

# Grundlagen der Multimediaetechnik

## Bildkomprimierung

10.12.2021, Prof. Dr. Enkelejda Kasneci



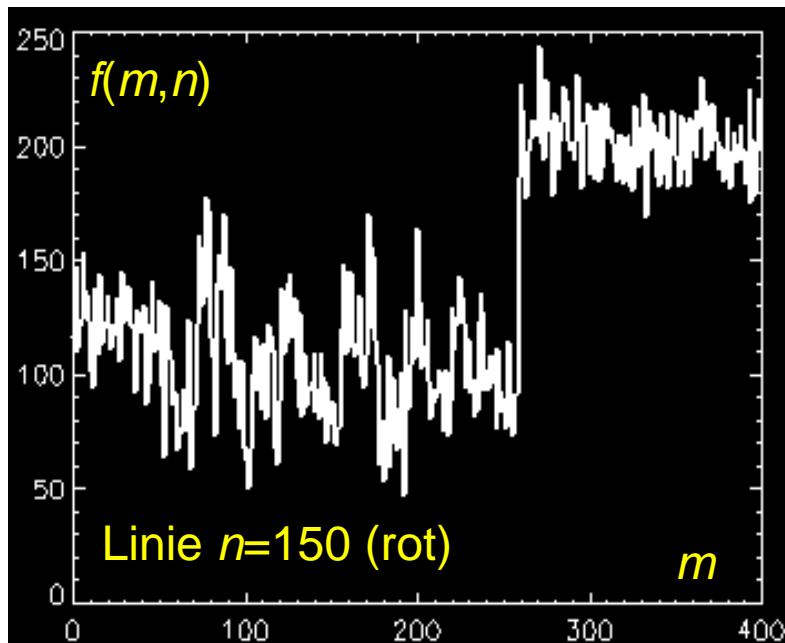
# Termine und Themen

22.10.2021	Einführung
29.10.2021	Menschliche Wahrnehmung – visuell, akustisch, haptisch, ...
05.11.2021	Informationstheorie, Textcodierung und -komprimierung
12.11.2021	Bildverbesserung
19.11.2021	Bildanalyse
26.11.2021	Grundlagen der Signalverarbeitung
03.12.2021	Bildkomprimierung
10.12.2021	Videokomprimierung
17.12.2022	Audiokomprimierung
14.01.2022	Videoanalyse
21.01.2022	Dynamic Time Warping
28.01.2022	Gestenanalyse
04.02.2022	Tiefendatengenerierung
11.02.2022	FAQ mit den Tutoren
18.02.2022	Klausur

# Wiederholung: Bild- und Videosignale

Zeilenausschnitt aus einem Schwarzweißbild

- Analogtechnik: Zeile enthält kontinuierliches Signal (z.B. PAL-TV)
- Digitaltechnik: Zeilen diskretisiert → Pixel (von „picture element“).





# Digitales Foto

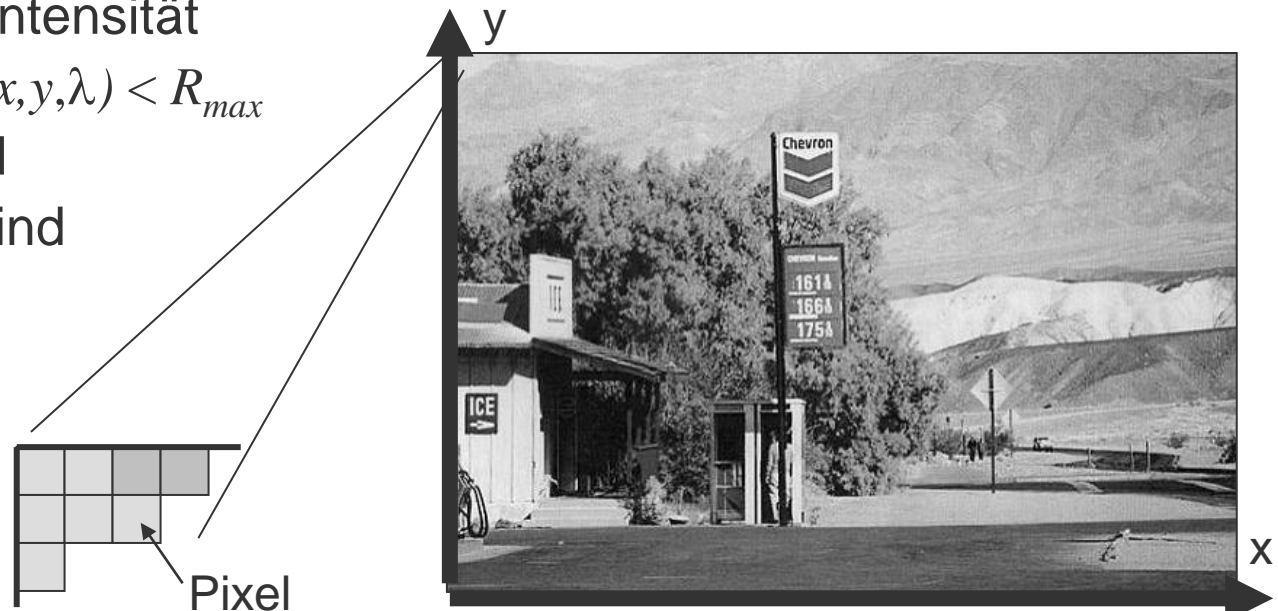
- Das Bild besteht aus einer Menge von Bildelementen (**Pixel** von „**pic**ture **el**ement“).
- Definitionsbereich: Ausdehnung in x- und y-Richtung, sowie Wellenlänge  $\lambda$ :

$$x_{\min} \leq x < x_{\max}, y_{\min} \leq y < y_{\max}, \lambda_{\min} \leq \lambda < \lambda_{\max}$$

- Wertebereich: Intensität

$$R_{\min} \leq R(x, y, \lambda) < R_{\max}$$

- Definitions- und Wertebereich sind beschränkt.

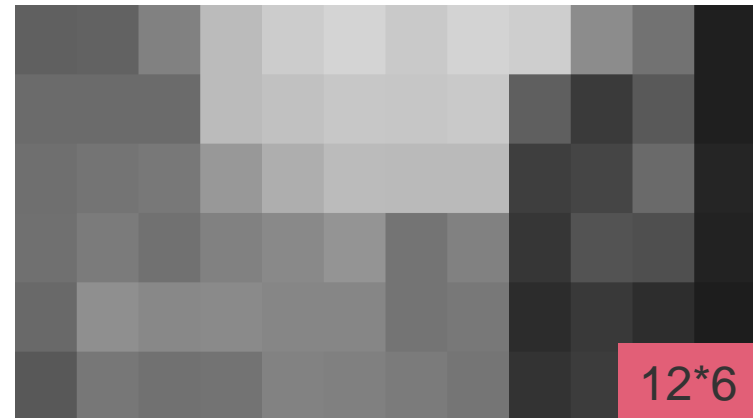
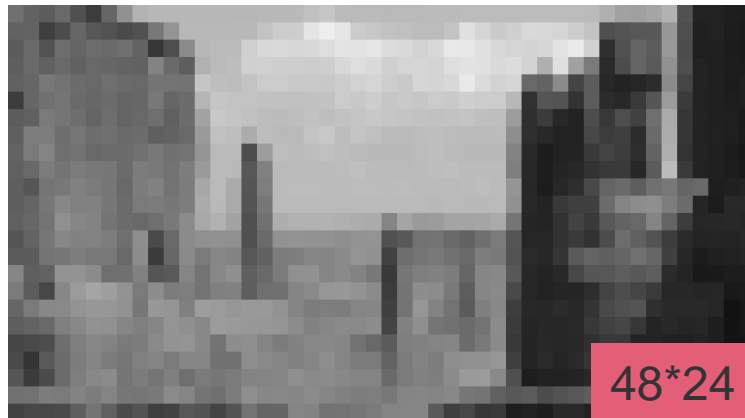


Quelle: Klaus Tönnies: Grundlagen der Bildverarbeitung, 2005

# Ortsauflösung (Abtastung)

←  $M$  →

↑  
 $N$   
↓



$$m = \left\lfloor M \cdot \frac{x - x_{\min}}{x_{\max} - x_{\min}} \right\rfloor \text{ und } n = \left\lfloor N \cdot \frac{y - y_{\min}}{y_{\max} - y_{\min}} \right\rfloor \text{ für } x_{\min} \leq x < x_{\max}, y_{\min} \leq y < y_{\max}$$

Quelle: Klaus Tönnies: Grundlagen der Bildverarbeitung, 2005



# Kontrastauflösung (Quantisierung)



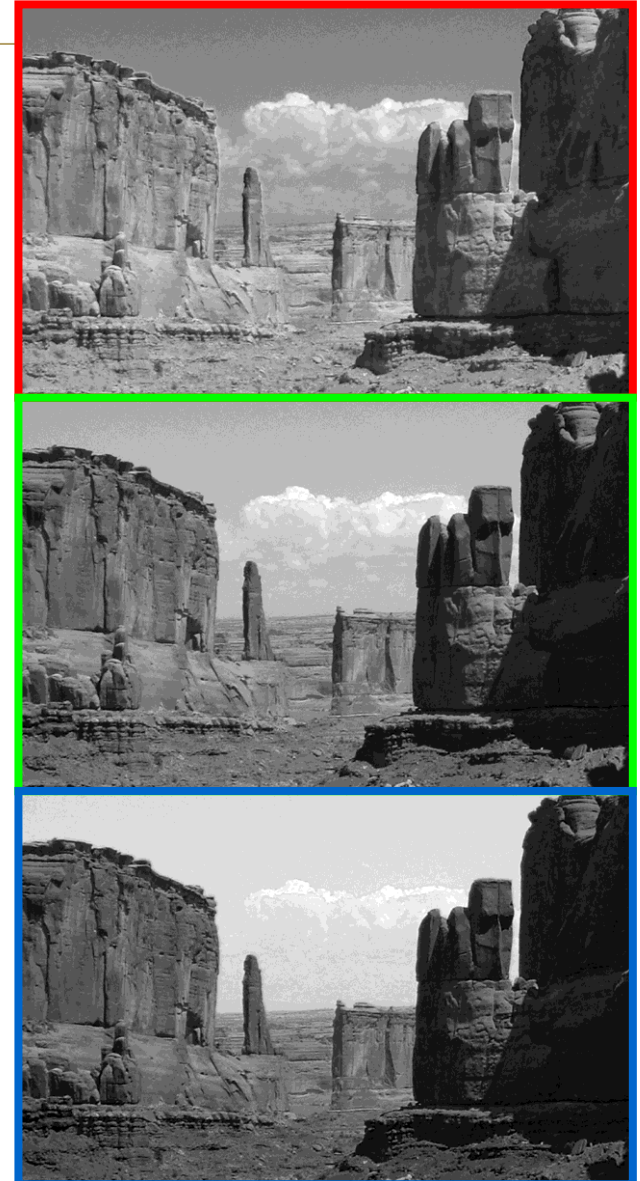
Reduzierung der Kontrastauflösung mindert die Erkennbarkeit weniger als die der Ortsauflösung

Quelle: Klaus Tönnies: Grundlagen der Bildverarbeitung, 2005



## RGB Farbmodell

- Jede Farbe wird durch ein Tripel (rot, grün, blau) repräsentiert.





