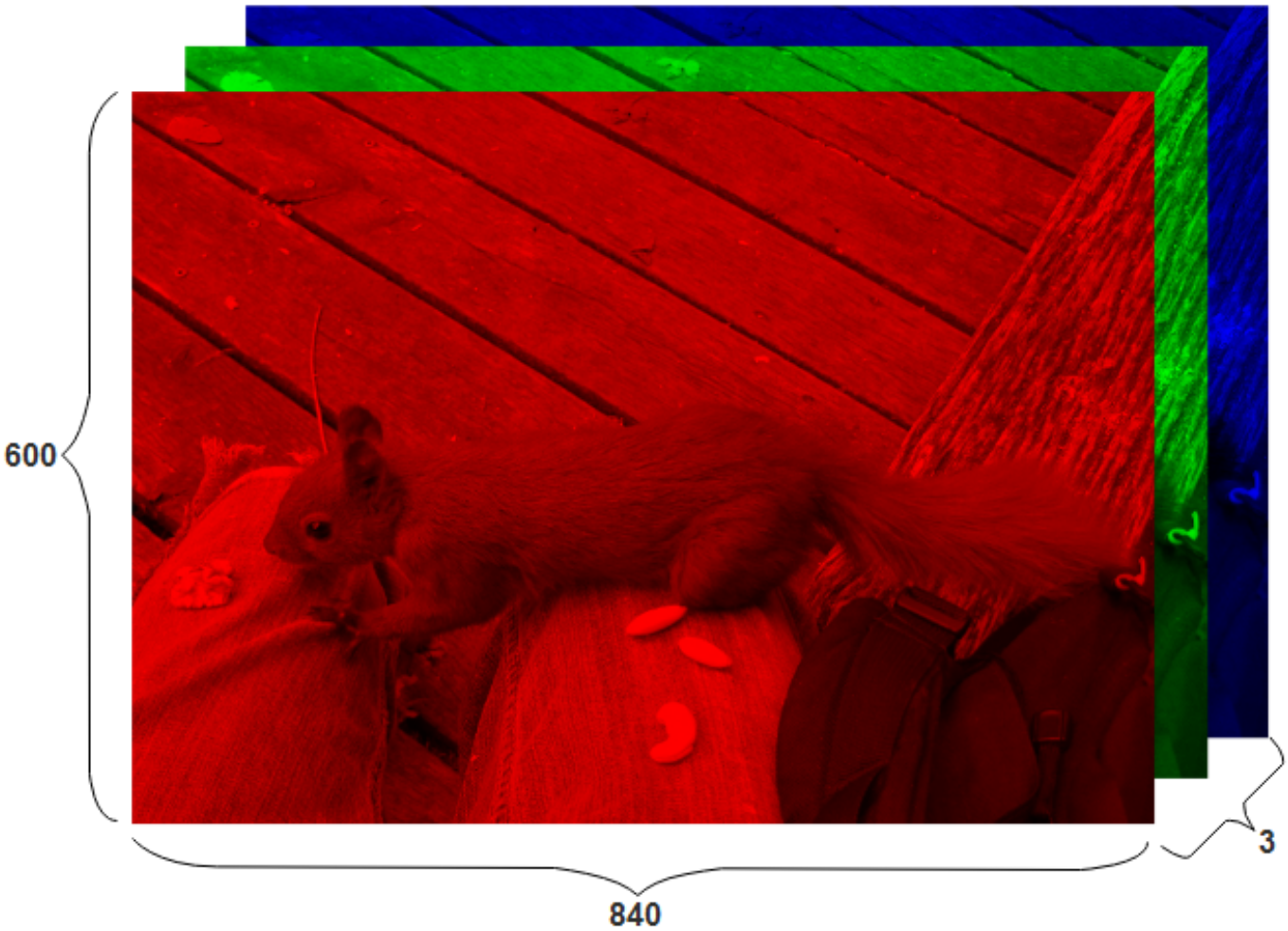
Bilddatenkompression

Hannes Benne

2. April 2017

Das Signal

Das Signal



600 x 840 x 3 = 1.512.000 8-bit-Zahlen.

