129	130	128	127	125
143	142	143	142	140	139	139	139
150	148	152	152	152	152	150	151
156	159	158	155	158	158	157	156

Führe DCT für jeden Block aus

. . . liefert diese DCT-Koeffizienten nach JPEG:

39,88	6,56	-2,24	1,22	-0,37	-1,08	0,79	1,13
-102,43	4,56	2,26	1,12	0,35	-0,63	-1,05	-0,48
37,77	1,31	1,77	0,25	-1,50	-2,21	-0,10	0,23
-5,67	2,24	-1,32	-0,81	1,41	0,22	-0,13	0,17
-3,37	-0,74	-1,75	0,77	-0,62	-2,65	-1,30	0,76
5,98	-0,31	-0,45	-0,77	1,99	-0,26	1,46	0,00
3,97	5,52	2,39	-0,55	-0,051	-0,84	-0,52	-0,13
-3,43	0,51	-1,07	0,87	0,96	0,09	0,33	0,01

Führe DCT für jeden Block aus

Beachte dabei folgenden Trick:

Jeder Pixelwert von 0 bis $2^P - 1$ wird (vor Ausführung der DCT) durch Subtraktion von 2^{P-1} in einen um den Nullpunkt symmetrischen Bereich abgebildet.

Warum benutzt JPEG DCT und nicht DFT?

Erinnerung:
Pro DCT: (nur) reelle Zahlen; keine Periodizitätsannahme
Contra DCT: Rechenzeit

Das JPEG-Kommittee hat DCT gewählt.
Hauptgrund: Man kann empirisch zeigen, dass die Pixel eines Bildes nicht einer periodischen Funktion entsprechen und daher der Grundannahme der DFT widersprechen.

Quantisierung

Die Standard-Quantisiermatrix von JPEG für Helligkeit

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

Die Matrix $\ell_{i,j}$ der Quantisierlabels des Sena-Blocks...

2	1	0	0	0	0	0	0
-9	0	0	0	0	0	0	0
3	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	0

...und die zugehörigen Repräsentanten

32	11	0	0	0	0	0	0
-108	0	0	0	0	0	0	0
42	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	0

Der Quantisierfehler ist die Hauptquelle der Verzerrung in JPEG. Das Verfahren benutzt eine 8×8 Quantisiermatrix Q und das heißt, dass JPEG für jeden 8×8-Block X die Werte

$$\ell_{i,j} = \lfloor \Theta_{i,j}/Q_{i,j} + 0,5 \rfloor$$

berechnet, wobei Θ die Matrix X nach der diskreten Cosinus-Transformation ist.

Im Beispiel des Sena-Bildblocks ist $\Theta_{0,0} = 39,88$, d.h.

$$\ell_{0,0} = \left\lfloor \frac{39,88}{16} + 0,5 \right\rfloor = \lfloor 2,9925 \rfloor = 2.$$

Der zugehörige Repräsentant ist daher $32 = 16 * 2$, und der Quantisierfehler wäre hier $7,88$.

JPEG definiert zwei Default-Quantisiermatrizen: die gezeigte für Luminanz und eine andere für Chrominanz.

Es ist möglich, in JPEG eine eigene Quantisiermatrix definieren. Dann fügt man diese Matrix in den Kopf des codierten Bildes.

Ein weiteres Beispiel

Unten sehen Sie die Werte nach der Quantisierung für den Block „Brücke" und die Default-Matrix Q:

$$\Theta = \begin{bmatrix} 1049,9 & -125,7 & -121,2 & 4,7 & 50,1 & -40,0 & 2,4 & 55,2 \\ -30,4 & 59,0 & -48,7 & 6,0 & -0,2 & -35,8 & -22,7 & 13,3 \\ -17,2 & 5,7 & 12,8 & 16,4 & 24,6 & -11,0 & 0,6 & -5,4 \\ 15,7 & -14,8 & -17,5 & -9,4 & 25,7 & 26,1 & -30,5 & -10,9 \\ -19,9 & 5,6 & -0,6 & 14,8 & -13,1 & 22,2 & 14,5 & 12,3 \\ -30,6 & -3,6 & 13,5 & 17,2 & -8,3 & -18,8 & 20,3 & 19,7 \\ 11,1 & -6,8 & 2,1 & -19,6 & 4,3 & 5,6 & -16,0 & 13,4 \\ 0,5 & 9,5 & -7,4 & 2,8 & 4,0 & 1,0 & 5,5 & -0,2 \end{bmatrix}$$

$$\rightsquigarrow Q \text{ komponentenweise mal } \begin{bmatrix} 66 & -11 & -12 & 0 & 2 & -1 & 0 & 1 \\ -3 & 5 & -3 & 0 & 0 & -1 & 0 & 0 \\ -1 & 0 & 1 & 1 & 1 & 0 & 0 & 0 \\ 1 & -1 & -1 & 0 & 1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

DPCM auf DC-Komponenten

DC-Komponenten sind groß und haben unterschiedliche Werte.
Man kann aber sehen, dass viele DC-Komponenten ähnliche Werte wie ihre Vorgänger haben.
Deshalb wählt man anstelle der direkten Codierung der DC-Komponenten ein DPCM-Verfahren:
JPEG codiert die Differenzen zwischen Nachbarkomponenten.
Diese Differenzen haben die Tendenz, klein zu sein.

Die Differenzen komprimiert man mit Hilfe des Huffman-Verfahrens oder der arithmetischen Codierung.

Zickzack-Abtastung der AC-Komponenten

Hierdurch wird der zweidimensionale Block in einen eindimensionalen Vektor umgewandelt, so dass diese Konversion die Energie des Blockes im Anfang des Vektores bündelt.

Lauflängencodierung auf AC-Komponenten

Man kann sehen, dass viele AC-Komponenten Null sind.
JPEG codiert diese Werte als Paare $(skip, value)$, wobei $skip$ die Anzahl der Nullen und $value$ die nächste Nicht-Null-Komponente ist.
$(0, 0)$ steht für das Ende.
Dann komprimiert JPEG diese Folgen mit Hilfe des Huffman-Verfahrens oder der arithmetischen Codierung.

Am Schluss. . . Rekonstruktion des Sena-Blocks

123	122	122	121	120	120	119	119
121	121	121	120	119	118	118	118
121	121	120	119	119	118	117	117
124	124	123	122	122	121	120	120
130	130	129	129	128	128	128	127
141	141	140	140	139	138	138	137
152	152	151	151	150	149	149	148
159	159	158	157	157	156	155	155

Vergleich mit dem Original

Original:

124	125	122	120	122	119	117	118
121	121	120	119	119	120	120	118
126	124	123	122	121	121	120	120
124	124	125	125	126	125	124	124
127	127	128	129	130	128	127	125
143	142	143	142	140	139	139	139
150	148	152	152	152	152	150	151
156	159	158	155	158	158	157	156

$\rightsquigarrow$

123	122	122	121	120	120	119	119
121	121	121	120	119	118	118	118
121	121	120	119	119	118	117	117
124	124	123	122	122	121	120	120
130	130	129	129	128	128	128	127
141	141	140	140	139	138	138	137
152	152	151	151	150	149	149	148
159	159	158	157	157	156	155	155