# Farbbildzerlegung in unterschiedliche Farbkanäle (RGB und YCrCb)







# Bildkompression

## Motivation

- **Erkenntnis: Echtbilder schwer verlustfrei zu komprimieren**
    - Korrelation über weiteren Bereich (2D)
    - Farbverläufe
    - Fein aufgelöste Strukturen
    - Verlustfreie Methoden bei maximal 50%  
→ bei mehreren MB pro Bild kein wesentlicher Unterschied
  - **Skalierbares Kompressionsformat**
    - Starke Orientierung an Dateninhalt
    - Verlustfreie Kompression, große Dateien
    - Verlustbehaftete Kompression, kleinere Dateien
    - Einstellbarer Kompressionsgrad
- **JPEG**



## Bildkompression (forts.) - JPEG

- **Joint Pictures Experts Group**
  - Wohldefiniertes Austauschformat für kodierte Daten
  - Definition eines allgemeinen Kompressionsschemas
  - Orientierung an professionellen Vorgaben
- **Unabhängigkeit** von
  - Bildauflösung
  - Bild- und Pixel-Seitenverhältnis
  - Farbdarstellung
  - Bildkomplexität und statistischen Eigenschaften
- **Implementierbarkeit**
  - Software
  - Eingebetteten System (Kameras)
- **Nutzung für Standbilder und Video**
  - Motion JPEG



# JPEG: Kompressionsschritte

## Bildvorverarbeitung

- Chroma-Subsampling



## Transformation in den Frequenzraum

- Diskrete Cosinustransformation



## Quantisierung

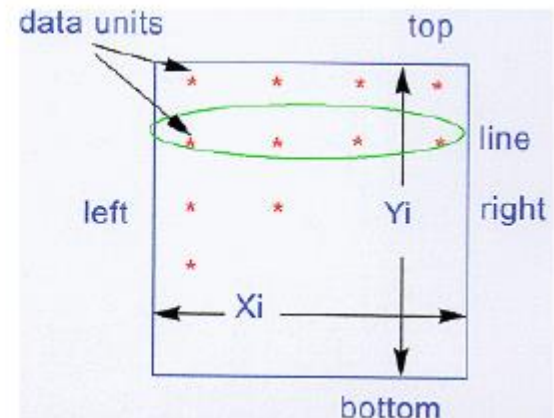
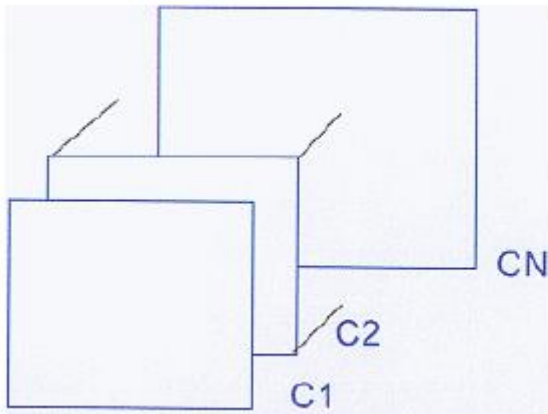
- Codierung höherer Frequenzanteile mit weniger Bits



## Entropiekodierung

- Lauflängencodierung
- Huffman-Codierung

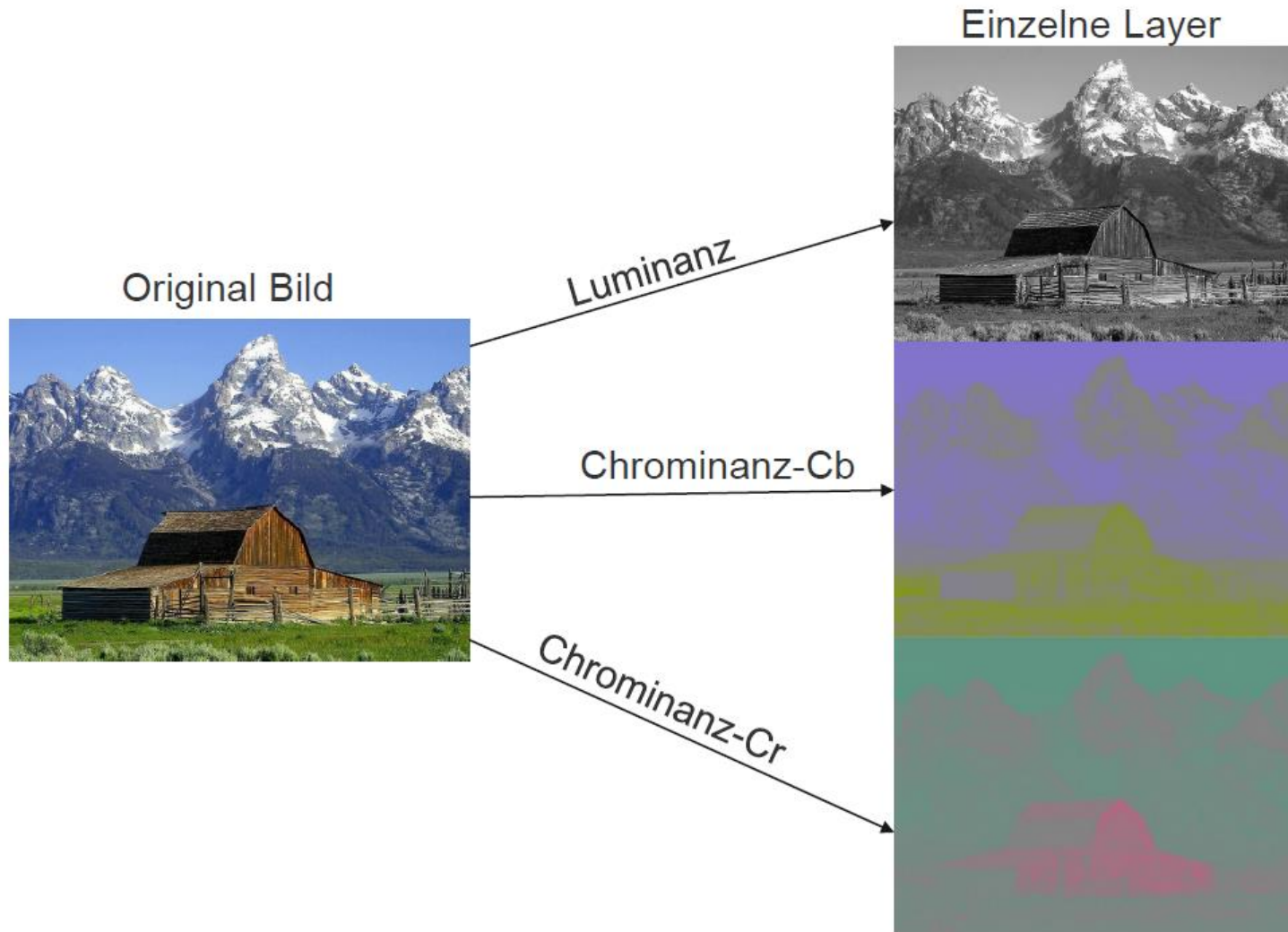
- **Dateneinheit (Data Unit) und MCUs (minimum coded units)**
  - Pro Farb-/Helligkeitskanal existieren Dateneinheiten ( $8 \times 8$  Pixel), die zu MCUs verzahnt zusammengefasst werden → am Rand wird „aufgefüllt“
  - Bei Grauwertbildern besteht eine MCU nur aus Luminanz-Dateneinheit
  - Verschiedene Auflösungen sind für die einzelnen Komponenten möglich
  - $0 \leq N \leq 255$  Komponenten pro Farb-/Helligkeitskanal  
→ Transformation in  $-128 \dots +127$  zur einfacheren DCT-Berechnung
- **Pixelauflösung**
  - **Basismodus (baseline process):** 8 Bit Tiefe, sequentiell
  - **Erweiterter Modus (extended process):** 8 oder 12 Bit Tiefe, sequentiell oder progressiv
  - **Verlustfreier Modus (lossless process):** 2 - 16 Bit Tiefe, sequentiell







# Aufteilung der Kanäle





# Chroma-Subsampling

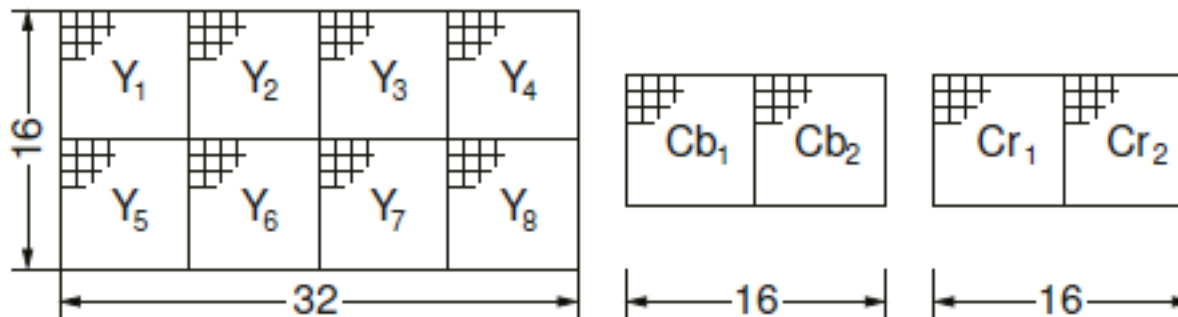
## Skalierbare Farbauflösung

- **JPEG** verwendet **4:2:0-Subsampling**

1/2 vertik. und horiz. Farbauflösung,  
Farbkomponenten mit 1/4 Abtastrate (eine Zeile auslassen)

Ein Chrominanz-Makroblock je vier Luminanz-Makroblöcke

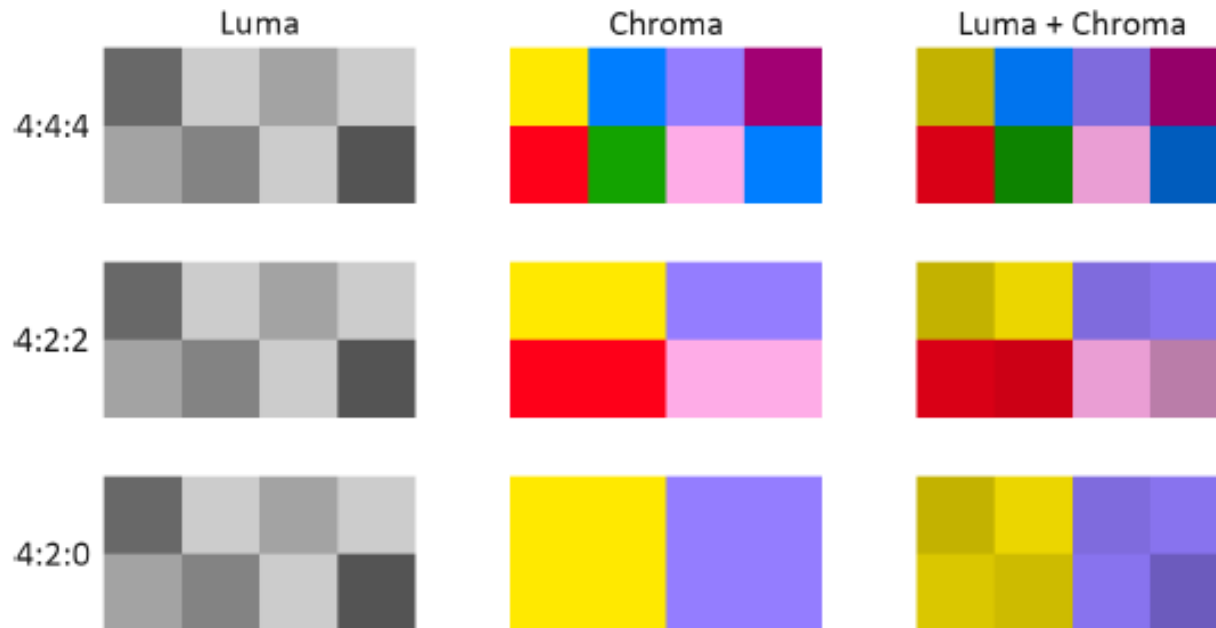
- Helligkeits- (Luminanz) und Farbdifferenzkanäle (Chrominanz Cb, Cr) werden verschachtelt dargestellt
- **Beispiel:**



4:2:0



# Chroma-Subsampling



<https://i.rtings.com/images/chroma-subsampling/subsampling.png>



# Chroma-Subsampling

4:2:0 Kompressionsfaktor 0,5



4:4:4 Original



4:2:2 (Kf = 0,66)