Idee

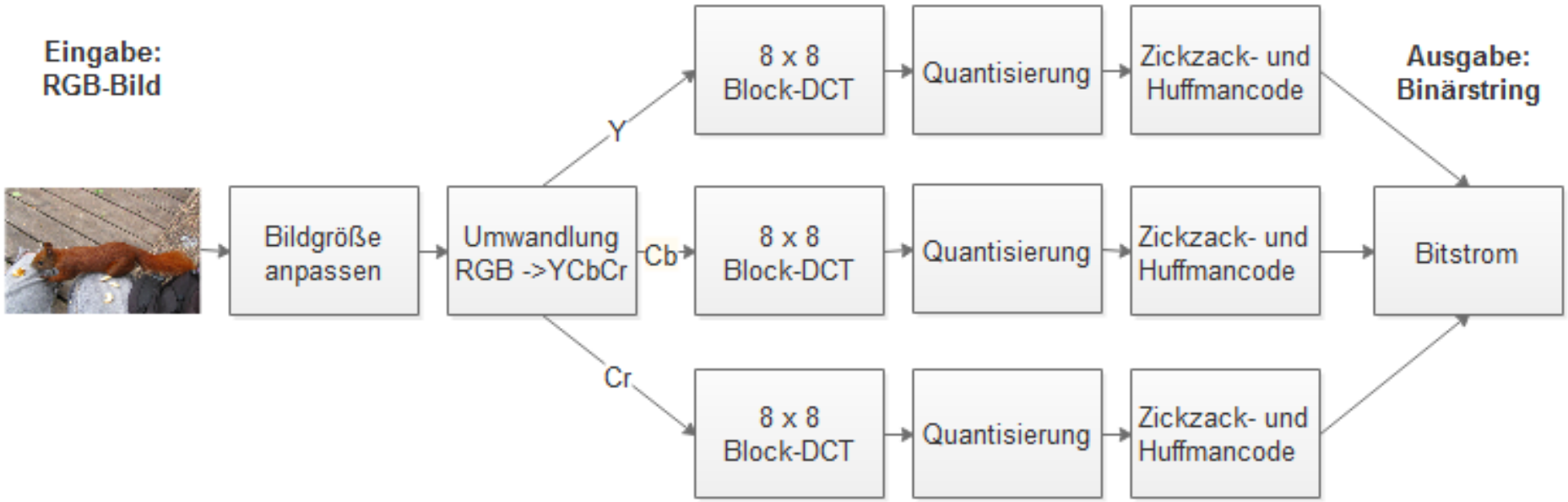
Kosinustransformation

- Bild mit 2-D Kosinustransformation in Spektralbereich transformieren.
- Konzentration der Information in wenigen DCT-Koeffizienten.
- Koeffizienten hochfrequenter Basisfunktionen Null setzen.

Huffmankodierung

- Ordnet jedem DCT-Koeffizienten ein Codewort variabler Länge zu.
- Länge der Codewörter hängt vom Informationsgehalt ab.

Ablaufplan


Eingabe:
RGB-Bild
Bildgröße anpassen
Umwandlung RGB ->YCbCr
Y
Cb
Cr
8 x 8 Block-DCT
8 x 8 Block-DCT
8 x 8 Block-DCT
Quantisierung
Quantisierung
Quantisierung
Zickzack- und Huffmancode
Zickzack- und Huffmancode
Zickzack- und Huffmancode
Ausgabe:
Binärstring
Bitstrom

Bildgröße anpassen

Für spätere Arbeitsschritte muss das Bild eine durch 8 teilbare Höhe und Breite haben $\rightarrow$ Bild erweitern oder zuschneiden.

```
1 function img = crop(img)
2 [h,b,~] = size(img);
3 img = img(1:h-mod(h,8),1:b-mod(b,8),:);
4 end
```

RGB $\rightarrow$ YCbCr

- Luminanz Y (Helligkeit) und zwei Chrominanz Cb (Blauabweichung von Y), Cr (Rotabweichung von Y) Komponenten.
- Menschen können Helligkeitsunterschiede besser wahrnehmen als Farbunterschiede $\rightarrow$ Cb- und Cr- Komponente können stärker komprimiert werden.
- Umwandlung in Matrixschreibweise:

$$\begin{pmatrix} Y \\ Cb \\ Cr \end{pmatrix} = \begin{pmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.500 \\ 0.500 & -0.419 & -0.081 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

- Matlab: YCbCr_Bild = rgb2ycbcr(RGB_Bild)

