



MULTIMEDIA BILDER CODIEREN / KOMPRIMIEREN

M114 / ARJ / 3-2025

Photographie



Grammophon



Cinematographie



Television



LCD-Display



+

1800

+

+

+

+

1850

+

+

+

+

+

1900

+

+

+

+

+

1950

+

+

+

+

2000

SW

Color

Photographie



Grammophon



Cinematographie



Television



LCD-Display



+

1800

+

+

+

+

1850



+

1900

+

+

+

+

SW



+

1950

+

+

+

+

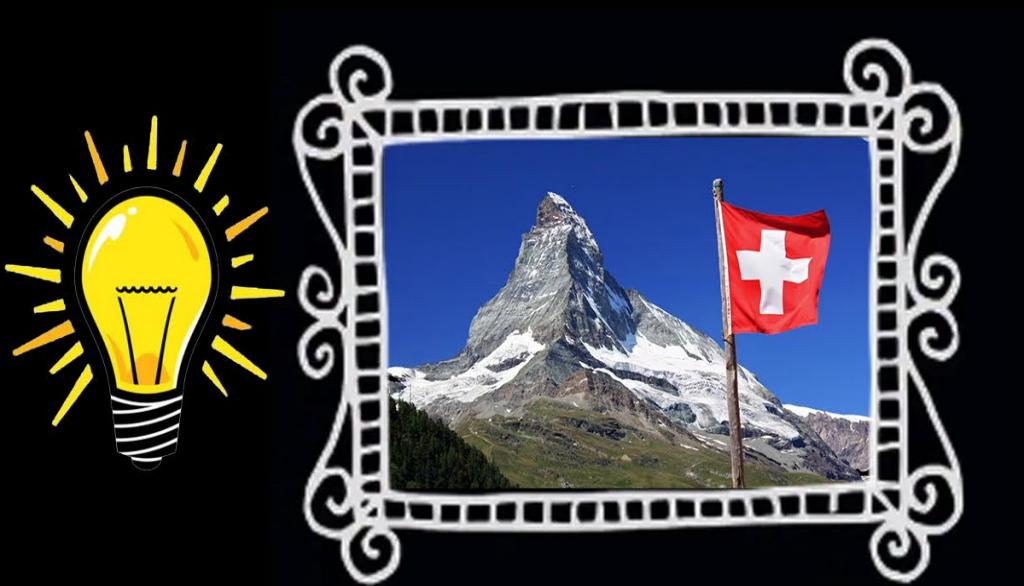
Color



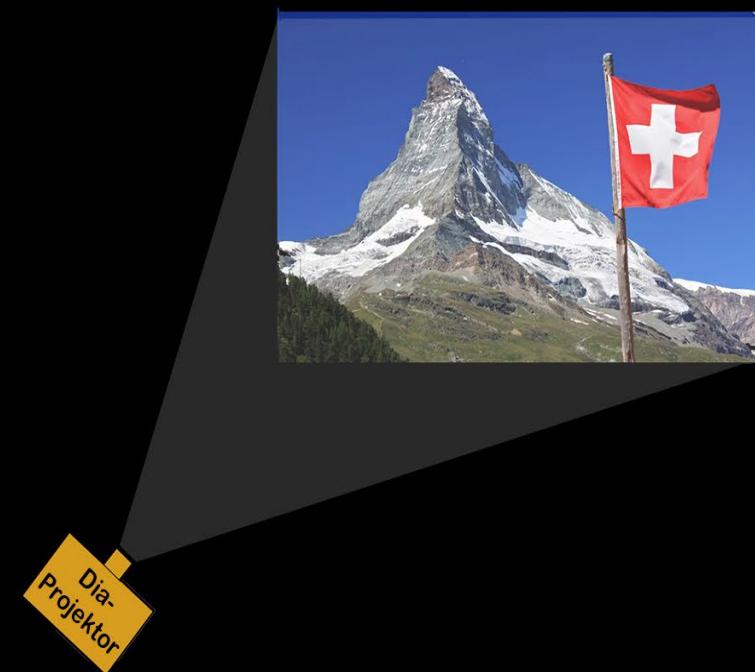
+

2000

Weissses Licht an

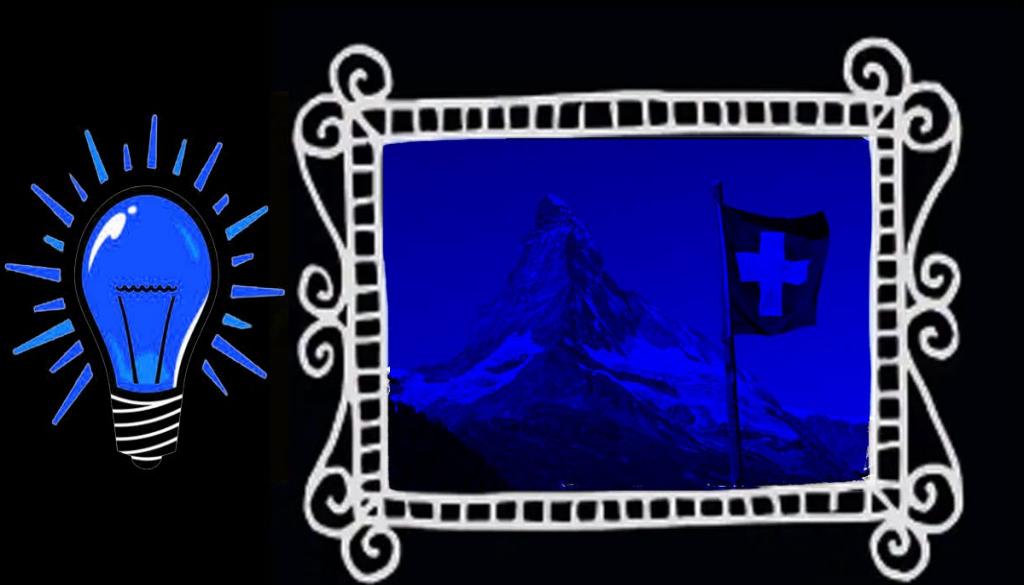


Subtraktives Farbsystem / CMYK

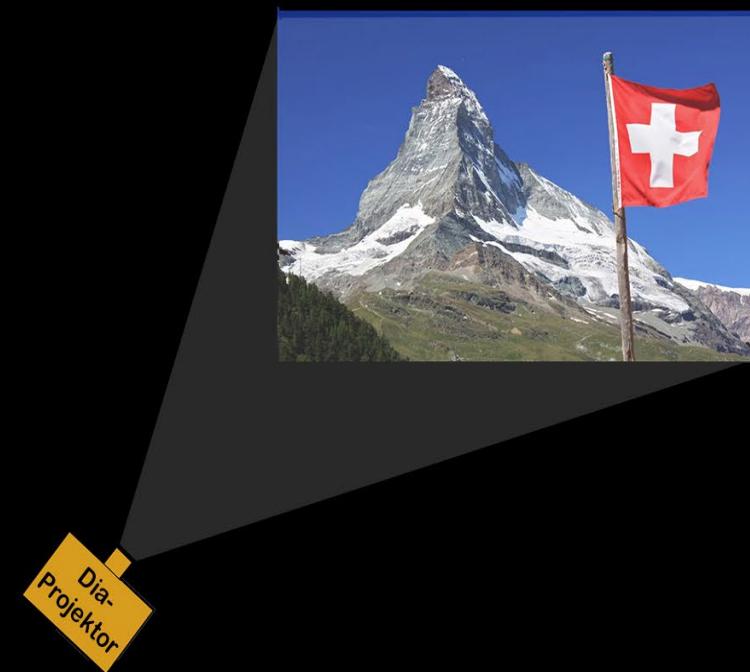


Additives Farbsystem / RGB

Blaues Licht an

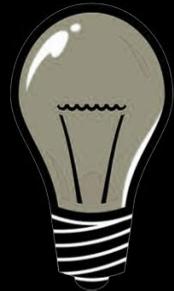


Subtraktives Farbsystem / CMYK



Additives Farbsystem / RGB

Licht aus



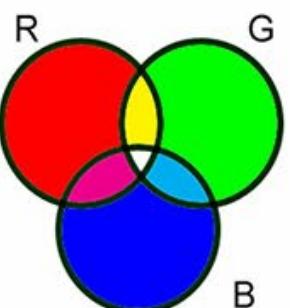
Dia-
Projektor



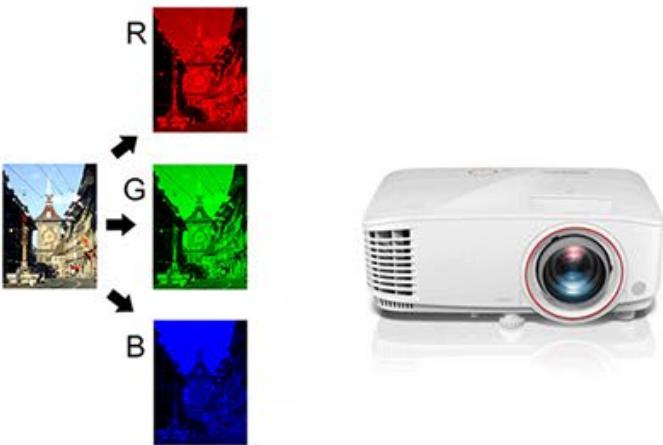
Subtraktives Farbsystem / CMYK

Additives Farbsystem / RGB

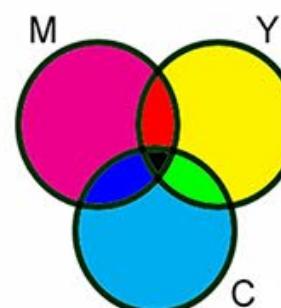
Additives Farbsystem



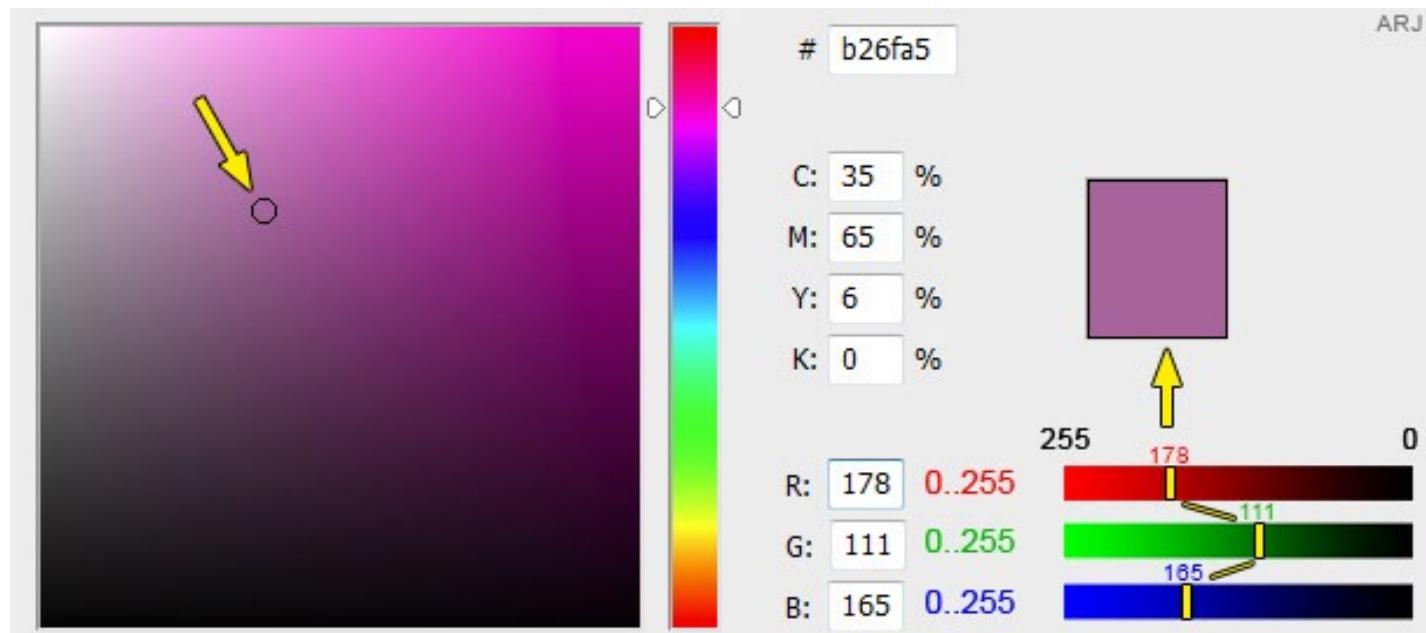
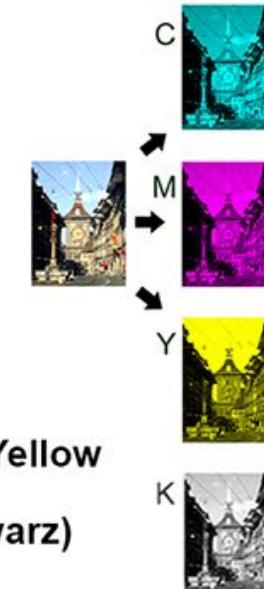
RGB = Rot/Grün/Blau



Subtraktives Farbsystem



CMYK = Cyan/Magenta/Yellow
(K=Keyfarbe meist Schwarz)



Probieren sie es gleich mal selber aus: *(Dauer: 7 Min.)*

Welcher Farbe entsprechen die folgenden RGB-Farbcodes?

(Bei 1. und 7. kann ihnen https://www.w3schools.com/colors/colors_hexadecimal.asp oder https://www.w3schools.com/colors/colors_rgb.asp helfen.)

1. **RGB(252, 178, 91) =**

2. **#FF0000 =**

3. **#00FF00 =**

4. **#0000FF =**

5. **#FFFF00 =**

6. **#000000 =**

7. **#00BC00 =**

Welcher Farbe entsprechen die folgenden CMYK-Farbcodes?

(Bei 6. kann ihnen https://www.w3schools.com/colors/colors_cmyk.asp helfen.)

1. **C:100%, M:0%, Y:0%, K:0% =**

2. **C:0%, M:100%, Y:100%, K:0% =**

3. **C:100%, M:100%, Y:0%, K:0% =**

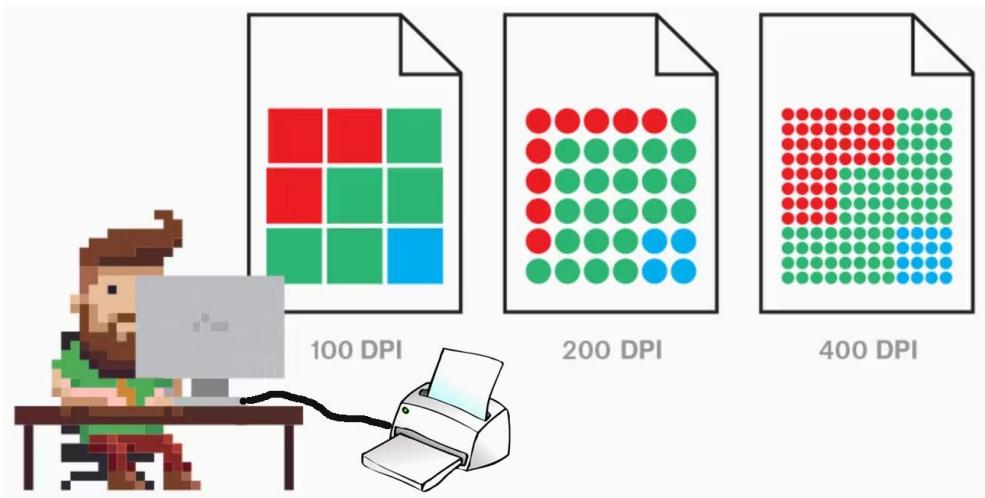
4. **C:100%, M:100%, Y:100%, K:0% =**

5. **C:0%, M:0%, Y:0%, K:100% =**

6. **C:0%, M:46%, Y:38%, K:22% =**



dpi beim Drucker

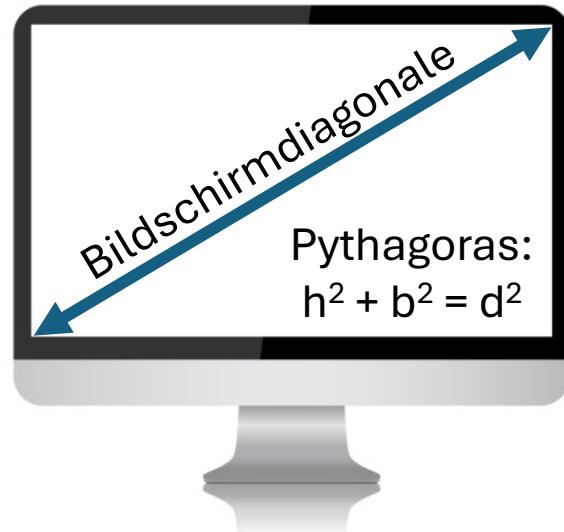