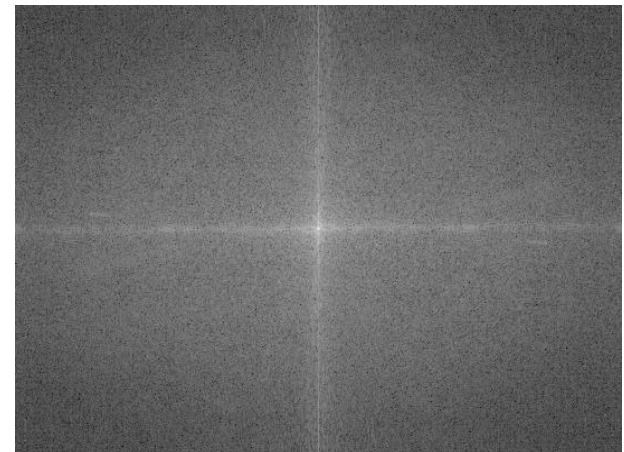






## Fouriertransformation

- Fouriertransformation: Beschreibung einer beliebigen Funktion als Summe von gewichteten periodischen Funktionen (Basisfunktionen) mit unterschiedlicher Frequenz (Frequenzraumrepräsentation).
- Anwendungen
  - Beschreibung des Informationsverlusts bei Digitalisierung
  - Restauration von linearen Störungen
  - Rekonstruktion von Bildern aus Projektionen
  - Schnelle Filterung
- Transformation muss invertierbar sein





## Basisfunktionen

- Bilder können als zweidimensionale Funktion  $f(m, n)$  aufgefasst werden.
- Jede Funktion  $f(n)$  kann als **Summe von  $N$  gewichteten Basisfunktionen  $b_u$**  aufgefasst werden:

$$f(n) = \sum_{u=0}^{N-1} w_u \cdot b_u(n)$$

- Die Gewichte  $w_u$  bilden eine neue Funktion  $w(u)$ , die  $f(n)$  zusammen mit den Basisfunktionen genau beschreibt.
- Die Auswirkung von Veränderungen auf die Gewichte (z.B. Filterung) hängen von den Basisfunktionen ab.
- Die Fouriertransformation ist die Transformation von einer Ortsbasis in eine Frequenzbasis.



## Diskrete Kosinustransformation (DCT)

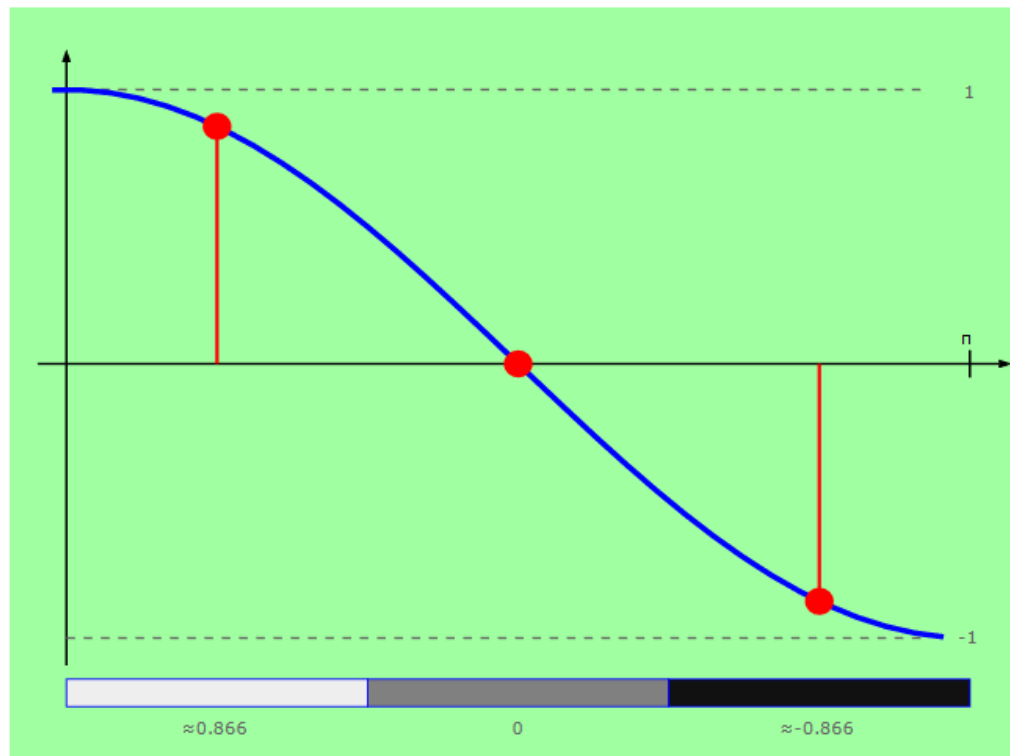
- Zerlegung in Wellen unterschiedlicher Frequenz
- Sehr ähnlich zur Fouriertransformation
- Basisfunktionen sind reell
- DCT wird üblicherweise auf quadratische Teilblöcke angewandt

**Wie lassen sich genügend viele reellwertige Basisfunktionen finden?**



# Veranschaulichung der DFT

We start by drawing the [cosine](#) function between 0 and  $\pi$ . We then *sample* the function at certain points: We divide the interval from 0 to  $\pi$  into three parts of equal size and take their centers, i.e.  $x = \frac{\pi}{6}$ ,  $x = \frac{\pi}{2}$ , and  $x = \frac{5\pi}{6}$ . Now we compute the values of the cosine function at these points. For image processing, we translate the results to different shades of gray, from black ( $-1$ ) to white ( $1$ ), so that 0 is medium gray.

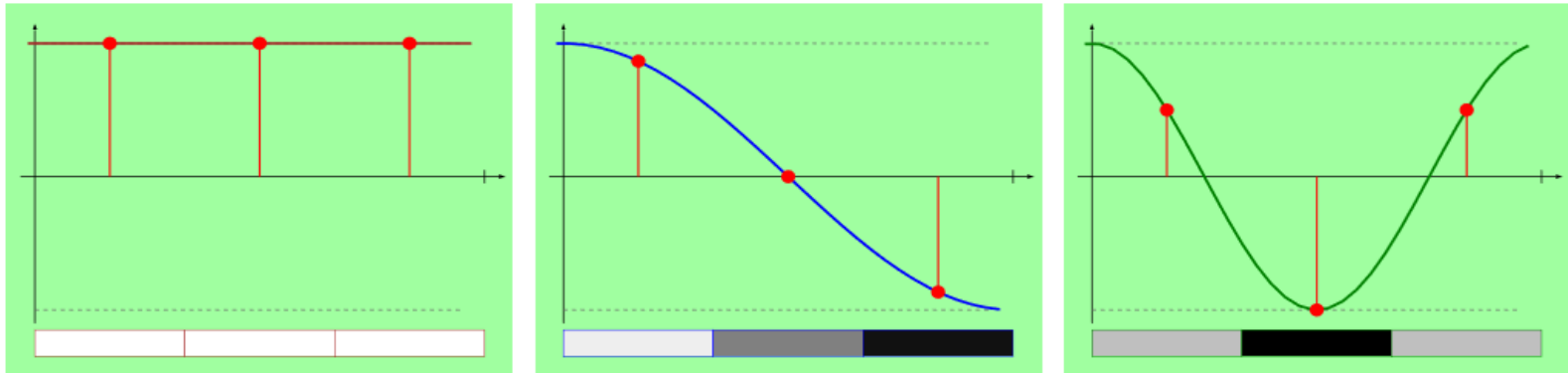



<http://weitz.de/dct/>






We not only do this with the function  $x \mapsto \cos(x) = \cos(1 \cdot x)$ , but also with  $x \mapsto \cos(2 \cdot x)$  and with  $x \mapsto \cos(0 \cdot x) = 1$ . This gives us three vectors of three numbers, which we can interpret as three rectangles, each consisting of three gray blocks:



It is now not hard to prove that these three vectors  $(1, 1, 1)$ ,  $(\frac{\sqrt{3}}{2}, 0, -\frac{\sqrt{3}}{2})$ , and  $(\frac{1}{2}, -1, \frac{1}{2})$  are linearly independent, which in turn implies that every vector of three numbers can be expressed as a linear combination of them. Or, in other words, each image consisting of three grayscale pixels can be pieced together using the three "Lego bricks" .

<http://weitz.de/dct/>



What does that mean? Suppose we want to "build" the "picture" , which has the gray values **0.9294**, **0.2000**, and **0.0667** (from left to right). We want to express it as  $\begin{bmatrix} \square & \square & \square \end{bmatrix} = \alpha \cdot \begin{bmatrix} \square & \square & \square \end{bmatrix} + \beta \cdot \begin{bmatrix} \square & \square & \square \end{bmatrix} + \gamma \cdot \begin{bmatrix} \square & \square & \square \end{bmatrix}$ .

Using numbers, this yields a simple [system of linear equations](#)

$$0.9294 = \alpha + \frac{\sqrt{3}}{2}\beta + \frac{1}{2}\gamma$$

$$0.2000 = \alpha + \quad - \gamma$$

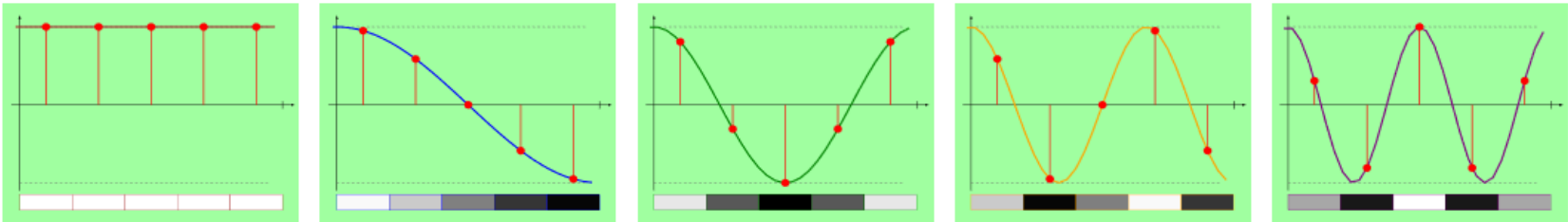
$$0.0667 = \alpha - \frac{\sqrt{3}}{2}\beta + \frac{1}{2}\gamma$$

with solutions  $\alpha \approx 0.3987$ ,  $\beta \approx 0.4981$ , and  $\gamma \approx 0.1987$ .

Our "graphical equation" from above thus has this solution:  $\begin{bmatrix} \square & \square & \square \end{bmatrix} = \begin{bmatrix} \square & \square & \square \end{bmatrix} + \begin{bmatrix} \square & \square & \square \end{bmatrix} + \begin{bmatrix} \square & \square & \square \end{bmatrix}$ . Note that adding something that's lighter than medium gray makes the overall result lighter while adding something darker makes the result darker (with medium gray itself being the [identity element](#)).