Diskrete 2D-Kosinustransformation

Algorithms

The discrete cosine transform (DCT) is closely related to the discrete Fourier transform. It is a separable linear transformation; that is, the two-dimensional transform is equivalent to a one-dimensional DCT performed along a single dimension followed by a one-dimensional DCT in the other dimension. The definition of the two-dimensional DCT for an input image A and output image B is

$$B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}, \quad \begin{matrix} 0 \le p \le M-1 \\ 0 \le q \le N-1 \end{matrix}$$

where

$$\alpha_p = \begin{cases} \frac{1}{\sqrt{M}}, & p = 0 \\ \sqrt{\frac{2}{M}}, & 1 \le p \le M-1 \end{cases}$$

and

$$\alpha_q = \begin{cases} \frac{1}{\sqrt{N}}, & q = 0 \\ \sqrt{\frac{2}{N}}, & 1 \le q \le N\text{-}1 \end{cases}$$

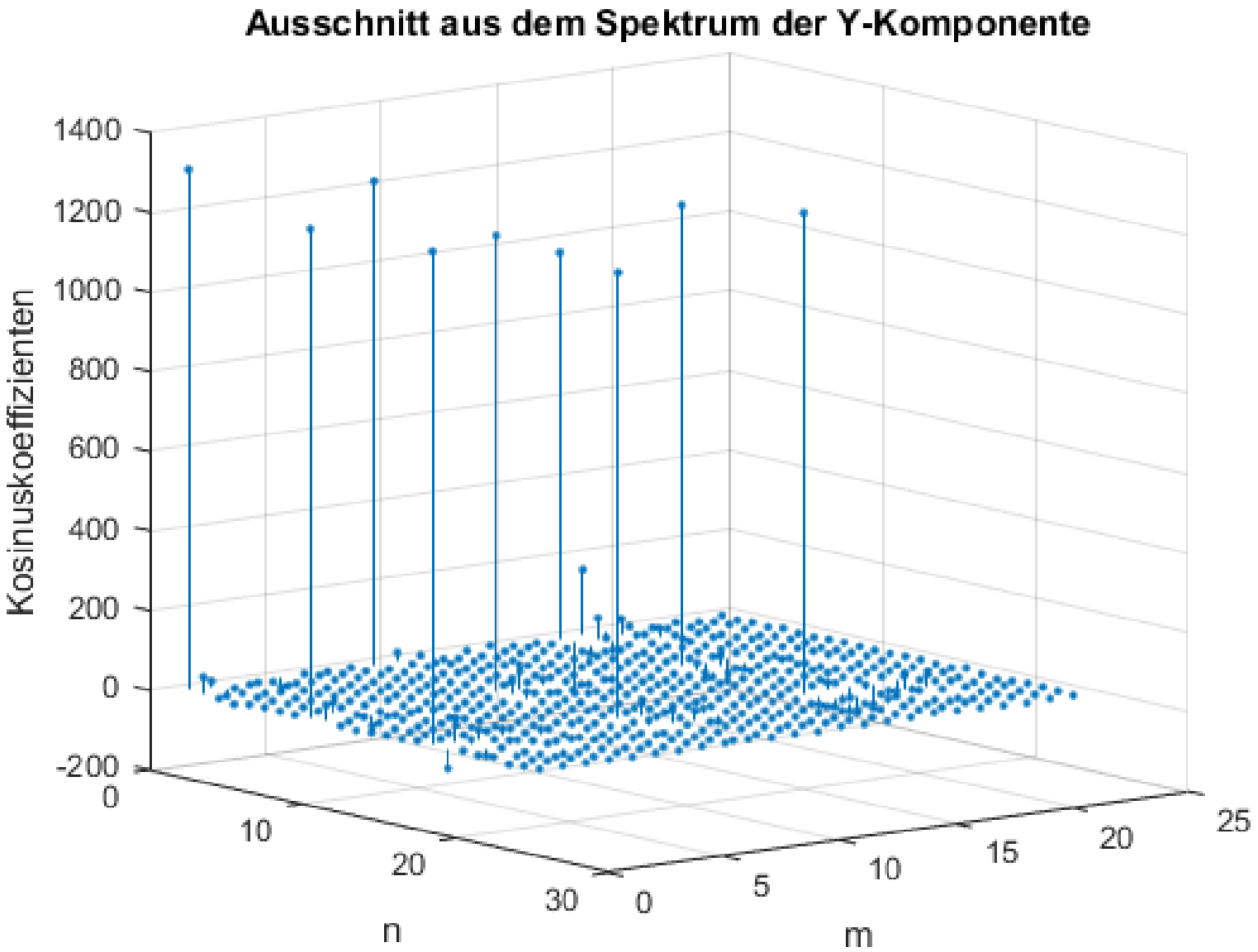
Abbildung: https://de.mathworks.com/help/images/ref/dct2.html