dpi = dots per inch

Anzahl Farbpunkte pro 2.54 cm Breite

1 Inch = 1 Zoll = 2.54 cm

ppi beim Bildschirm



Bildschirmkennwerte:

- **Bildschirmdiagonale**
in Zoll (Bsp.: 28 Zoll)
- **Auflösung**
in Pixel (Bsp.: 1080x1920)
- **Seitenverhältnis**
Breite zu Höhe (Bsp.: 16:9)
- **ppi**
Pixel pro Zoll in der Breite

ppi = pixel per inch

Anzahl Bildpunkte pro 2.54 cm Breite

Bsp.: Bildschirmdiagonale = **28"**, Seitenverhältnis **16:9**

$$28^2 = (16x)^2 + (9x)^2$$

$$784 = 256x^2 + 81x^2$$

$$784 = 337x^2$$

$$784/337 = x^2$$

$$1.5253 = x$$

Breite somit = 24.40 Zoll oder **h=61.99 cm**

Höhe somit = 13.73 Zoll oder **b=34.87 cm**

ppi: 1920 Pixel / 24.40 Zoll = **68.68ppi**

Ein 28 Zoll-Bildschirm mit 1080x1920 Pixel ist
ein **mittelmässiger** Bildschirm.

Und nun sie: *(Dauer: 5 Min.)*

1. dpi / Drucker:

Sie drucken ein **quadratisches Foto** mit einer Kantenlänge von 2000 Pixel mit 600dpi.
Berechnen sie die Höhe und die Breite in Zentimeter dieses Fotos.



2. ppi / Bildschirm:

Ihr neues **30-Zoll-Display** weist ein Seitenverhältnis von 16:10 bei 100ppi auf.
Berechnen sie die horizontale und vertikale Pixelauflösung.

1000 x 600 Pixel

Speicherbedarf RGB (1B/Kanal)
1000 x 600 x 3 = **1'800'000 Byte**



100 x 60 Pixel

Speicherbedarf RGB (1B/Kanal)

$$100 \times 60 \times 3 = 18'000 \text{ Byte}$$

Vorteil:

100x kleiner als Original 1000 x 600 !

Nachteil:

**Information ging unwiederbringlich
verloren (Verlustbehaftete Datenreduktion)**



Zurück zur Originalgrösse?

100 x 60 expandiert auf 1000x600 bedeutet:

Jedes Pixel wird zu einer 10x10 Pixel grosser farbmonotonen Fläche. Bild wirkt verschwommen.



Und nun gleich sie: *(Dauer: 8 Min.)*

1. HD1080:

Ein weit verbreitetes Videoformat ist HD1080. Damit meint man 1080 Bildzeilen bei einem Seitenverhältnis 16:9

- Berechnen sie Anzahl Pixel pro Bildzeile
- Berechnen sie den Speicherbedarf für ein einziges Bild, wenn jedes Pixel 3 Byte (RGB) benötigt.
- Berechnen sie den Speicherbedarf für ein Dreiminuten-Video, wenn pro Sekunde 25 Bilder gezeigt werden.