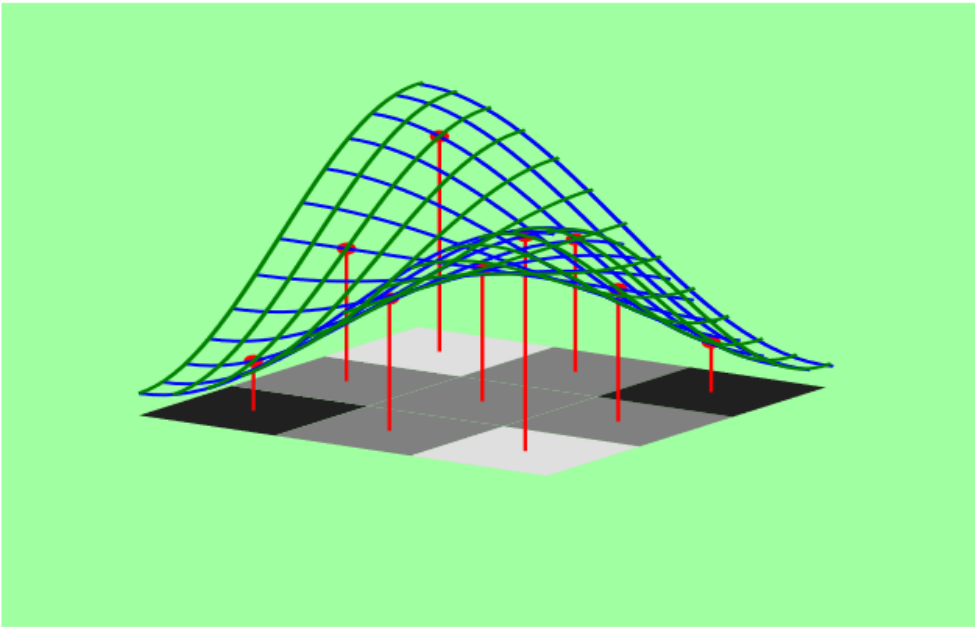
<http://weitz.de/dct/>

We can now go on and add the two functions  $x \mapsto \cos(3x)$  and  $x \mapsto \cos(4x)$  and sample all five functions at five (instead of three) equidistant points. That'll give us more and larger "Lego bricks" which we can use to construct all pictures consisting of *five* pixels:

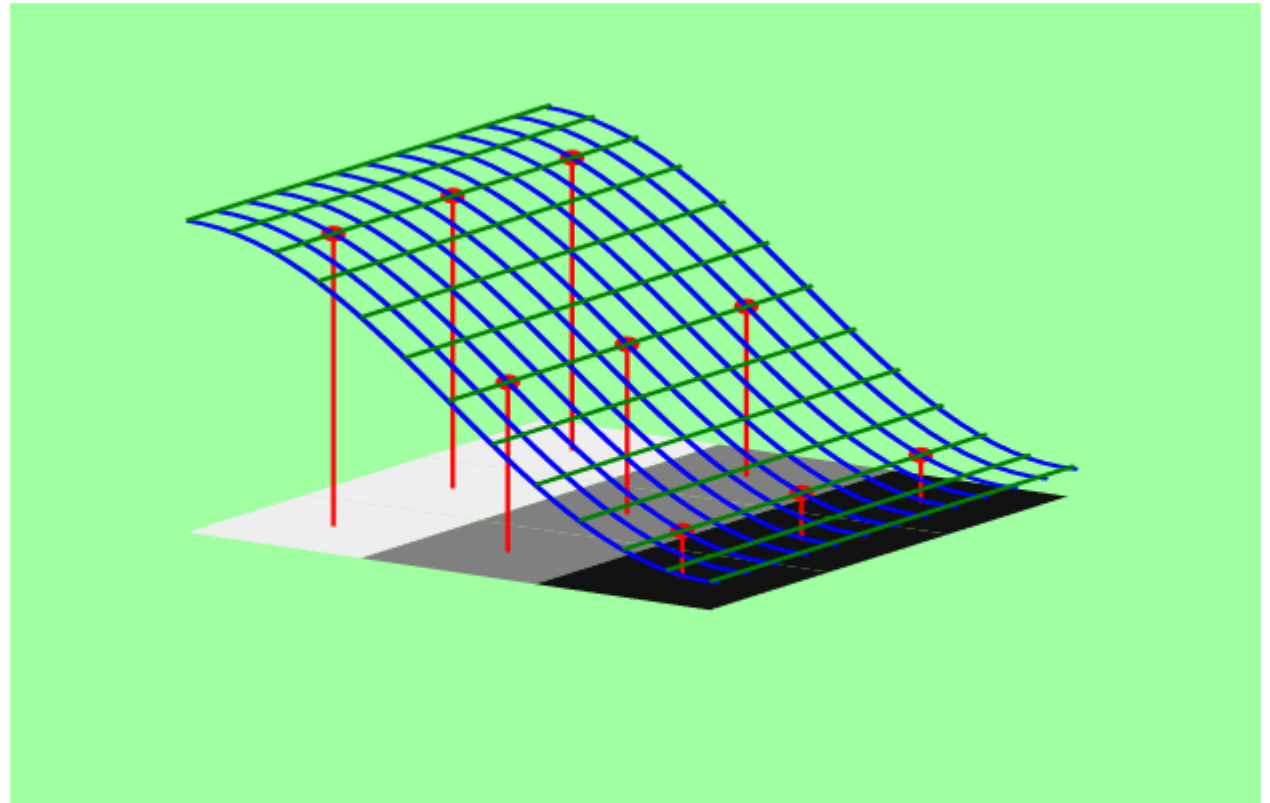
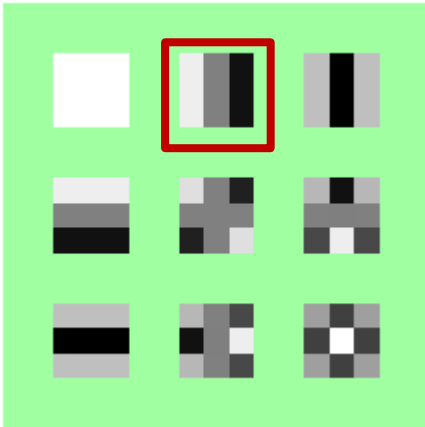


And we can of course do this for ten, or fifty, or many, many more pixels. But there's something even more interesting we can do. We can arrange the pixels in two dimensions! For example, we can arrange nine pixels together with their "sample points" in a three-by-three square. We then let one cosine function run in one direction and another one perpendicular to it. The *product* of the two functions will yield the values determining the gray tones.

You can see what we get below. Click on of the numbers at the bottom of the picture to see different patterns. And within the picture, move your mouse with the button pressed to see the cosine curves from different angles.

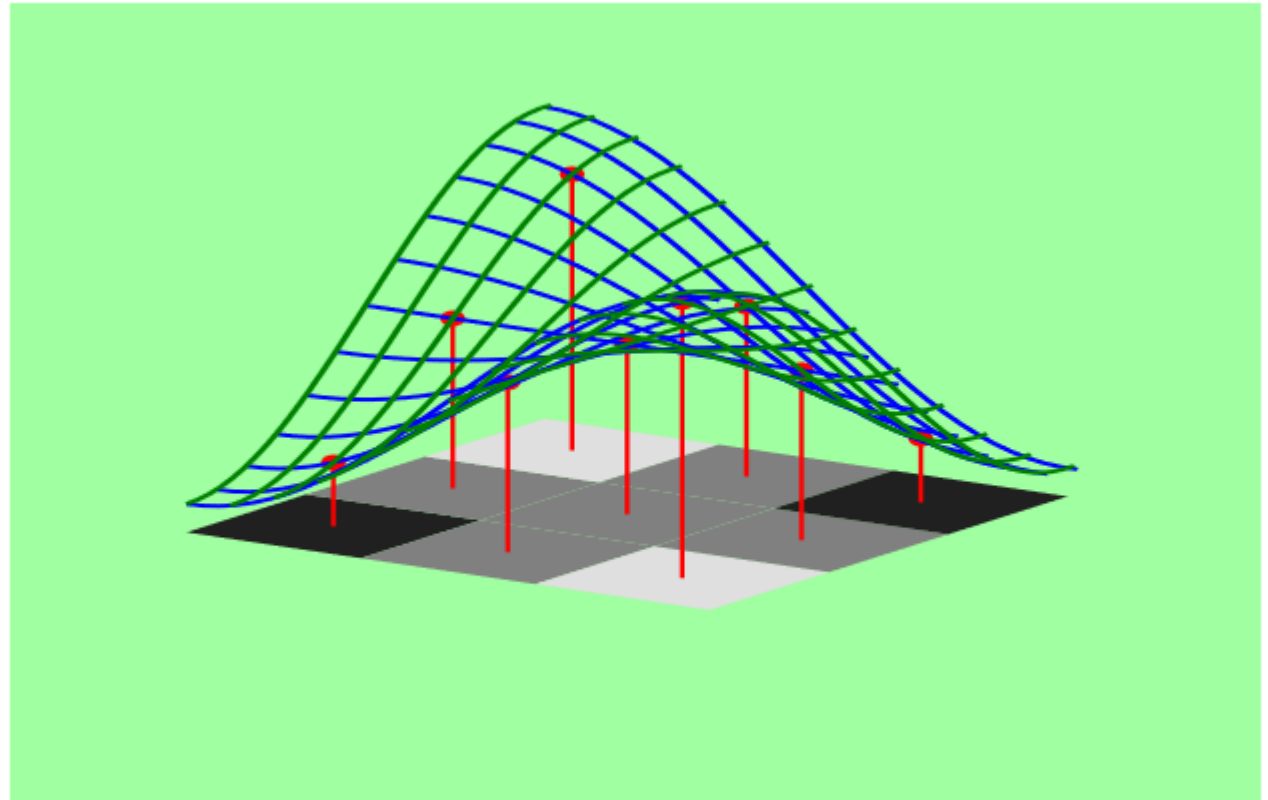
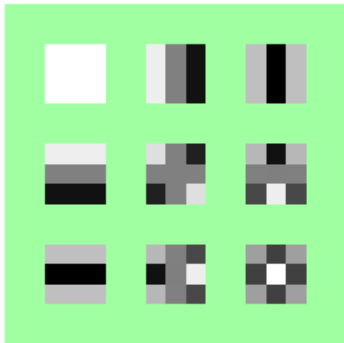


<http://weitz.de/dct/>



1 2 3 4 5 6 7 8 9


























1 2 3 4 5 6 7 8 9

Note that we have arranged the patterns in a certain way. If you go through this table from left to right, you're observing increasing horizontal frequencies in the patterns. If you go from top to bottom, you're seeing increasing vertical frequencies. In other words, if you traverse the table diagonally (from the upper left to the lower right corner), the way the patterns are numbered, you move from the pattern with no variation at all towards more "turbulent" patterns.

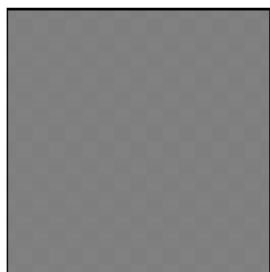
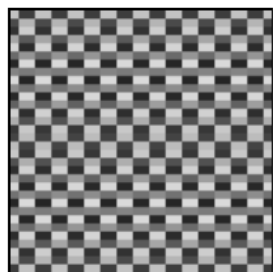
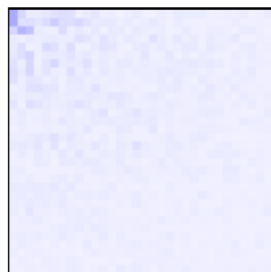
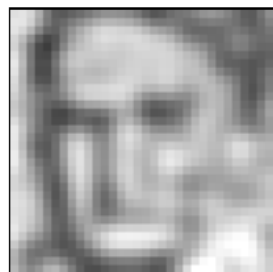
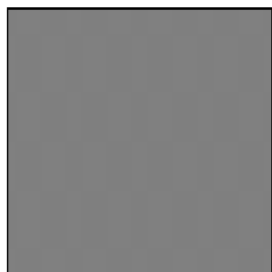
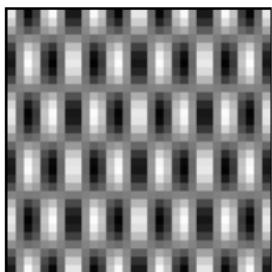
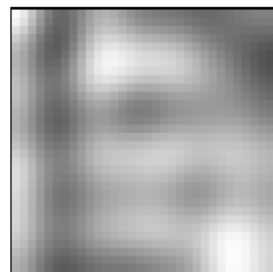
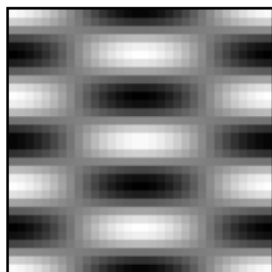
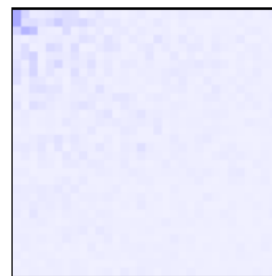
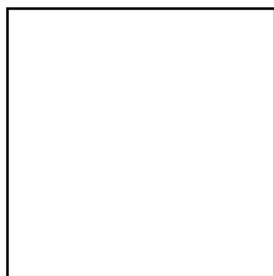
It's again easy to see mathematically that *all* three-by-three pictures can be represented (uniquely, by the way) as combinations of these nine patterns. And it's instructive to look at a few examples.