Diskrete 2D-Kosinustransformation

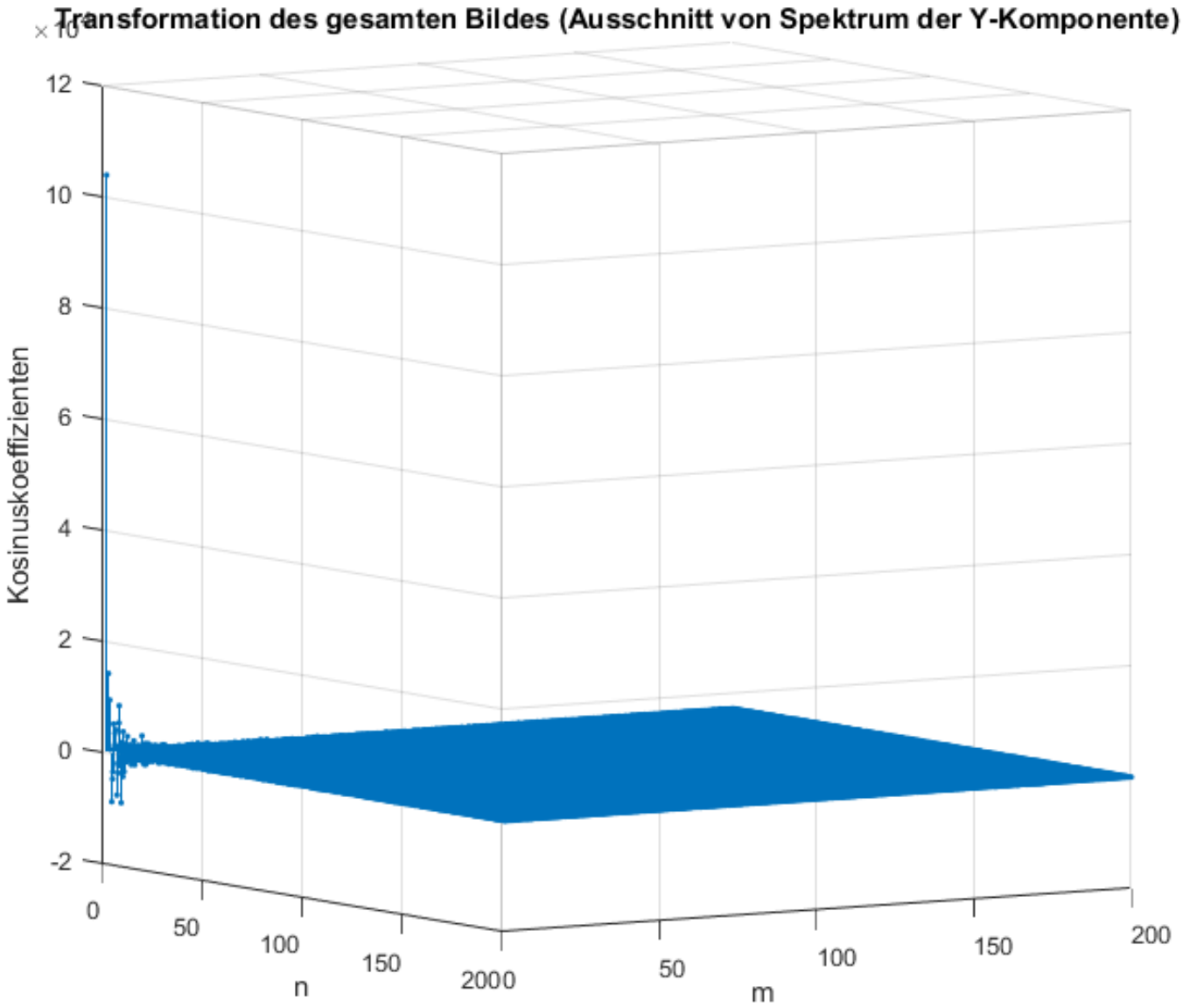
- Transformation von 8 x 8 Pixel großen Blöcken.
- Matlab: dct2(img)
- Besser/ schneller Transformation in Matrixschreibweise.

```
1  function img = blockDCT(img)
2  [h, b] = size(img);
3  img = double(img);
4  DCT = dctmtx(8);
5  for m = 1:8:h
6      for n = 1:8:b
7          img(m:m+7,n:n+7) = DCT*img(m:m+7,n:n+7)*DCT';
8      end
9  end
10 end
```