2. HD720:

Ein ebenfalls verbreitetes Videoformat ist HD720. Damit meint man 720 Bildzeilen bei einem Seitenverhältnis 16:9

- Berechnen sie Anzahl Pixel pro Bildzeile
- Berechnen sie den Speicherbedarf für ein einziges Bild, wenn jedes Pixel 3 Byte (RGB) benötigt.
- Berechnen sie die Speicherersparnis in % pro Bild gegenüber HD1080.
- Berechnen sie den Speicherbedarf für ein Dreiminuten-Video, wenn pro Sekunde 25 Bilder gezeigt werden.
- Berechnen sie die Speicherersparnis in % des Dreiminuten-Videos gegenüber HD1080

Die Alternative zu Bitmap/Rastergrafik:

Ohne Verluste skalierbar wie dieser A-Character
(Bsp.: Vektorgrafik z.B. SVG, Font/Schriftarten)



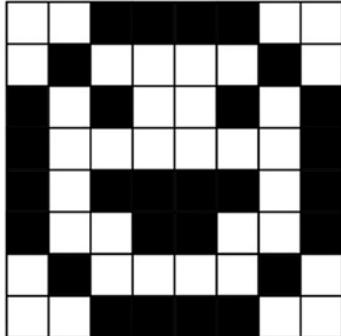
Skalierbar = Vergrösserbar, Verkleinerbar

Siehe Font-Manager Schriftarten wie z.B. TrueType ttf, OpenType otf, Postscript, etc

Rastergrafik oder Bitmap



Originalbild



Bitmapbild

0	0	1	1	1	1	0	0
0	1	0	0	0	0	1	0
1	0	1	0	0	1	0	1
1	0	0	0	0	0	0	1
1	0	1	1	1	1	0	1
1	0	0	1	1	0	0	1
0	1	0	0	0	0	1	0
0	0	1	1	1	1	0	0

Bitmap Binär

3	C
4	2
A	5
8	1
B	C
9	9
4	2
3	C

Bitmap Hex

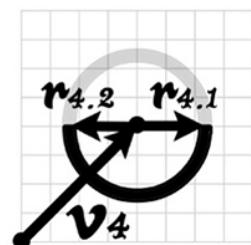
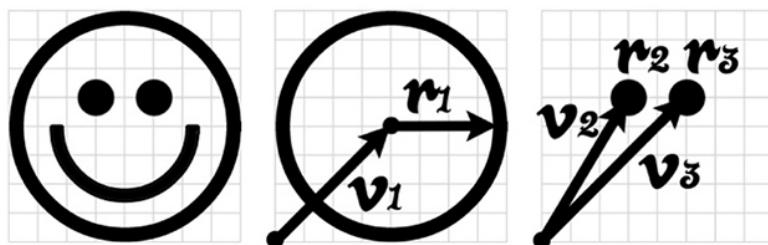


250x250Pixel

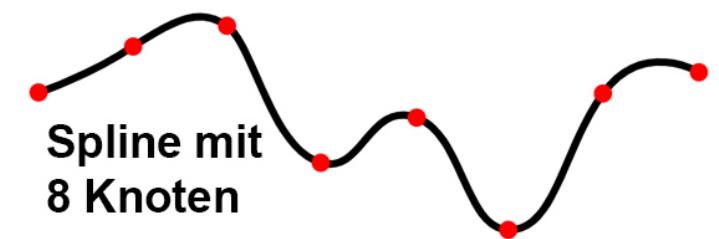


25x25Pixel

Vektorgrafik



$$\begin{aligned}v_1 &= (4/4) \quad r_1 = 4 \\v_2 &= (3/5) \quad r_2 = 0.5 \\v_3 &= (5/5) \quad r_3 = 0.5 \\v_4 &= (4/4) \quad r_4 = 2.5 \\r_{4.1} &= (6.5/4) \quad r_{4.2} = (1.5/4)\end{aligned}$$



Typische Anwendung von Vektorgrafiken → Schriftzeichen (Font)



Geometrische
Beschreibung
des Buchstabens
(Vektorgrafik)



So sähe es aus,
wenn der Buchstabe als
Rastergrafik abgerufen
und anschliessend stark
vergrössert würde:
Es entstehen
Treppenartefakte.

Vektorgrafik im Web mit SVG (Scalable Vector Graphics / Skalierbare Vektorgrafik)

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE svg PUBLIC
  "-//W3C//DTD SVG 1.1//EN"
  "http://www.w3.org/Graphics/SVG/1.1/DTD/svg11.dtd" >

<!-- Ein Smiley als SVG-Vektorgrafik-->
<svg xmlns="http://www.w3.org/2000/svg" width="240" height="240">

<!--Grundflaeche-->
<rect x="0" y="0" width="240" height="240" fill="lightgray"></rect>

<!--Kopfform-->
<circle r="100" cx="120" cy="120" fill="white" stroke="black" stroke-width="12" />

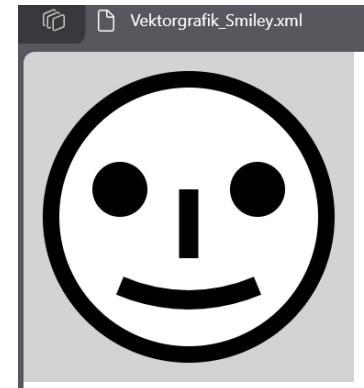
<!--Augen-->
<circle r="20" cx="170" cy="100" fill="black" />
<circle r="20" cx="70" cy="100" fill="black" />

<!--Nase-->
<line x1="120" y1="100" x2="120" y2="150" stroke="black" stroke-width="14px"></line>

<!--Mund-->
<path d="M 70 170 A 150 180 0 0 0 170 170" style="fill:none; stroke: black; stroke-width: 14;"/>