		1	2	3	4	5	6	7	8	9	
											
A		0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	
B		0.667	0.000	0.000	-0.333	0.000	0.000	0.000	0.000	0.000	
C		0.400	0.000	0.693	0.000	0.000	-0.200	0.000	0.000	0.000	
D		0.300	0.173	0.693	-0.100	0.000	-0.200	0.000	0.000	0.000	
E		0.200	-0.065	0.064	0.435	-0.527	0.041	-0.547	-0.267	-0.014	
F		0.200	-0.065	0.064	0.435	-0.527	0.041	-0.547	-0.267	0.000	

Note that the **blue squares** on the right visualize how important the individual patterns are to reconstruct a picture. One can see that the less "turbulent" a picture is, the more "energy" is concentrated in the upper left of the square.

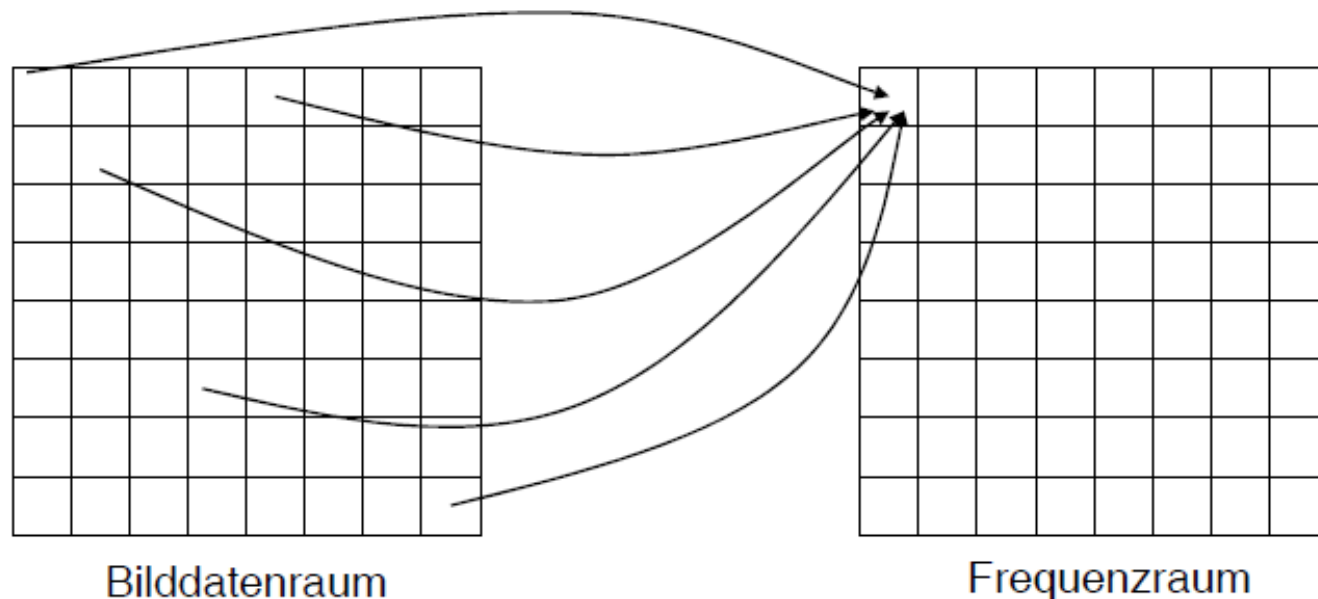


		1	2	3	4	5	6	7	8	9	
A		0.000	0.000	1.000	0.000	0.000	0.000	0.000	0.000	0.000	
B		0.667	0.000	0.000	-0.333	0.000	0.000	0.000	0.000	0.000	
C		0.400	0.000	0.693	0.000	0.000	-0.200	0.000	0.000	0.000	
D		0.300	0.173	0.693	-0.100	0.000	-0.200	0.000	0.000	0.000	
E		0.200	-0.065	0.064	0.435	-0.527	0.041	-0.547	-0.267	-0.014	
F		0.200	-0.065	0.064	0.435	-0.527	0.041	-0.547	-0.267	0.000	





# DCT: Zusammenhang Datenraum - Frequenzraum



- Ein **Punkt** im **Frequenzraum fasst** die **Informationen** aus dem aktuell betrachteten **Bildraum (8x8 Pixel) zusammen**
- Kanten erscheinen als Anteile hoher Frequenzen; bei Flächen sind die hohen Frequenzen fast Null
  - Gute Voraussetzung für spätere Kompression der „Null-nahen“ Werte durch Entropiekodierung



## Diskrete Cosinus-Transformation (DCT)

$$C(u, v) = \alpha(u) \cdot \alpha(v) \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) \cdot \cos\left(\frac{(2m+1) \cdot u \cdot \pi}{2M}\right) \cdot \cos\left(\frac{(2n+1) \cdot v \cdot \pi}{2N}\right)$$

## Inverse DCT (IDCT)

$$f(m, n) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} \alpha(u) \cdot \alpha(v) \cdot C(u, v) \cdot \cos\left(\frac{(2u+1) \cdot m \cdot \pi}{2M}\right) \cdot \cos\left(\frac{(2v+1) \cdot n \cdot \pi}{2N}\right)$$

$$\alpha(u) = \begin{cases} \sqrt{1/M} & , \text{für } u = 0 \\ \sqrt{2/M} & , \text{für } u \neq 0 \end{cases} \quad \text{und} \quad \alpha(v) = \begin{cases} \sqrt{1/N} & , \text{für } v = 0 \\ \sqrt{2/N} & , \text{für } v \neq 0 \end{cases}$$

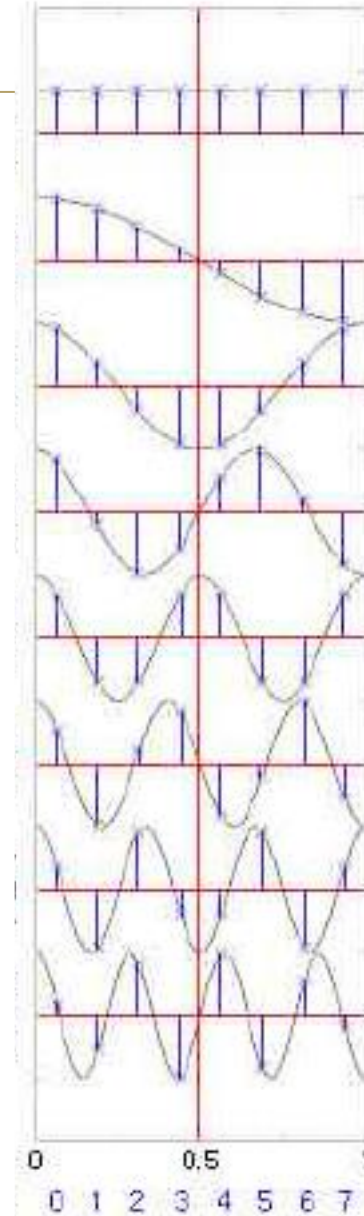
**Basisfunktionen** der DCT/IDCT sind Schwingungen, deren Wert für jede zweite Frequenz  $u$  am Anfang und am Ende des Intervalls **unterschiedliches Vorzeichen** haben.





## Invertierbarkeit der Transformation

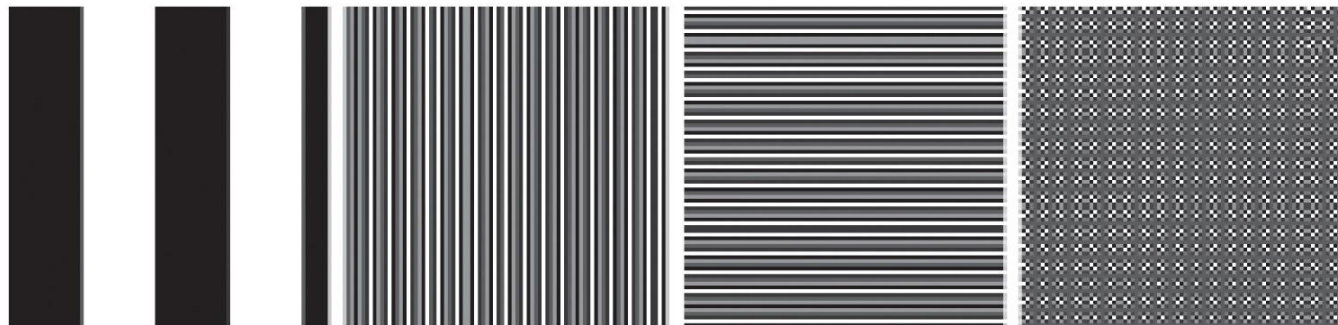
- Die Transformation ist **invertierbar**, wenn die Basisfunktionen eine **orthogonale Basis** bilden.
  - Was ist Orthogonalität für Funktionen?
  - Wann bilden Basisfunktionen eine orthogonale Basis?
- Transformation: „Projektion“ der Funktion auf die neue Basis.
- Inverse Transformation: Projektion auf die alte Basis.
- Orthogonalität, Projektion, Transformation etc. sind Begriffe aus der **Vektoralgebra** und können auch für Funktionen genutzt werden.
- Beispiel: Basisfunktionen der DCT → siehe rechts





## Eigenschaften der DCT

- DCT ist invertierbar
- Amplitude sinkt schnell mit steigender Frequenz
- Transformation wird aus eindeutig definierten endlichen Menge von Cosinus-Funktionen zusammengesetzt
- Koeffizientenmatrizen können statisch vorberechnet werden
- Nullpunkt wird in die Ecke verlegt



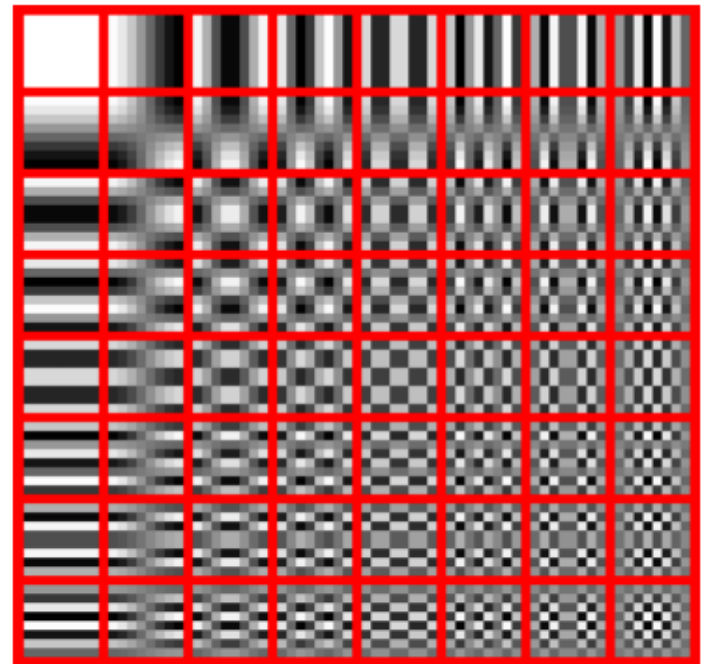
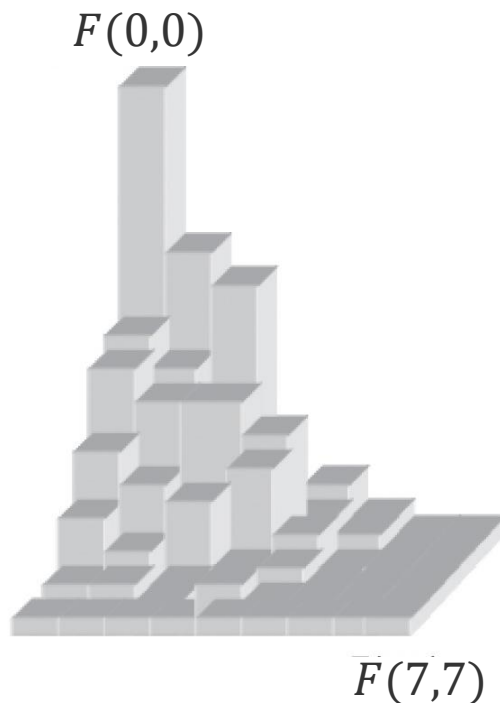