Diskrete 2D-Kosinustransformation



Diskrete 2D-Kosinustransformation



Quantisierung

- Weniger einflussreiche Koeffizienten werden durch elementweise Division mit einer Quantisierungsmatrix Q und Runden auf Null gesetzt.
- Berechnung von Q: $Q_{m,n} = 1 + (a(1 + m + n))$.
- Je größer a ist, desto mehr Koeffizienten werden zu Null und desto stärker wird das Bild komprimiert.

$$\mathbf{Q} = \begin{pmatrix} 16 & 21 & 26 & 31 & 36 & 41 & 46 & 51 \\ 21 & 26 & 31 & 36 & 41 & 46 & 51 & 56 \\ 26 & 31 & 36 & 41 & 46 & 51 & 56 & 61 \\ 31 & 36 & 41 & 46 & 51 & 56 & 61 & 66 \\ 36 & 41 & 46 & 51 & 56 & 61 & 66 & 71 \\ 41 & 46 & 51 & 56 & 61 & 66 & 71 & 76 \\ 46 & 51 & 56 & 61 & 66 & 71 & 76 & 81 \\ 51 & 56 & 61 & 66 & 71 & 76 & 81 & 86 \end{pmatrix}$$

Abbildung: Quantisierungsmatrix für a=5

Quantisierung

```
1  function img = quantization(img,a)
2  Q = zeros(8,8);
3  for m = 1:8
4      for n = 1:8
5          Q(m,n)=1+a*(m+n+1);
6      end
7  end
8  [h, b] = size(img);
9  for m = 1:8:h
10     for n = 1:8:b
11         img(m:m+7,n:n+7) = round(img(m:m+7,n:n+7)./Q);
12     end
13 end
14 end
```

Beispiel

$$\begin{pmatrix} 141 & 143 & 148 & 150 & 152 & 150 & 151 & 142 \\ 135 & 138 & 148 & 154 & 152 & 151 & 151 & 147 \\ 108 & 123 & 139 & 141 & 144 & 147 & 151 & 150 \\ 88 & 103 & 119 & 135 & 146 & 148 & 148 & 147 \\ 84 & 87 & 97 & 118 & 141 & 139 & 143 & 150 \\ 79 & 80 & 87 & 95 & 110 & 127 & 136 & 146 \\ 72 & 73 & 79 & 93 & 99 & 109 & 125 & 143 \\ 61 & 66 & 78 & 90 & 99 & 100 & 105 & 116 \end{pmatrix}$$

Bildausschnitt aus der Y-Komponente

2D-DCT

$$\begin{pmatrix} 977.1250 & -127.9211 & -20.8742 & -7.4874 & 4.8750 & -1.0569 & -0.4187 & 2.0523 \\ 162.8486 & 55.2582 & -19.8293 & 2.0257 & -9.9175 & 4.2666 & -1.8773 & 1.9779 \\ -16.0895 & 38.0781 & 10.4043 & -3.2185 & 0.1209 & 4.1490 & 0.5240 & -0.6493 \\ 5.2009 & -6.4569 & 16.5723 & 10.4362 & 2.9922 & 0.1936 & -0.0853 & 0.1953 \\ -5.8750 & -0.1140 & -16.9081 & 6.2685 & 3.8750 & -2.9565 & -2.6027 & 2.2111 \\ -1.6264 & -4.9024 & 4.5773 & -4.8294 & -3.5649 & 1.0096 & -0.8740 & -1.6463 \\ -7.8125 & 1.6031 & 1.7740 & -0.0943 & -7.2209 & -1.0894 & 2.0957 & 1.5555 \\ 1.0318 & -2.5451 & 2.3904 & -1.5773 & 3.7640 & -1.2672 & -1.6790 & 1.7960 \end{pmatrix}$$

zugehöriges DCT-Spektrum

Multiplikation mit Q (a=5)

$$\begin{pmatrix} 61 & -6 & -1 & 0 & 0 & 0 & 0 & 0 \\ 8 & 2 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