</svg>
```

Resultat im
Webbrowser:



Und nun wieder sie: *(Dauer: 10 Min.)*

Wer macht die schönste Web-Vektorgrafik?

Erstellen sie ein XML-Dokument mit einer SVG-Grafik.

Ungefähr so, wie der Smiley von vorher, aber anders, denn der war meine Idee ;-)



Wenn's mal rund sein soll:

Webseite mit einem runden Button



Vordergrund



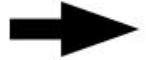
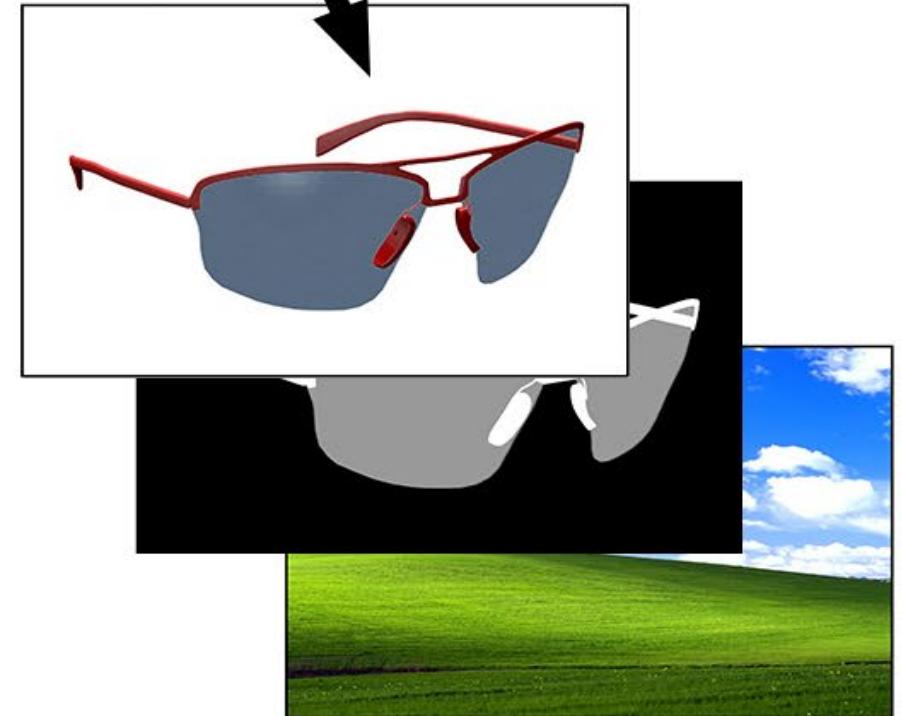
Alphakanal



Hintergrund



Komposition



Das Resultat

Green-Screen

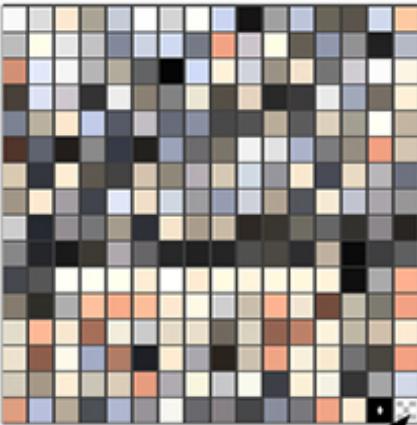


Auch anzutreffen
bei Kino und TV

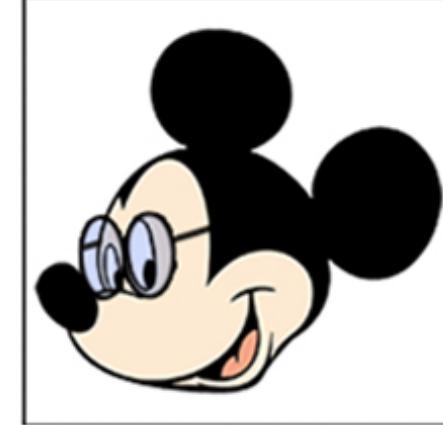
GIF-Bild



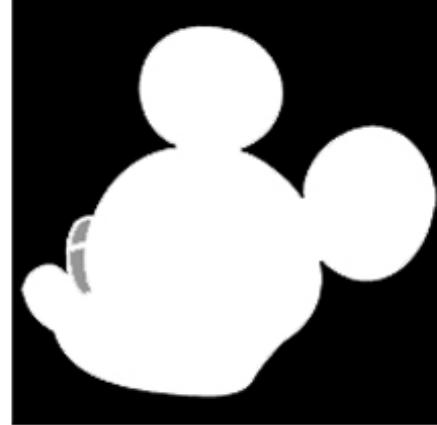
Farbtabelle



TIF-Bild



Alpha-Kanal



Harte Übergänge / Treppen
Keine Halbtransparenz
GIF: Webbrowserunterstützt

Weiche Übergänge
Halbtransparenz
TIF: Nicht Webbrowserunterstützt
PNG: Webbrowserunterstützt

Fassen wir zusammen: *(Dauer: 5 Min.)*

BILD-FORMAT	Von aktuellen Webbrowsen runterstützt [J/N]	Raster-grafik [J/N]	Vektor-grafik [J/N]	α-Kanal [J/N]	Transparenzfarbe [J/N]	Geeignet für...
JPG						
GIF						
PNG						
TIF						
BMP						
WEBP						
SVG						





Noch Fragen?

Photographie



Grammophon



Cinematographie



Television



LCD-Display



+

1800

+

+

+

+

1850

+

+

+

+

+

1900

+

+

+

+

+

1950

+

+

+

+

2000

SW

Color

Photographie



Grammophon



Cinematographie



Television



LCD-Display



1800

1850

1900

1950

2000

SW

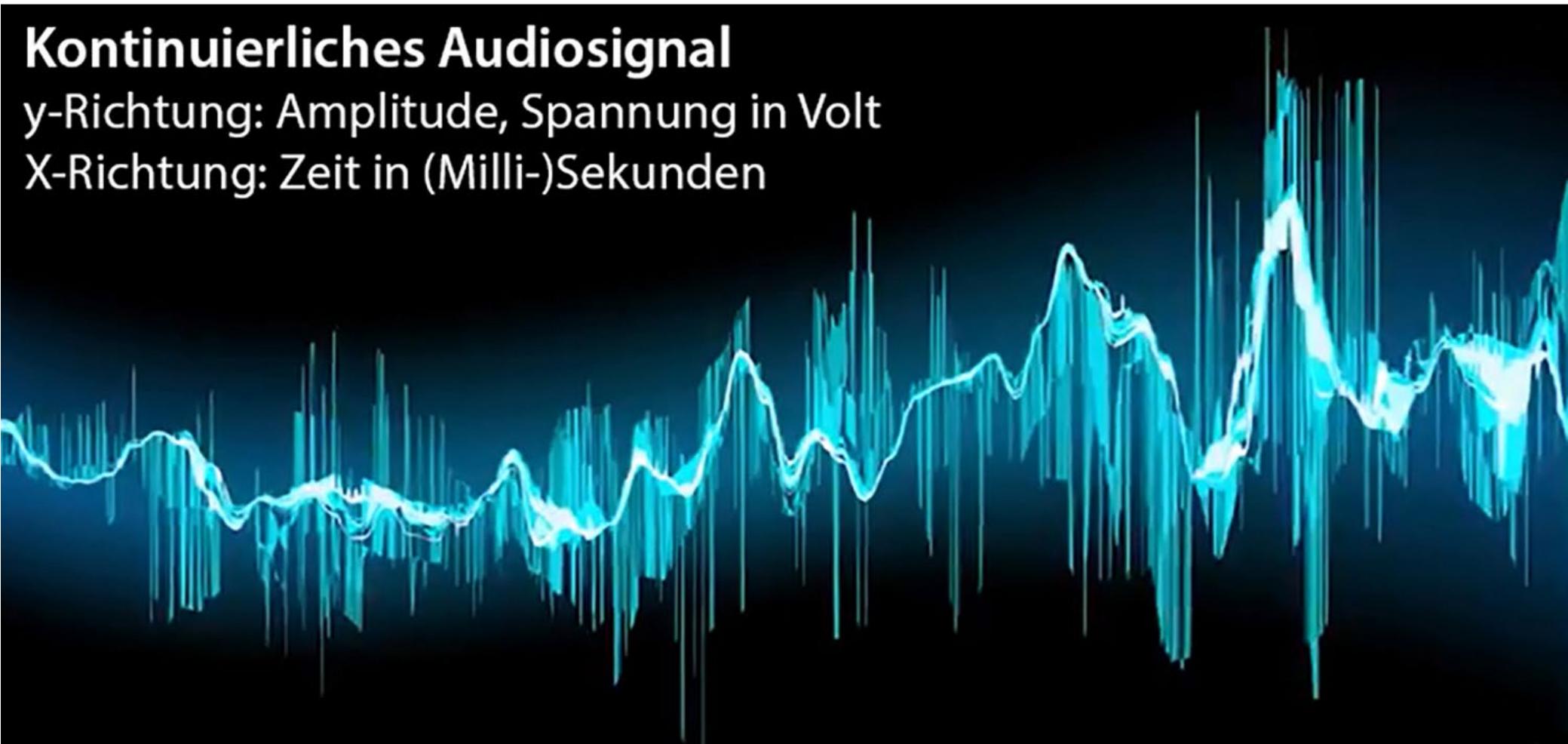
Color

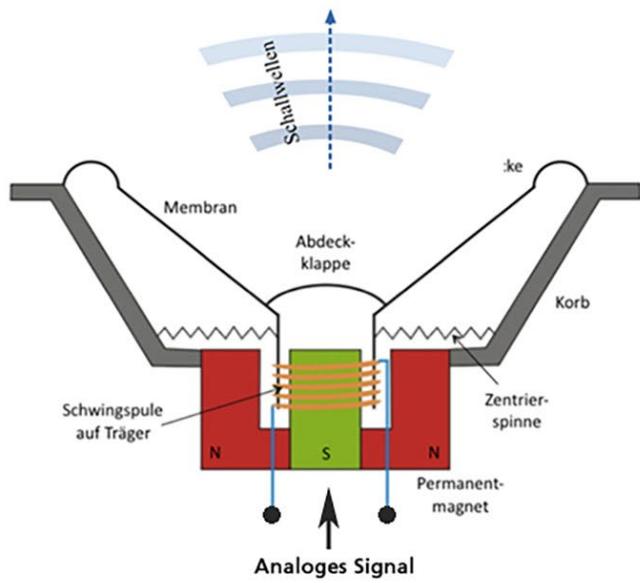
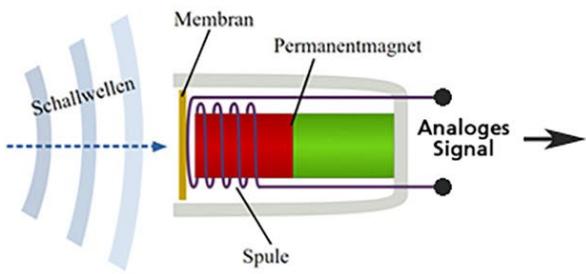


Kontinuierliches Audiosignal

y-Richtung: Amplitude, Spannung in Volt

X-Richtung: Zeit in (Milli-)Sekunden

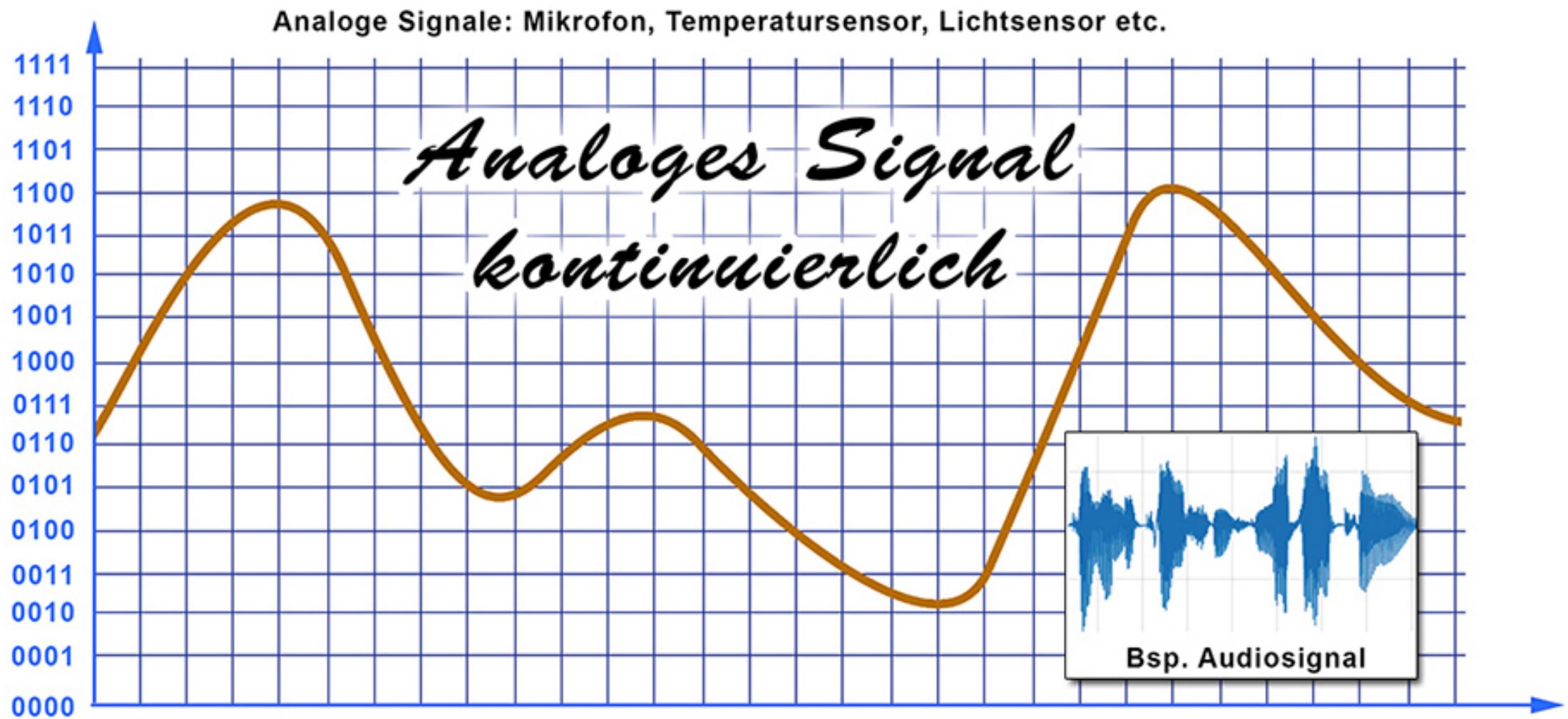




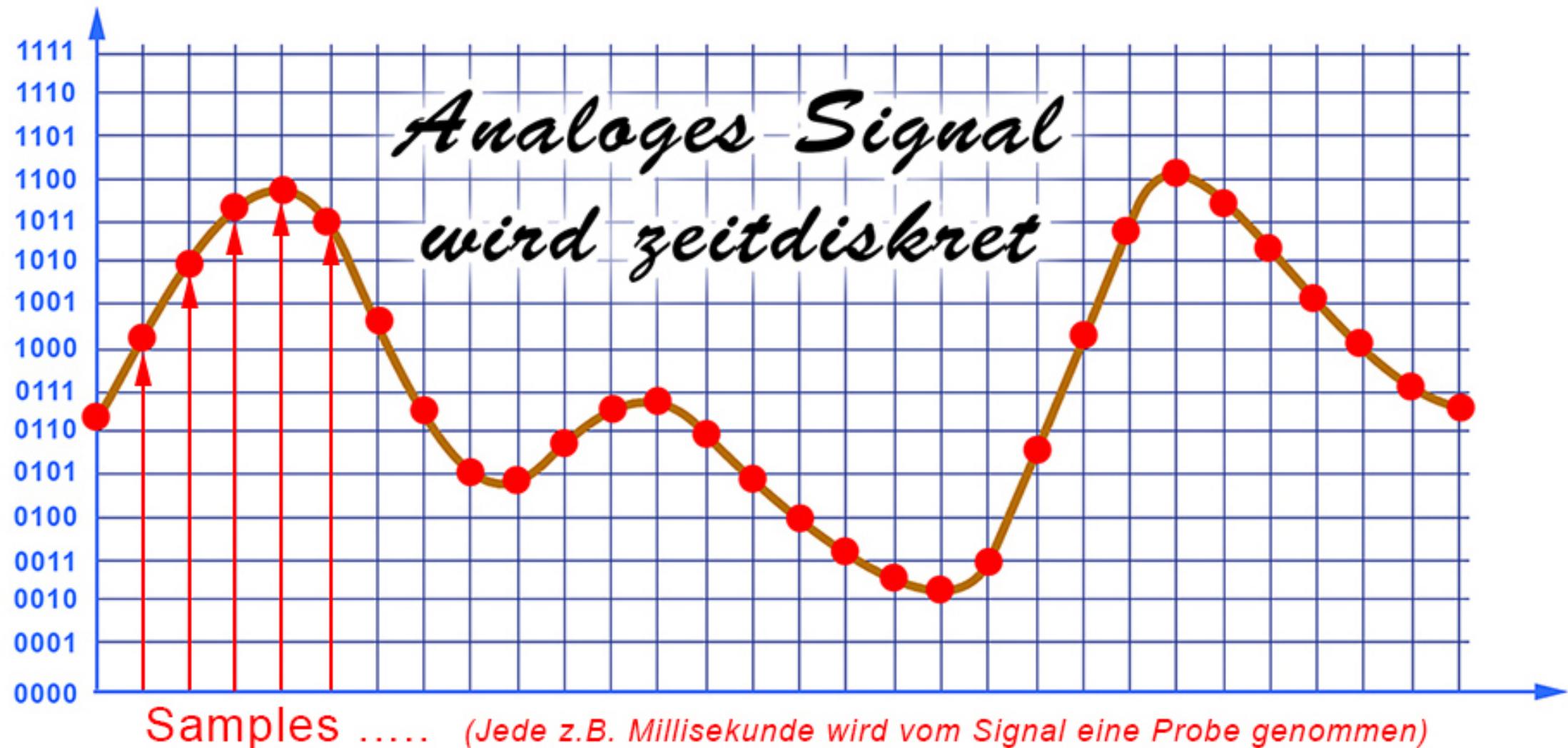
Analog/Digital-Wandlung