# Matrixdarstellung zur Durchführung einer DCT

$$\begin{bmatrix}
 \frac{1}{4}\sqrt{2} & \frac{1}{4}\sqrt{2} & \frac{1}{4}\sqrt{2} & \frac{1}{4}\sqrt{2} & \frac{1}{4}\sqrt{2} & \frac{1}{4}\sqrt{2} & \frac{1}{4}\sqrt{2} & \frac{1}{4}\sqrt{2} \\
 \frac{1}{2}\cos\left(\frac{1}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{3}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{5}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{7}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{9}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{11}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{13}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{15}{16}\pi\right) \\
 \frac{1}{2}\cos\left(\frac{1}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{3}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{5}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{7}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{9}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{11}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{13}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{15}{8}\pi\right) \\
 \frac{1}{2}\cos\left(\frac{3}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{9}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{15}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{21}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{27}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{33}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{39}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{45}{16}\pi\right) \\
 \frac{1}{2}\cos\left(\frac{1}{4}\pi\right) & \frac{1}{2}\cos\left(\frac{3}{4}\pi\right) & \frac{1}{2}\cos\left(\frac{5}{4}\pi\right) & \frac{1}{2}\cos\left(\frac{7}{4}\pi\right) & \frac{1}{2}\cos\left(\frac{9}{4}\pi\right) & \frac{1}{2}\cos\left(\frac{11}{4}\pi\right) & \frac{1}{2}\cos\left(\frac{13}{4}\pi\right) & \frac{1}{2}\cos\left(\frac{15}{4}\pi\right) \\
 \frac{1}{2}\cos\left(\frac{5}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{15}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{25}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{35}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{45}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{55}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{65}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{75}{16}\pi\right) \\
 \frac{1}{2}\cos\left(\frac{3}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{9}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{15}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{21}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{27}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{33}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{39}{8}\pi\right) & \frac{1}{2}\cos\left(\frac{45}{8}\pi\right) \\
 \frac{1}{2}\cos\left(\frac{7}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{21}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{35}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{49}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{63}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{77}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{91}{16}\pi\right) & \frac{1}{2}\cos\left(\frac{105}{16}\pi\right)
 \end{bmatrix}$$

# Diskrete Cosinustransformation (DCT)

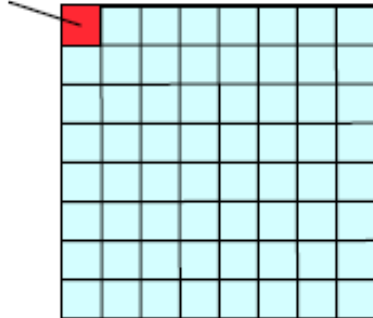
- Funktionsweise analog zu Fouriertransformation
- Vorteil: Eigenschaft der „Energieverdichtung“ in Richtung der linken oberen Ecke





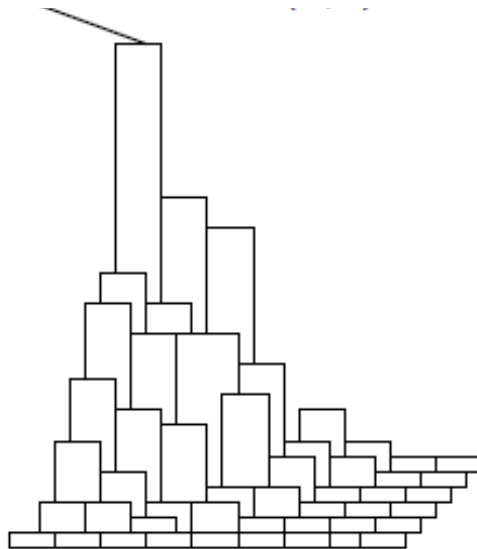
# Interpretation der DCT-Koeffizienten

DC-Koeffizient



Alle anderen:  
AC-  
Koeffizienten

DC-Koeffizient  $F(0,0)$



- **DC-Koeffizient** gibt den **Grundton** des beschriebenen Bereichs (8x8) im Bild an (in der aktuellen Komponente)
- Die **AC-Koeffizienten** geben mit **aufsteigenden Indizes den Anteil „höherer Frequenzen“** an, d.h. die Zahl der (vertikalen bzw. horizontalen) Streifen
- Z.B.:
  - $F(7,0)$  gibt an, zu welchem Anteil extrem dichte waagrechte Streifen vorkommen;
  - $F(0,7)$  gibt an, zu welchem Anteil extrem dichte senkrechte Streifen vorkommen
- DC: Gleichstrom, AC: Wechselstrom



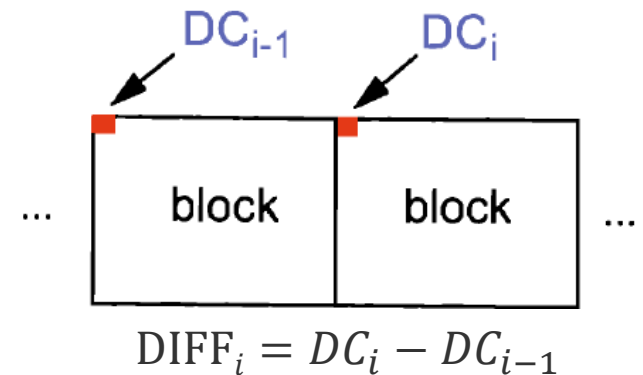


# Entropiekodierung (DC-Koeffizienten)

- DC-Koeffizienten bestimmende Einheit
- **Berechnung der Differenzen aufeinanderfolgender DC-Koeffizienten**

$$DIFF_i = DC_i - DC_{i-1}$$

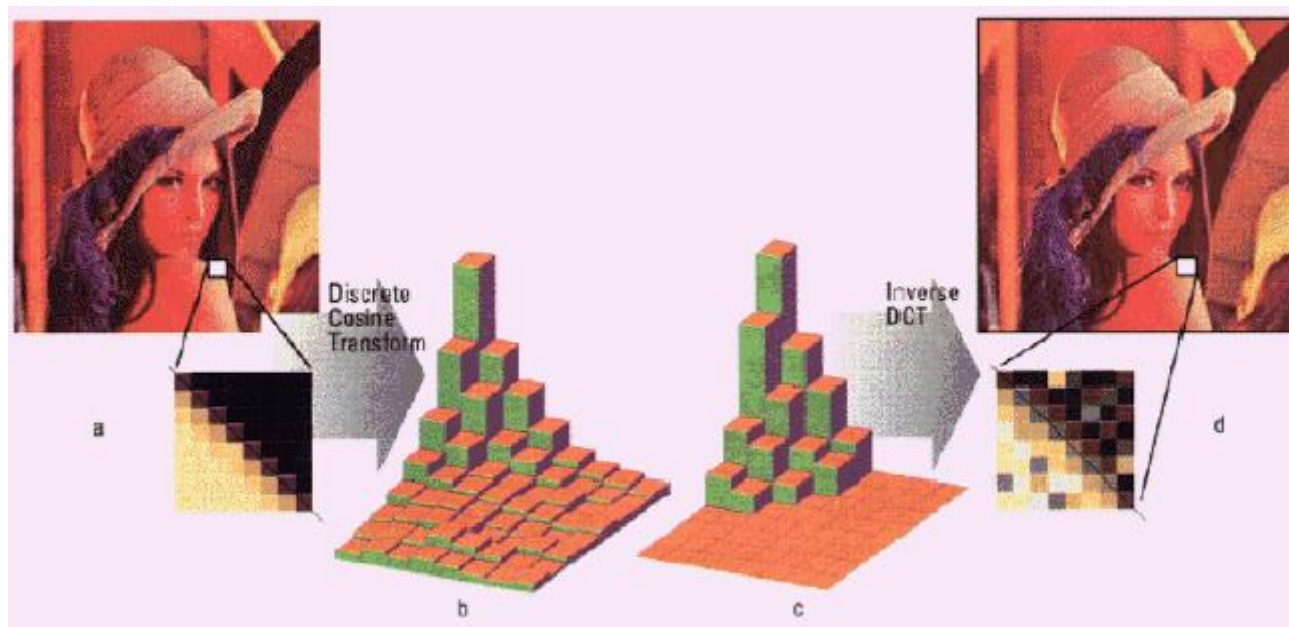
- **Verwendung der Differenzen**
- Bis jetzt: verlustfreie Kompression!



Kategorie	<i>DIFF</i> -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

# Quantisierung

- **Quantisierung der DCT-Koeffizienten**
    - Abbildung eines Intervalls reeller Zahlen auf Integer-Zahl
    - Unterschiedliche Granularität pro Koeffizient möglich
- **Destruktiver Kompressionsschritt**





- Entscheidender Schritt zum **Informationsverlust** und damit zur starken Kompression!
  - Runden der Koeffizienten erzeugt viele Nullen und ähnliche Werte
  - Damit besser mit verlustfreien Verfahren komprimierbar
- **Quantisierungstabelle:**
  - Enthält 64 vorgegebene und konstante Bewertungs-Koeffizienten  $Q(u, v)$
  - Bedeutung: Bewertung der einzelnen Frequenzanteile des Bildes
  - Größere Tabelleneinträge bedeuten stärkere Vergröberung
  - Konkrete Tabellen nicht Bestandteil des Standards (nur zwei Beispiele)
    - Typisch: Verschiedene Bewertung für hohe und niedrige Frequenzen
  - Benutzte Quantisierungstabellen werden als Bestandteil der komprimierten Daten abgelegt und bei Dekompression benutzt
- Berechnung:
 
$$F'(u, v) = \left\lfloor \frac{F(u, v)}{Q(u, v)} \right\rfloor$$



# Rechenbeispiel: Quantisierung

7842	199	448	362	342	112	31	22
198	151	181	264	59	37	14	3
142	291	218	87	27	88	27	12
111	133	159	119	58	65	36	2
49	85	217	50	8	3	14	12
58	120	60	40	41	11	2	1
30	121	61	22	30	1	0	1
22	28	2	33	24	51	44	81

DCT-Koeffizientenmatrix (Eingabe)

Dividiere  
durch  
Quant.-  
Matrix

Dividiere  
durch  
Quant  
Scale

980	12	23	16	13	4	1	0
12	9	8	11	2	1	0	0
7	13	8	3	0	2	0	1
5	6	6	4	2	1	0	0
2	3	8	1	0	0	0	0
2	4	2	1	1	0	0	0
1	4	2	1	0	0	0	0
0	0	1	0	0	0	0	0

DCT-Koeffizientenmatrix (Ergebnis)

8	16	19	22	26	27	29	34
16	16	22	24	27	29	34	37
19	22	26	27	29	34	34	38
22	22	26	27	29	34	37	40
22	26	27	29	32	35	40	48
26	27	29	32	35	40	48	58
26	27	29	34	38	48	56	69
27	29	35	38	46	56	59	83

Quantisierungsmatrix

Code	Linear Quant Scale	Non- linear Quant Scale
1	2	1
8	16	8
16	32	24
20	40	40
24	48	56
28	56	88
31	62	112

Quelle: broadcastpapers.com

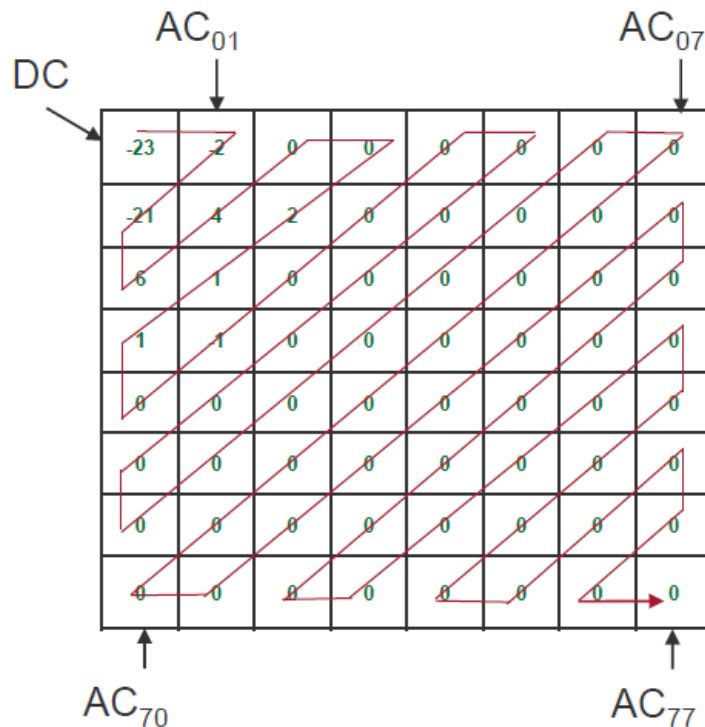


## Quantisierungseffekt





# Entropiekodierung (AC-Koeffizienten)



**DC-Koeffizient** gibt den **Grundton** des beschriebenen Bereichs (8x8) im Bild an (in der aktuellen Komponente)

Die **AC-Koeffizienten** geben mit **aufsteigenden Indizes den Anteil „höherer Frequenzen“** an, d.h. die Zahl der (vertikalen bzw. horizontalen) Streifen