quantisiertes DCT-Spektrum

Beispiel

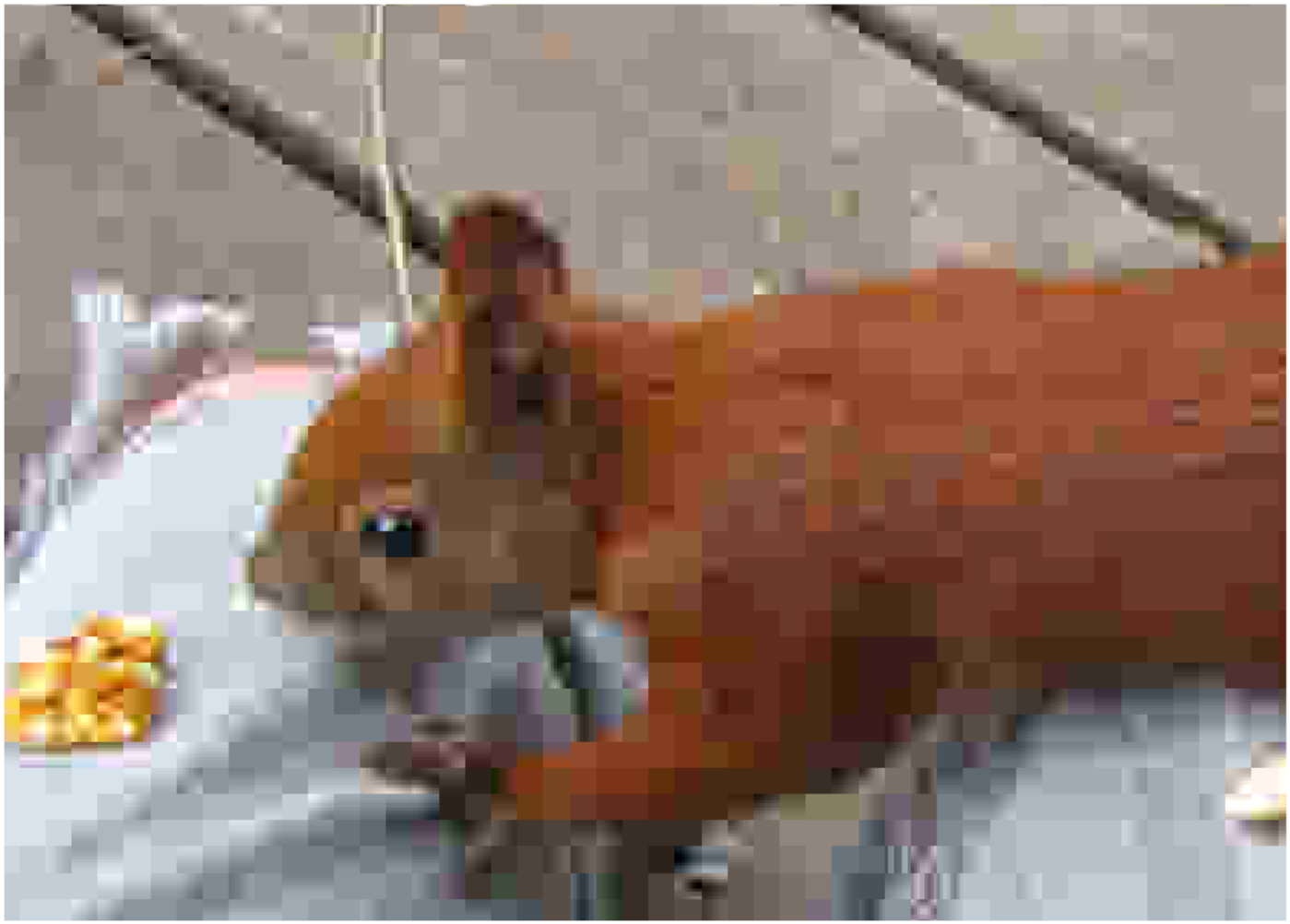
Für das gesamte Eichhörnchenbild (Kompression mit $a = 5$).

600 x 840 x 3 = 1.512.000 8-bit-Zahlen $\overset{\text{DCT}\ ,\ ./\text{Q}}{\longrightarrow}$ 69.455 8-bit-Zahlen ungleich Null.

Abbildung: Rücktransformation mit $\approx 4,6\%$ der Koeffizienten.

Beispiel

Bei starker Kompression (bspw. a=25) treten Kompressionsartefakte auf.