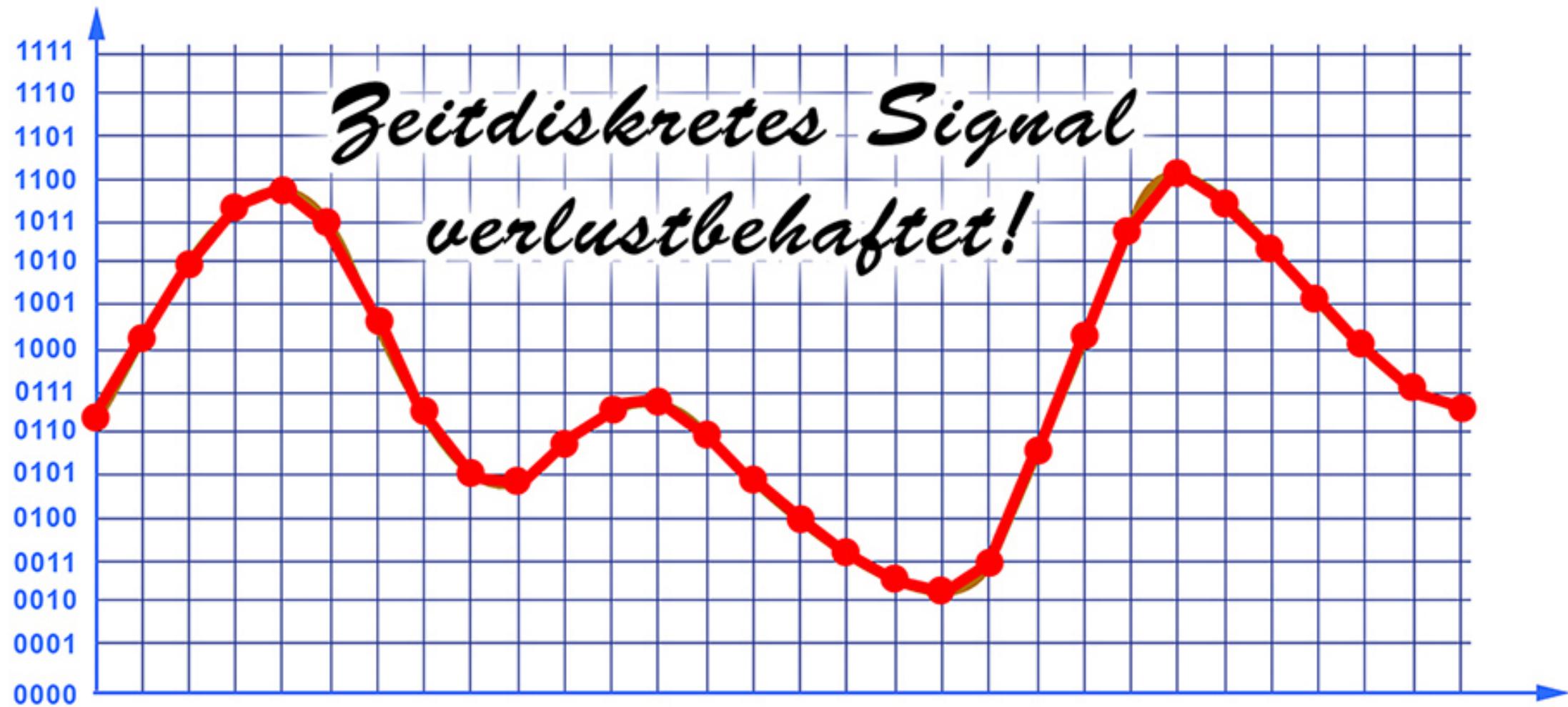


Analog/Digital-Wandlung



Analog/Digital-Wandlung





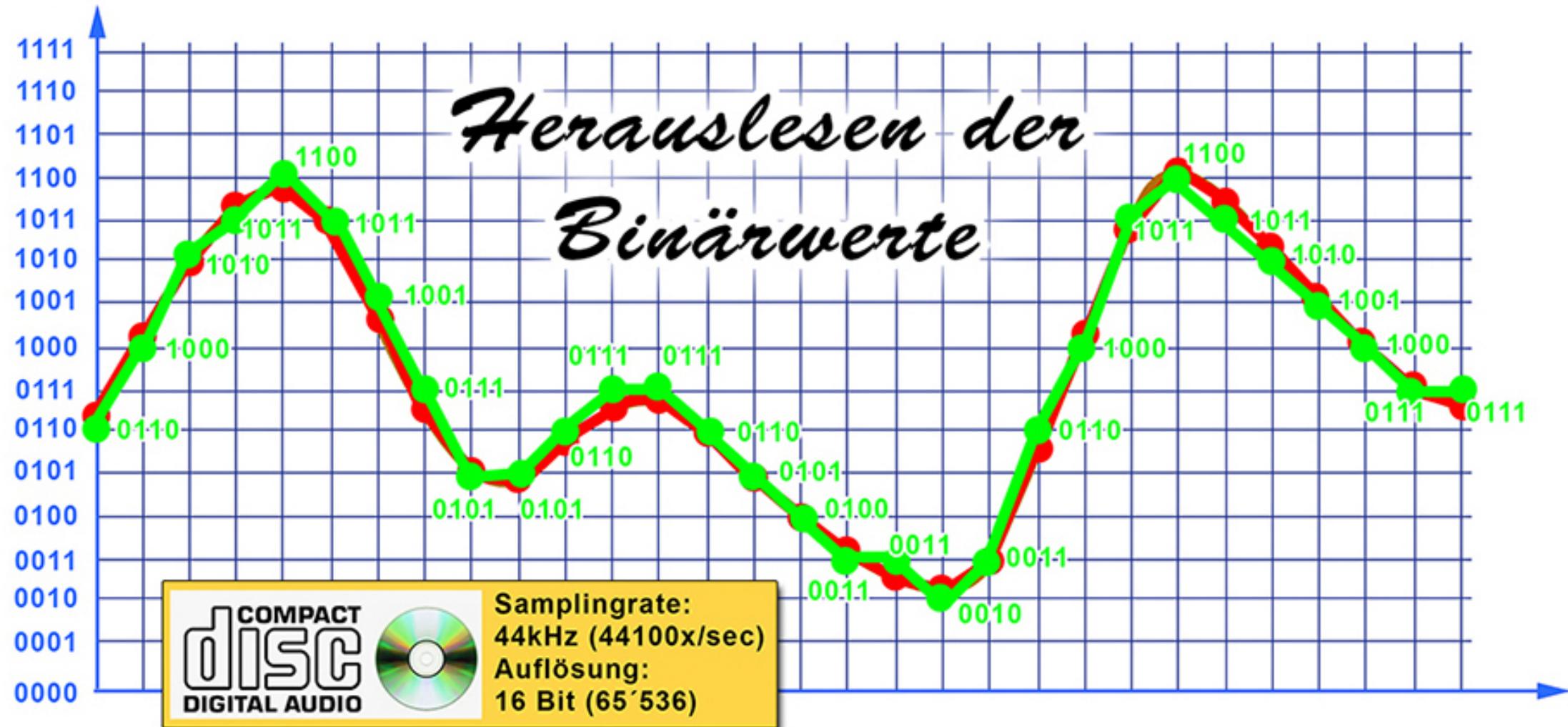
Analog/Digital-Wandlung

Braune, rote und grüne Linie sind nicht deckungsgleich. Das bedeutet: Informationsverluste!



Analog/Digital-Wandlung

Je höher die Auflösung in X und Y, umso akurater das digitale Abbild des analogen Signals.



Art der Schwingung

Sinus → Dreieck → Rechteck



Mensch → Rauschen
100-600Hz → Alle Frequenzen

Frequenzen

100Hz → 440Hz → 1kHz → 5kHz → 10kHz → 12kHz → 14kHz → 20kHz



20kHz → 10kHz → 5kHz → 1kHz



**Tiefpass
Obere
Grenzfrequenz**

**Zielbitrate
pro Sekunde**

96kb/s

48kb/s

20kb/s

**Gleiche
Dateigrösse:
Was bringt
mehr?**

44.1kHz bei 8Bit
(352'800b/s)

11.025kHz bei 32Bit
(352'800b/s)

Photographie



Grammophon



Cinematographie



Television



LCD-Display



+

1800

+

+

+

+

1850

+

+

+

+

+

1900

+

+

+

+

+

1950

+

+

+

+

2000

SW

Color

Photographie



Grammophon



Cinematographie



Television



LCD-Display



+

1800

+

+

+

+

1850

+

+

+

+

+

1900

+

+

+

+

+

1950

+

+

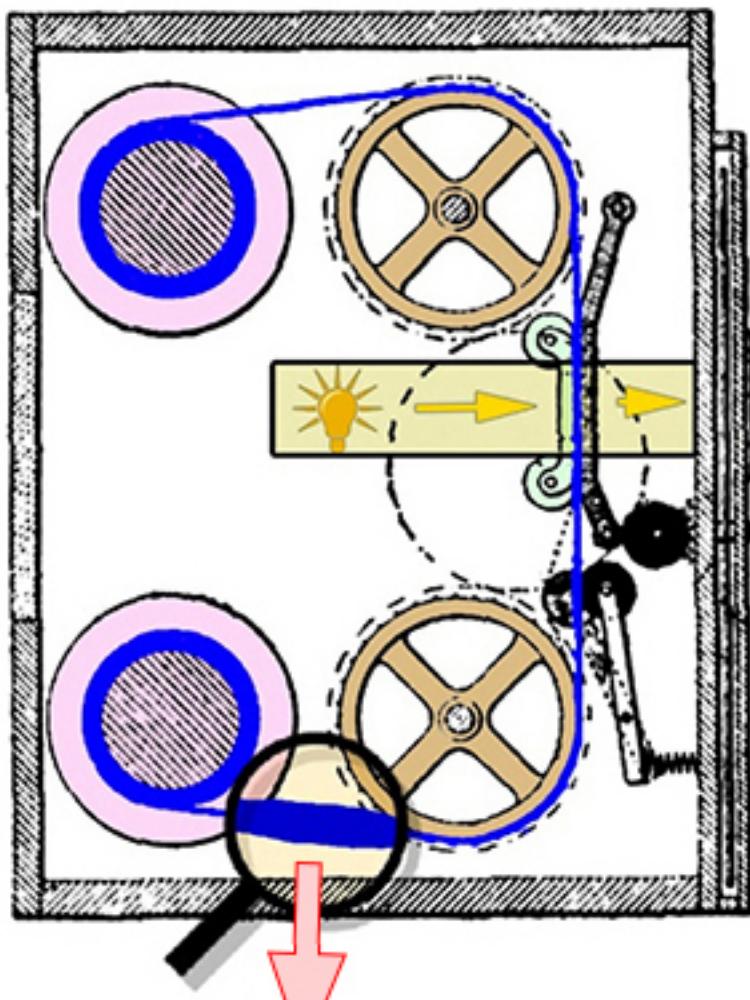
+

+

2000

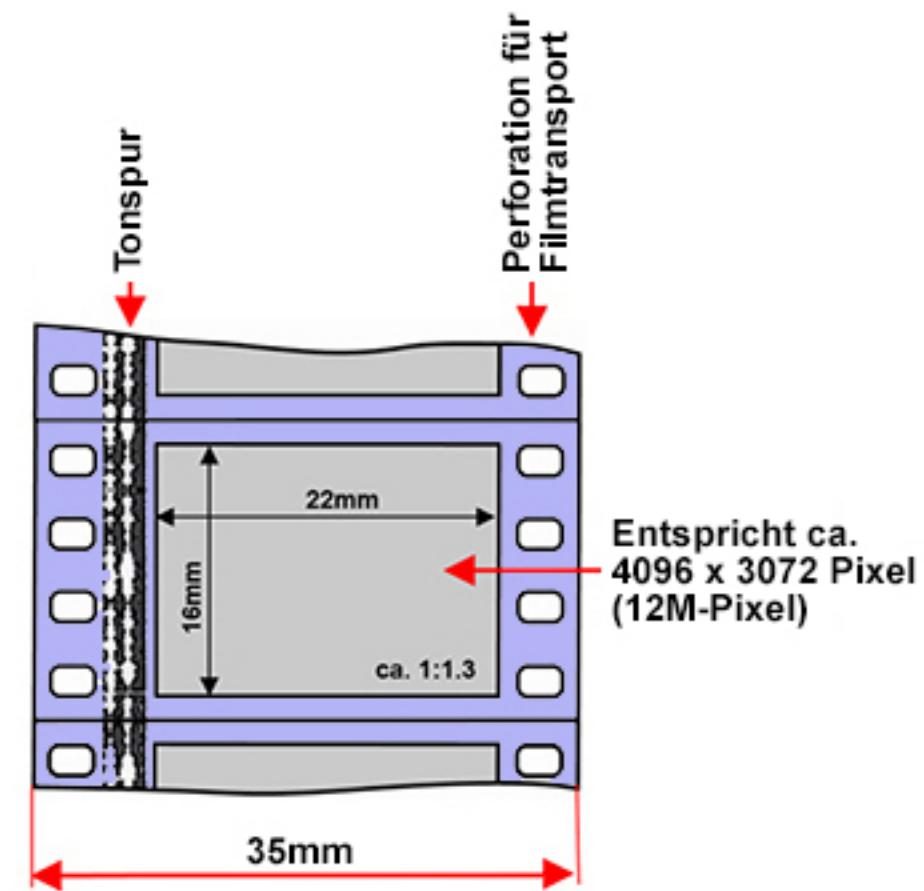
SW

Color



Jedes Bild wird
2x projiziert

24 Bilder pro Sekunde



Photographie



Grammophon



Cinematographie



Television



LCD-Display



+

1800

+

+

+

+

1850

+

+

+

+

+

1900

+

+

+

+

+

1950

+

+

+

+

2000

SW

Color

Photographie



Grammophon



Cinematographie



Television



LCD-Display



+

1800

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

1900

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

+

2000

SW

Color

Interlaced / Halbbildverfahren:

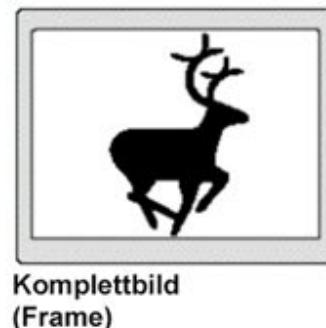


Fernsehen:
Historisch mit
50 Halbbildern
pro Sekunde



Kammbildung beim
Halbbildverfahren