Anordnung der AC-Koeffizienten in **RLE-günstiger Form**

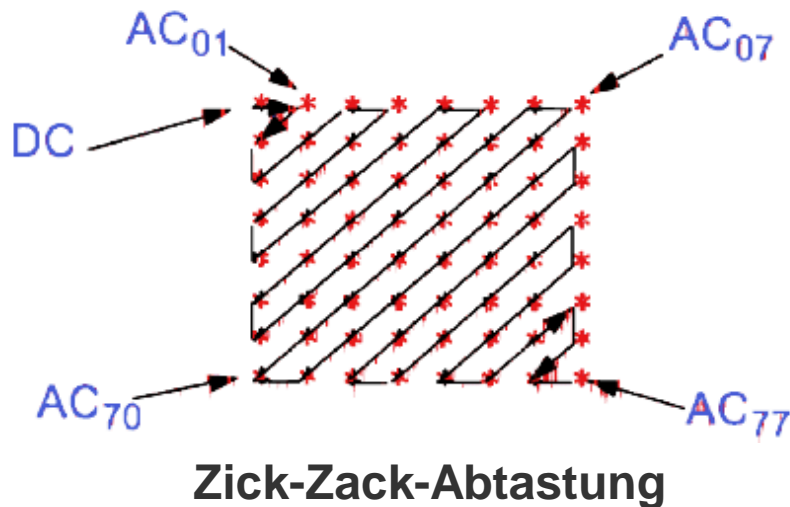
-23	-2	-21	6	4	0	0	2	1	1	2	0	0	-1	0	0	0	...
-----	----	-----	---	---	---	---	---	---	---	---	---	---	----	---	---	---	-----





## → Durchlaufen in Zick-Zack-Form

- Alle AC-Koeffizienten ungleich Null werden in **Kategorien** unterteilt
- Kombination aus **Lauflänge** und **Kategorie** wird **Huffman-kodiert**
  - Zahl der Bits abhängig von der Häufigkeit des Wertes



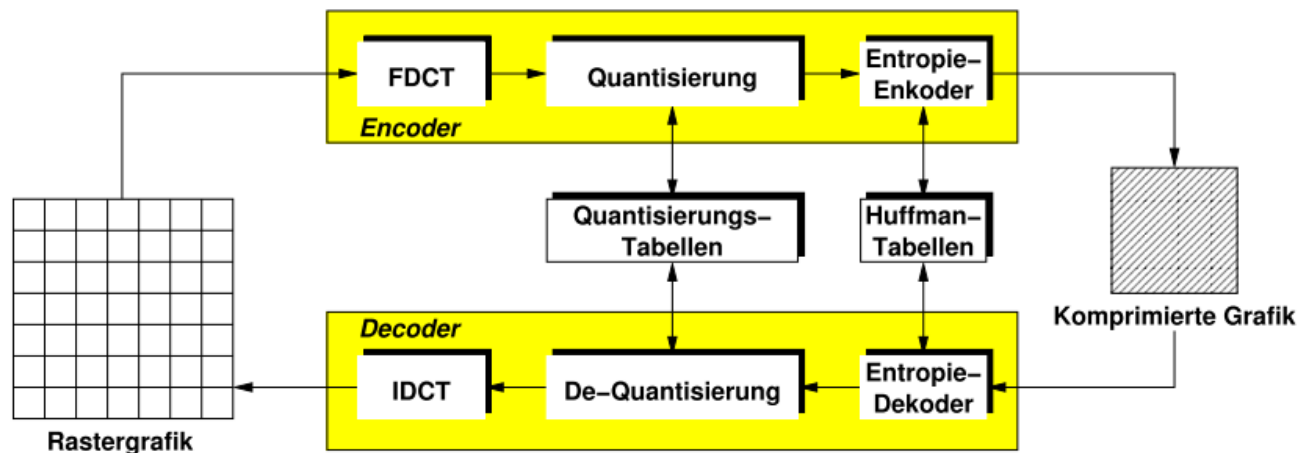
Kategorie	AC-Wert
1	-1, 1
2	-3, -2, 2, 3
3	-7, ..., -4, 4, ..., 7
4	-15, ..., -8, 8, ..., 15
5	-31, ..., -16, 16, ..., 31
6	-63, ..., -32, 32, ..., 63
7	-127, ..., -64, 64, ..., 127
8	-255, ..., -128, 128, ..., 255
9	-511, ..., -256, 256, ..., 511
10	-1023, ..., -512, 512, ..., 1023

**Kategorienbildung**



## Anwendungsbeispiel: JPEG

- **Transformation von Pixel-Blöcken** (z.B. 8x8)
- DCT transformiert **Pixel-Block in Koeffizienten** für korrespondierende Frequenz-Repräsentation
- **Tabellenbasierte Quantisierung** für 64 Frequenzkomponenten, Gewichtung nach menschlichem Sehvermögen
- **Entropie-Kodierung**



# Ergebnis (verschiedene Quantisierungsstufen)



229 KByte, 3:1, 7 bpp



24 KByte, 32:1, 0.74 bpp



18 KByte, 43:1



12 KByte, 66:1, 0.36 bpp

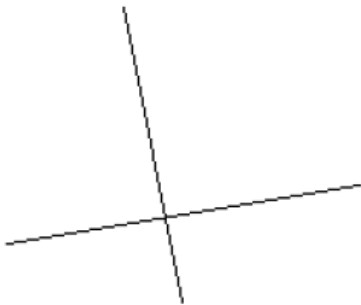


10 KByte, 78:1

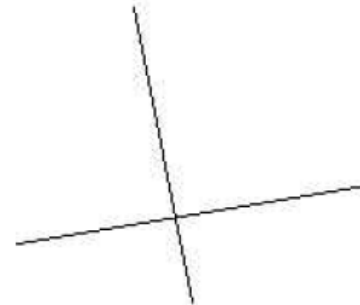
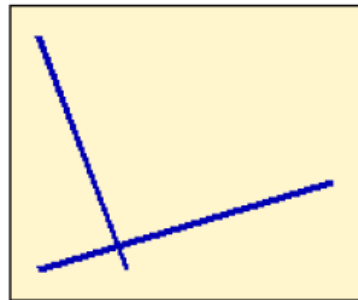


5 KByte, 140:1, 0.17 bpp

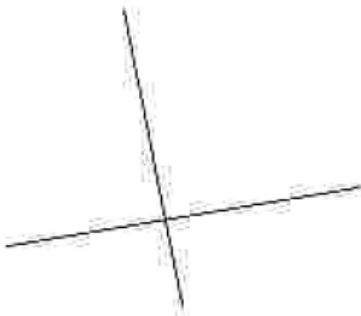
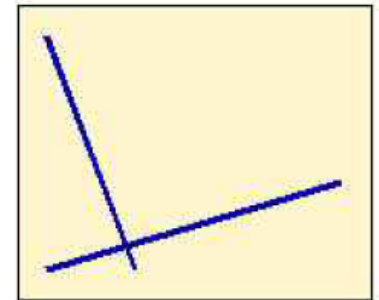
# Ergebnis (harte Kanten)



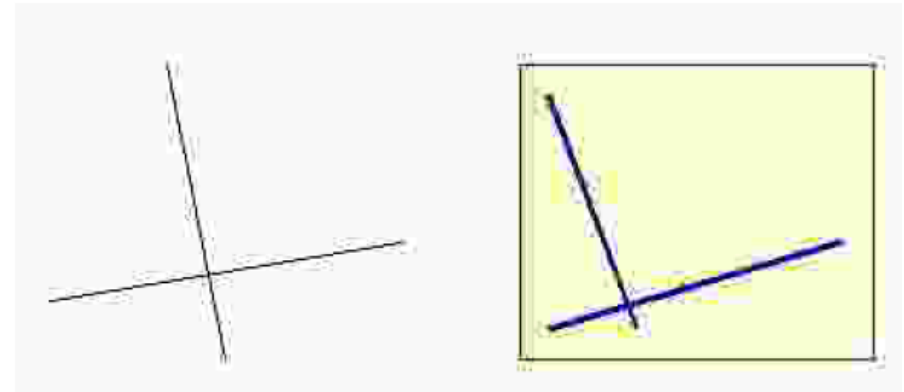
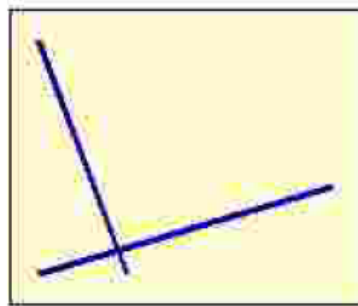
Original (GIF)



JPEG xv quality 40%



JPEG xv quality 10%



JPEG xv quality 5%

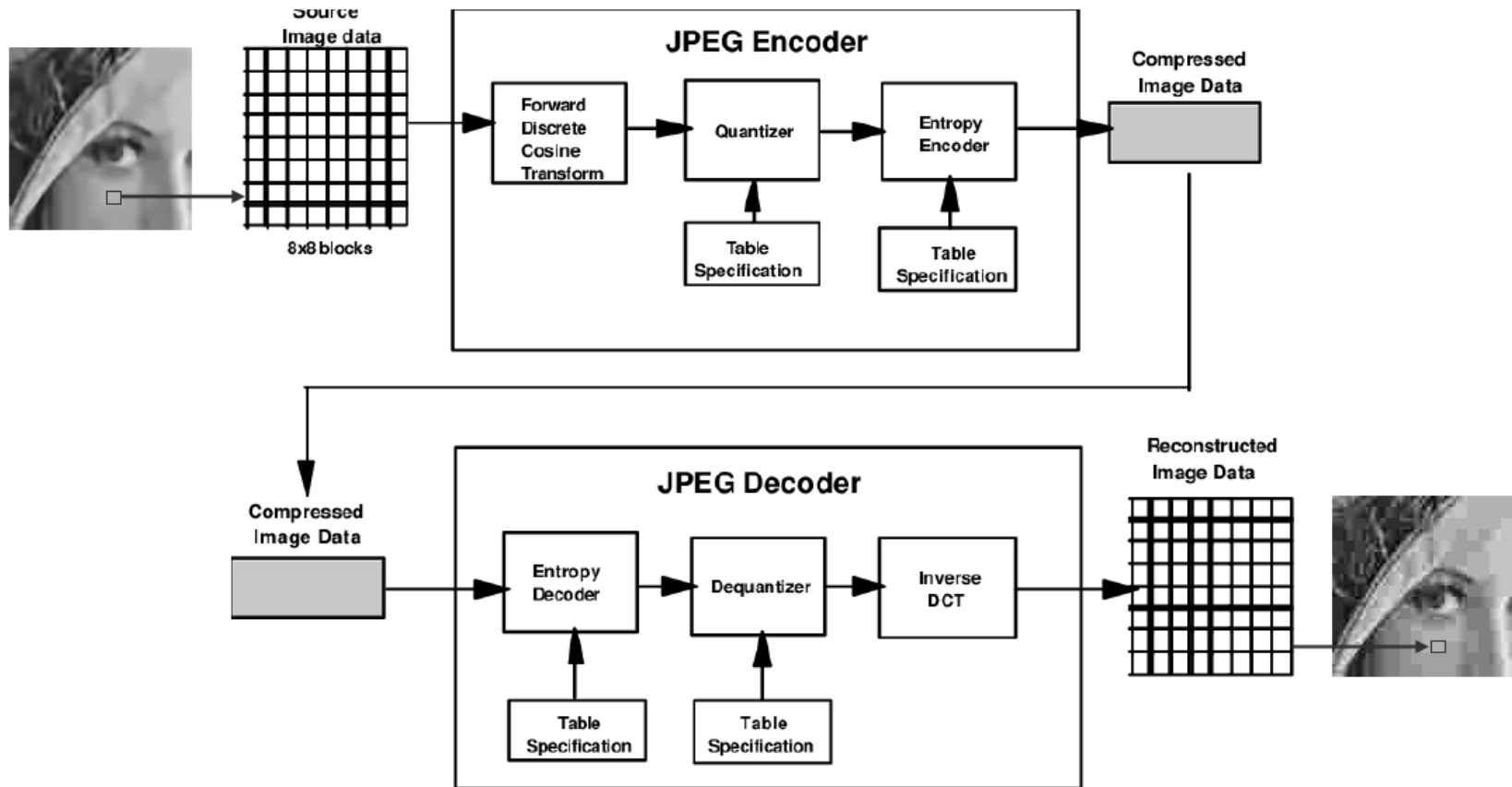


# Zusammenfassung

- Bildkomprimierung (JPEG)
  - Helligkeitsinformationen (Y) besitzen hohen Informationsgehalt
  - Transformation in Frequenzraum → DCT
  - Geringere Quantisierung höherer Frequenzanteile
  - Lauflängen- und Huffman-Kodierung