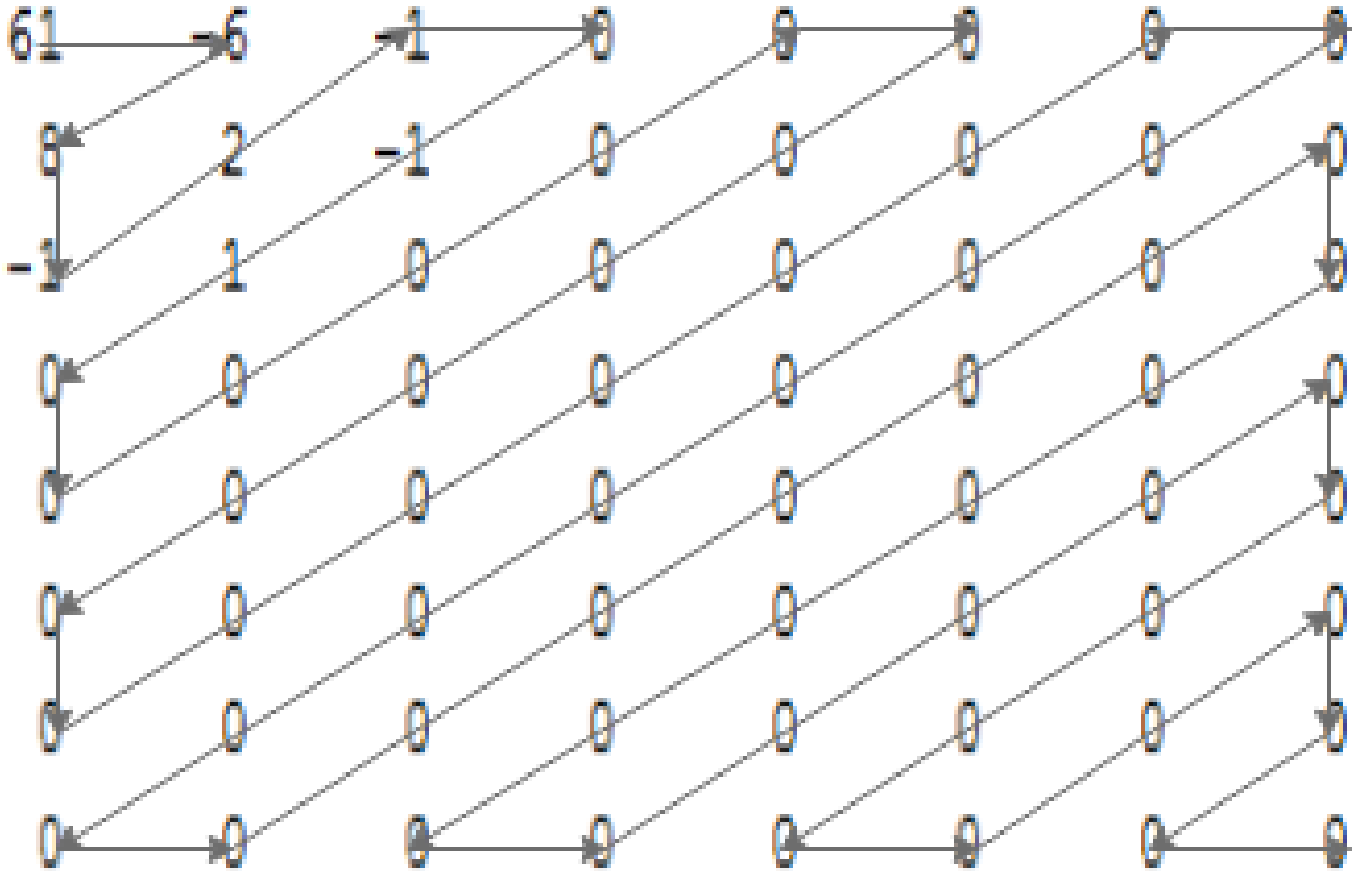
Beispiel

Nur die Y Komponente komprimiert (a=25)