bei schnellen Be-
wegungsabläufen.
Milderung dieses
Effekts bietet das
Deinterlacing.

Progressive Scan / Vollbildverfahren:



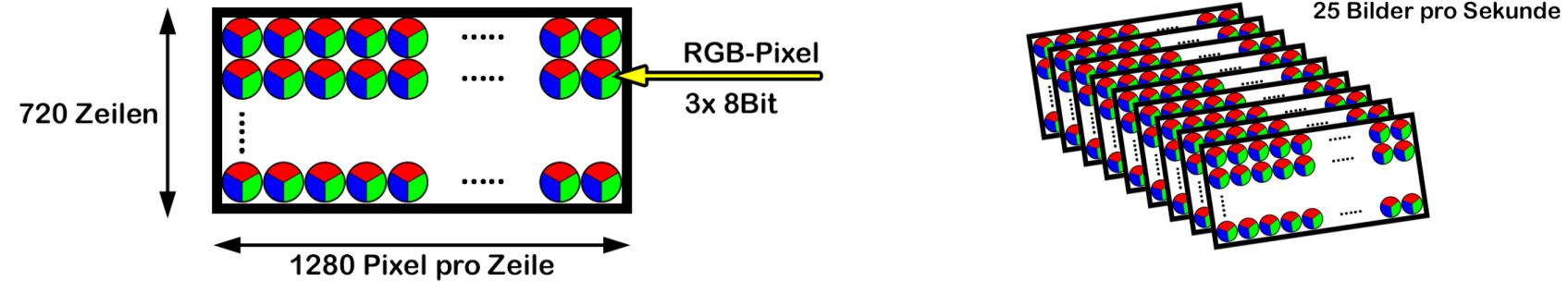
Auswahl aktueller Videonormen:

Norm	Auflösung	Progressiv	Interlaced	Format	Bitrate
NTSC-SD	640 x 480		60 fps	4:3	
PAL-SD	768 x 576		50 fps	4:3	
ATSC-HD-720p60	1280 x 720	60 fps		16:9	19 Mb/s
ATSC-HD-720i60	1280 x 720		60 fps	16:9	
EBU-HD-720p50	1280 x 720	50 fps			
ATSC-HD-1080p60	1920 x 1080	60 fps		16:9	25Mb/s
ATSC-HD-1080i60	1920 x 1080		60 fps	16:9	
EBU-HD-1080p50	1920 x 1080	50 fps		16:9	
EBU-HD-1080i50	1920 x 1080		50 fps	16:9	25Mb/s
UHD-1 (4k)	3840 x 2160	≤ 120 fps			10.2Gb/s
UHD-2 (8k)	7680 x 4320	≤ 120 fps			24 Gb/s

SD=Standard Definition / **HD**=High Definition / **NTSC**=National Television System Committee / **PAL** Phase Alternating Line / **ATSC**=Advanced Television System Committee / **EBU**=European Broadcasting Union / **UHD**=Ultra High Definition

Und schon wieder sie: *(Dauer: 7 Min.)*

Wir haben soeben ein Videoclip (Unkomprimiert, HD720i50) von 90 Minuten Dauer abgedreht



Berechnen sie die Speicherkapazität bzw. Downloadrate für

1. Ein Bild:
2. 1 sec. Video:
3. 90 min. Video:
4. Harddiskspeicherplatz:
5. BluRay ($4\text{Layer} = 100\text{GB}$) :
6. Videostreaming Mb/s:
7. Download mit Internetgeschwindigkeit 1Gb/s:
8. Schlussfolgerung:

Verlustbehaftete Komprimierung bei Bild und Ton?

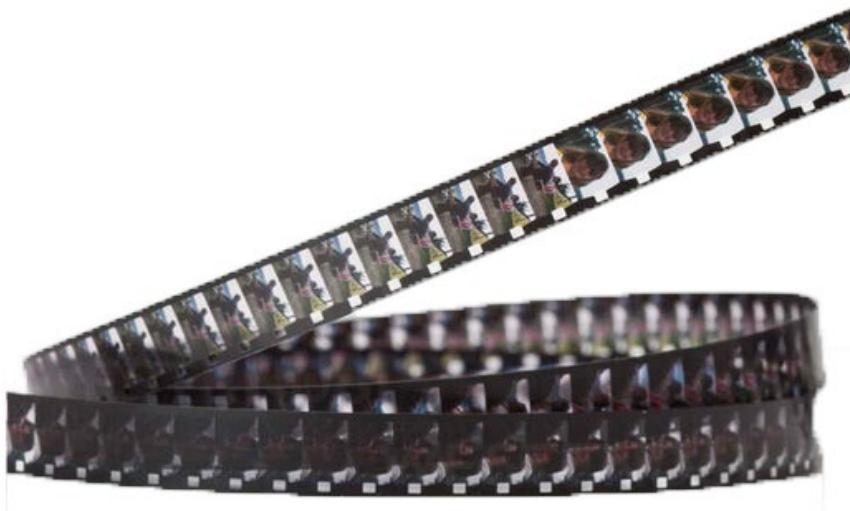


Wieviel kann komprimiert werden, ohne das es weh tut?

Bildkomprimierung erfolgt...



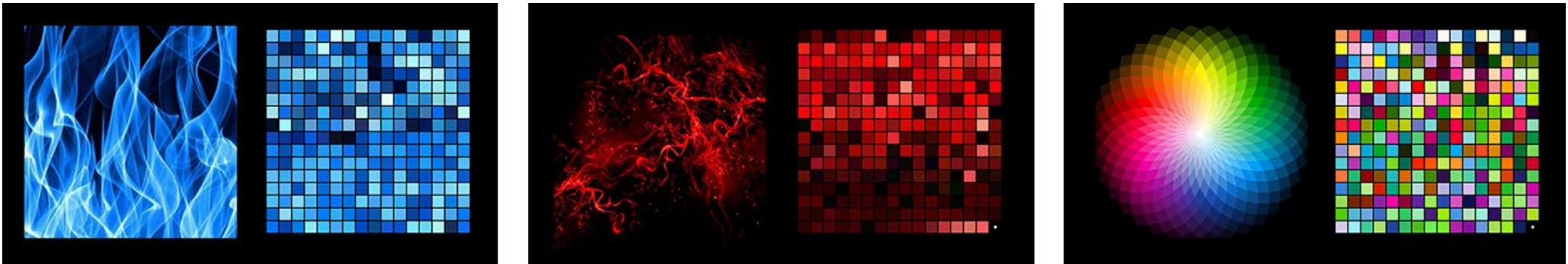
**... innerhalb eines Bildes
(Intraframe)**



**... über eine Bildserie
(Interframe)**

Naheliegende Verfahren:

- Intraframe: Bildgrösse / Farbauflösung reduzieren
- Interframe: Bildwiederholrate reduzieren. Z.B. 16B/s anstatt 25B/s