# Zusammenfassung







<https://www.youtube.com/watch?v=URYoWZnXISg>



## HEIF

- High Efficient Image File Format (HEIF)
- Verlustbehaftetes Format
- Vorgestellt 2000 von Moving Pictures Expert Group
- Containerformat für Bilder und Bildsequenzen
- Enthält Alphakanal und Tiefeninformationen
- Auch Text und Audio darstellbar



## HEIF Algorithmus

- Komprimierung erfolgt nach High Efficient Video Codec (HEVC)
- Nachfolger des MPEG-4 Standards
- Kompression durch Bewegungskompensation
- Verbesserte Beschreibung von Bewegungsvektoren  
→ 35 Richtungen statt 9 Richtungen bei H.264

## HEIF Algorithmus

- Coding Tree Units statt variabel bis 64x64 px (statt 16x16 px Makroblöcke bei H.264)

HEVC



H.264



<https://teradek.com/blogs/articles/3-reasons-why-hevc-x265-matters-and-how-you-can-start-using-it-now>



## HEIF

HEIF: 38 KiB



JPG: 49 KiB

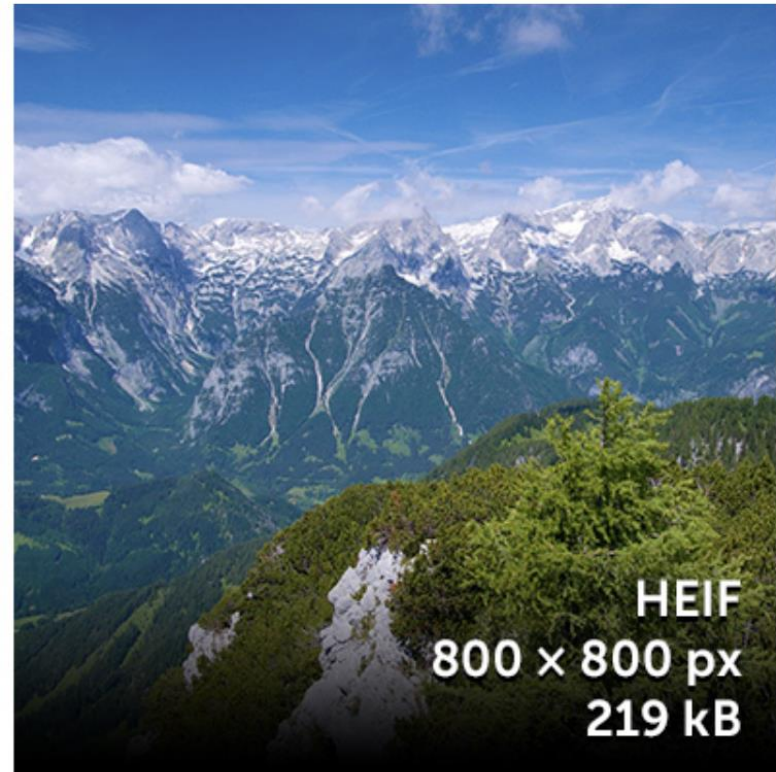
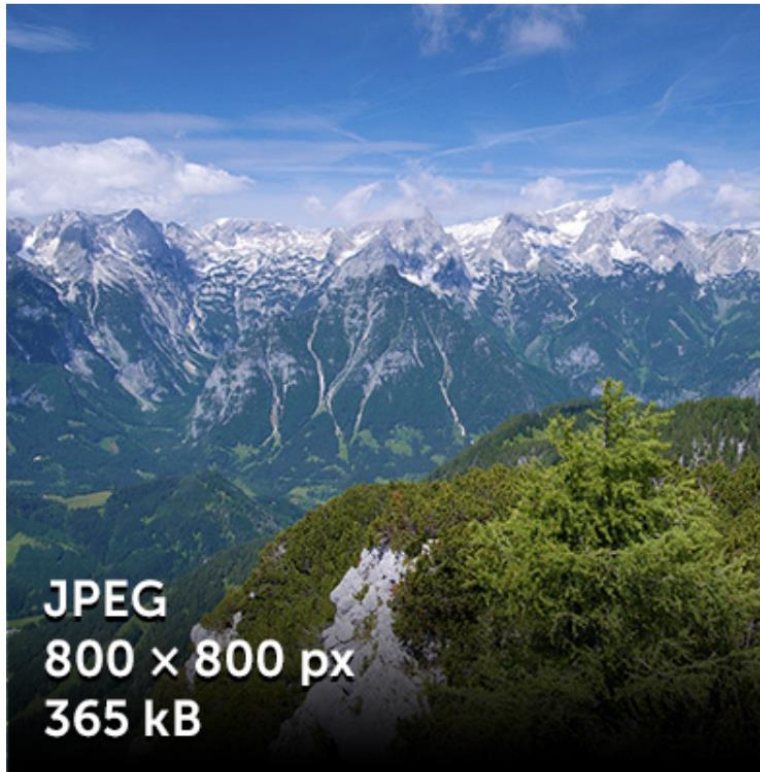


---

<http://nokiatech.github.io/heif/comparison.html>



# JPEG vs. HEIF



Zoner Photo Studio X Brings the Future of Image Storage



## HEIF



---

<http://nokiatech.github.io/heif/comparison.html>





## HEIF

- Kompressionsperformanz (Bitratenerhöhung für gleiche Qualität)

Auflösung	Bildart	JPEG	JPEG200
2560x1600	Cropped 4K	87%	48%
1920x1080	HD Format	124%	15%
832x480	Mobile Video	122%	50%
1280x720	720p VideoConf	170%	23%
1024x768	Computer Screen Content	223%	87%
<b>AVERAGE</b>		139%	44%

→ JPEG Datei bei gleicher Qualität 2,39-fache Dateigröße

<https://nokiotech.github.io/heif/technical.html>



## WebP

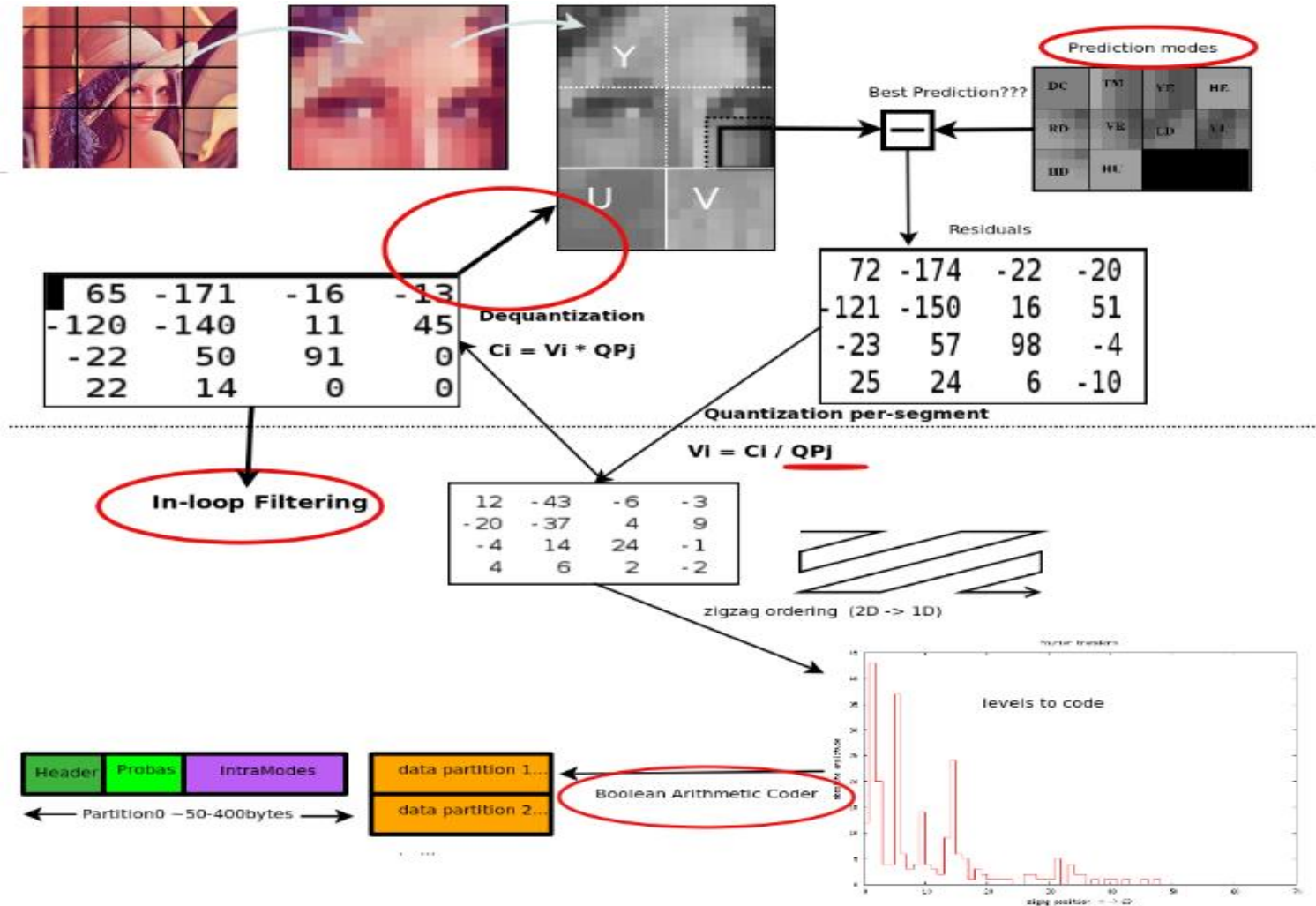
- Grafikformat von Google (2010)
- Komprimiert statische oder animierte Bilder
- Erlaubt verlustbehaftete als auch verlustfreie Kompression
- Ermöglicht Transparenz (Alphakanal)
- Speicherung von Metadaten
- → 30% kleiner als JPEG bei vergleichbarem Bild



## WebP Kompressionsverfahren

- Basierend auf Intra-Frame-Codierung von VP8
- Blockbasiertes Transformationsverfahren (4x4px)
- Farbunterabtastung 4:2:0
- Entropiekodierung durch Art binäre arithm. Codierung

## WebP Kompressionsverfahren (verlustbehaftet)



[https://developers.google.com/speed/webp/images/compression-webp\\_lossy.png](https://developers.google.com/speed/webp/images/compression-webp_lossy.png)

## WebP

JPEG 120.78KB



WebP 80.76%



<https://developers.google.com/speed/webp/gallery1>