Beispiel

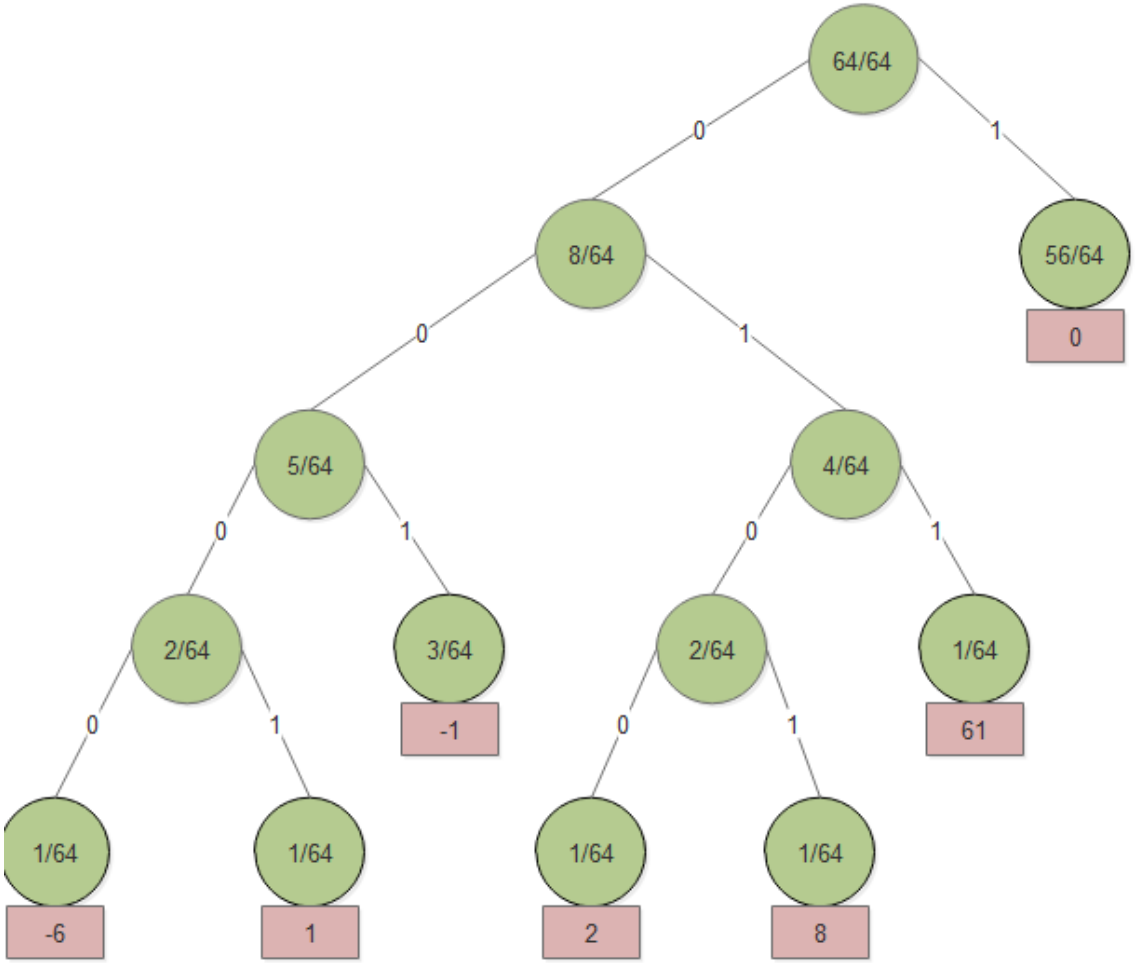
Nur die Cb Komponente komprimiert (a=25)

Zickzack-Auslesen der Matrizen



Das führt auf den Vektor: $[61, -6, 8, -1, 2, -1, 0, -1, 1, 0, 0, \ldots]$.

Huffmanbaum



Koeffizient	relative Häufigkeit	Codewort
-6	1/64	0000
1	1/64	0001
-1	3/64	001
0	56/64	1
2	1/64	0100
8	1/64	0101
61	1/64	011

Vektor: 61,-6,8, -1,2,-1,0,-1,1,0,0,0,...

Binärcode mit 7 bit pro Zeichen -> 448 Bit.

Durchnummerieren: 3 Bit pro Zeichen -> 192 Bit.

Huffmancode: 011000001010001000011001000111111111... (81 Bit)

Decodierung

Um aus dem Binärstring wieder ein Bild zu erhalten, werden die folgenden Schritte ausgeführt:

- Huffman-Decodierung
- Koeffizientenvektor wieder nach Zickzack-Muster in Matrix anordnen.
- Blockweise mit Quantisierungsmatrix Q multiplizieren.
- Blockweise inverse Kosinustransformation anwenden.
- Farbkomponenten zu Gesamtbild anordnen.
- Umwandlung von YCbCr nach RGB.