- Audio: Samplingrate reduzieren, Mono anstatt Stereo
- Intraframe: Eine Farbtabelle benutzen, wie z.B. bei GIF



- Intraframe: Anstatt Farbe (RGB=3B/Pixel) nur Graustufen (1B/Pixel)



Ein erstes verlustbehaftetes Bildkomprimierungsverfahren ist **Subsampling**.

Um das zu verstehen, müssen wir neben RGB und CMYK einen weiteren Farbraum einführen:

Helligkeit-Farbigkeit-Modelle oder Luminanz-Chrominanz-Modell YCrCb

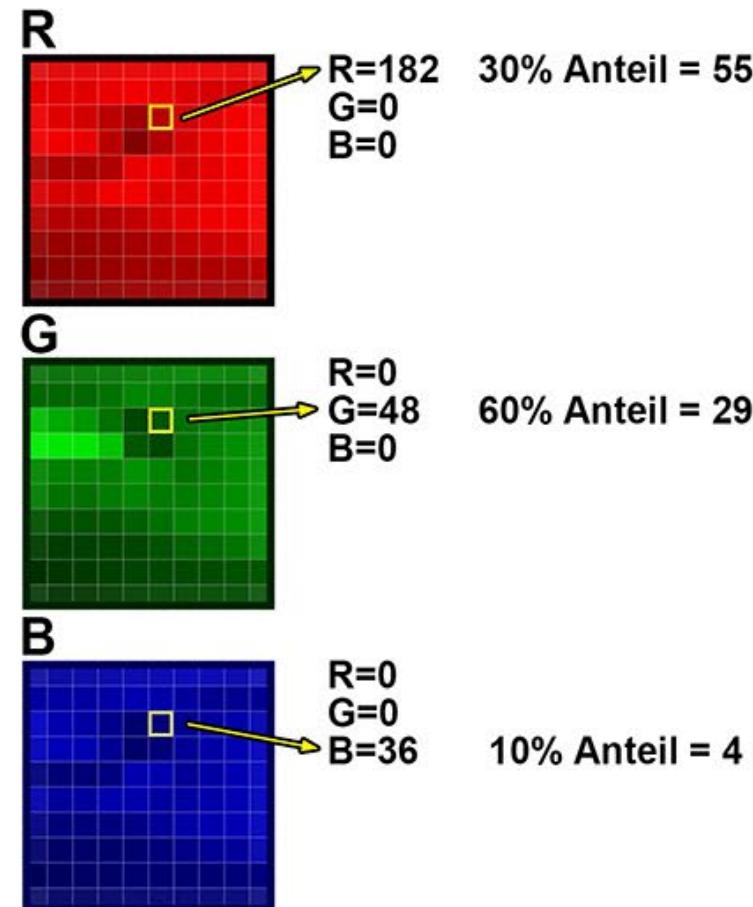
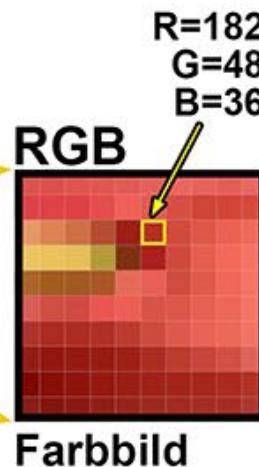
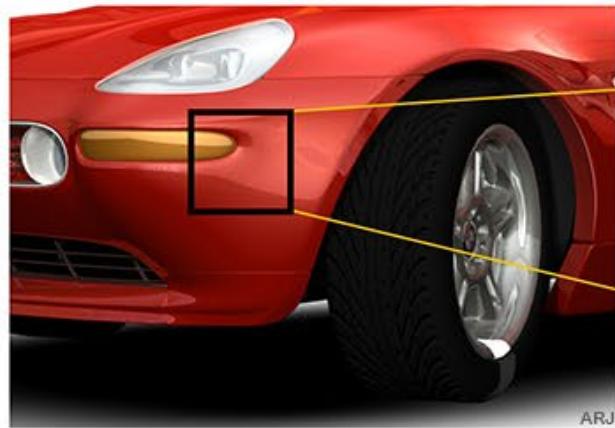
Y=Luminanz (Graustufen)

Cr=Chrominanz Rot

Cb=Chrominanz Blau

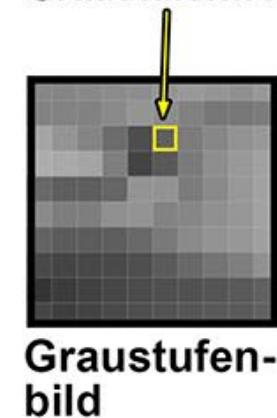
Also ebenfalls 3 Kanäle wie RGB!

Luminanzberechnung (Graustufen):



$$\begin{array}{r} 55 + \\ 29 + \\ 4 = \\ \hline 88 \end{array}$$

Graustufenwert = 88



Weil's gerade passt, hier ein paar Aufgaben: *(Dauer: 7 Min.)*

Welchen **YC_bCr-Werten** und **Farbe** entsprechen diese **RGB-Farbcodes**?

(Bei dieser Aufgabe kann Ihnen <https://colorizer.org/> weiter helfen.)

1. #000000 =
2. #FFFFFF =
3. #FF0000 =
4. #00FF00 =
5. #0000FF =



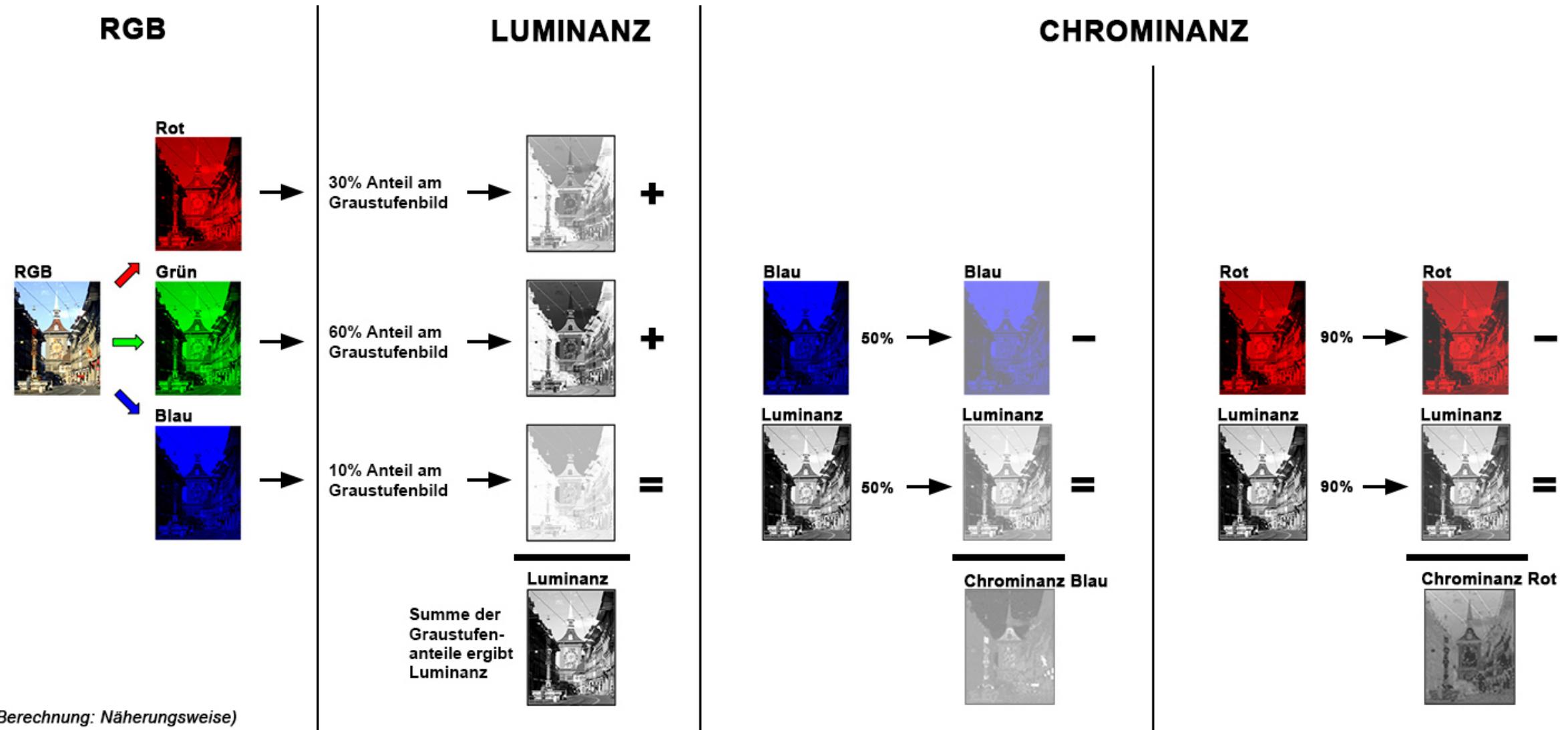
Welchen Graustufenwerten oder Luminanzwerten (0..255) entsprechen diese **RGB-Farbcodes**?

1. #000000 =
2. #FFFFFF =
3. #FF0000 =
4. #CD2FC0 =

Verständnisfrage: Warum hat bei der Umwandlung eines Farbbildes in ein Graustufenbild der Grünanteil am meisten Gewicht?

Chrominanzberechnung (Farbanteile):

(Nur zur Info, wird nicht geprüft!)

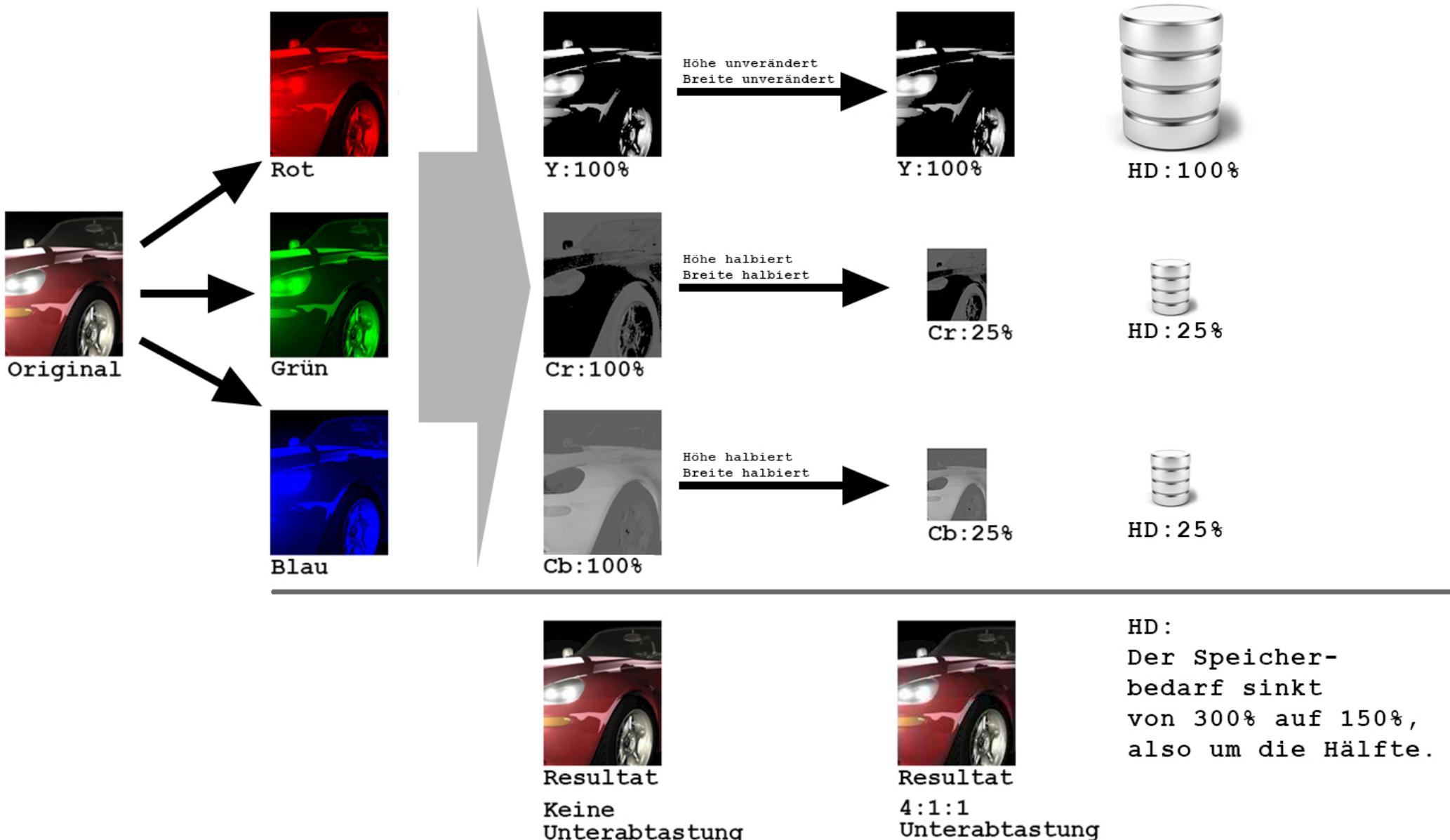


Und was gewinnen wir dabei?

Im Gegensatz zu RGB können wir in einzelnen Kanälen die Auflösung (Anzahl Pixel) reduzieren und so Speicherplatz sparen.

Und so geht das:

Chroma-Subsampling/Farbunterabtastung



Das soll mit ein paar Aufgaben nun vertieft werden: *(Dauer: 7 Min.)*

Berechnen sie die Speichereinsparung in % gegenüber dem RGB-Originalbild.

1. Bei Subsampling 4:4:4
2. Bei Subsampling 4:2:2
3. Bei Subsampling 4:1:1
4. Bei Subsampling 4:2:0



Verständnisfragen:

1. Kann man durch die Bildumwandlung vom RGB- in den YCbCr-Farbraum Speicherplatz einsparen?
2. Warum verschlechtert sich die Bildschärfe von 4:1:1-Subsampling gegenüber 4:4:4-Subsampling nicht?
3. Kann ein Video-Beamer ein Bild im YCbCr-Farbbereich darstellen?



Fragen?

Intraframekomprimierung am Beispiel JPG

Strategie: Das Bild derart "umwandeln", dass RLE und Huffman effizient wirken können. Ähnlich wie bei der Burrows-Wheeler-Transformation.

Umsetzung: DCT-Algorithmus (**D**irect **C**osinus **T**ransformation)

Um die Datenreduzierung bei JPG zu verstehen, müssen wir noch einen Begriff klären:

Quantisierung, aber was versteht man darunter?

Dazu ein Vergleich mit **Schulnoten**:

Notenskala ungerundet: 1.0 bis 6.0 Dies entspricht 50 Noten.

Notenskala gerundet auf halbe Noten. Dies entspricht 11 Noten, nämlich 1 / 1.5 / 2 / 2.5 / 3 / 3.5 / 4 / 4.5 / 5 / 5.5 / 6

Beispiel

Name	Note ungerundet (6Bit oder 0..63)	Note gerundet (4Bit oder 0..15)	Bemerkung
Hans	4.3	4.5	Informationsverlust, wenn gerundet
Susi	5.2	5.0	Informationsverlust, wenn gerundet
Felix	5.8	6.0	Informationsverlust, wenn gerundet

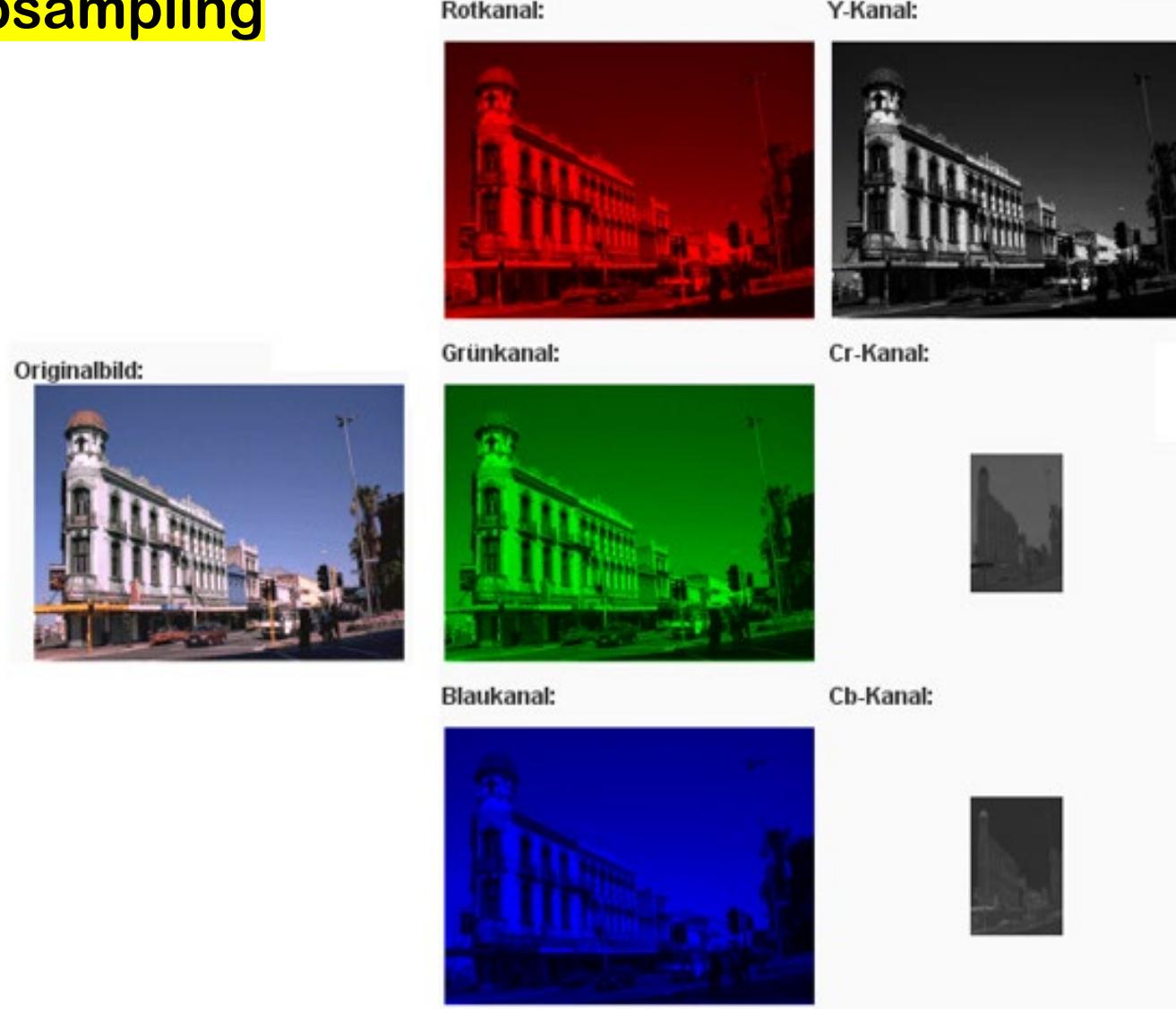
Der **Benutzer** des
Grafikprogramms
kommt von allem
selbstverständlich nix mit.

Er hat nur einen
Schieberegler für
die **Qualitätseinstellung**.

Was aber steckt dahinter?

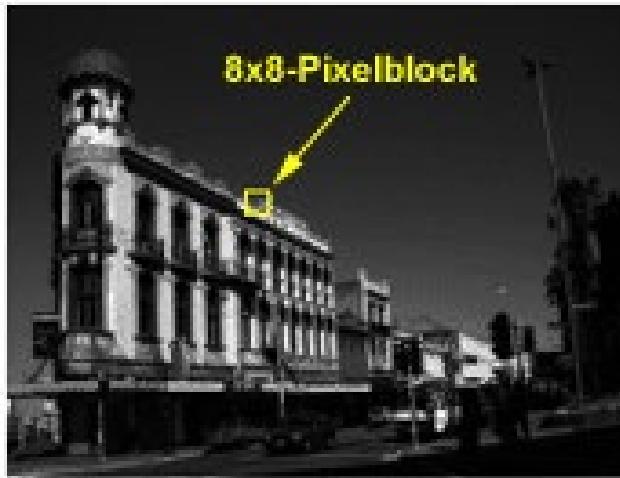


1. Schritt: Subsampling



Die folgenden Bilder zeigen den Komprimierungsvorgang im Y-Kanal. Bei den beiden anderen Kanälen C_b und C_r wird nach demselben Verfahren vorgegangen.

2. Schritt: 8x8Pixel Blockbildung und DCT-Transformation



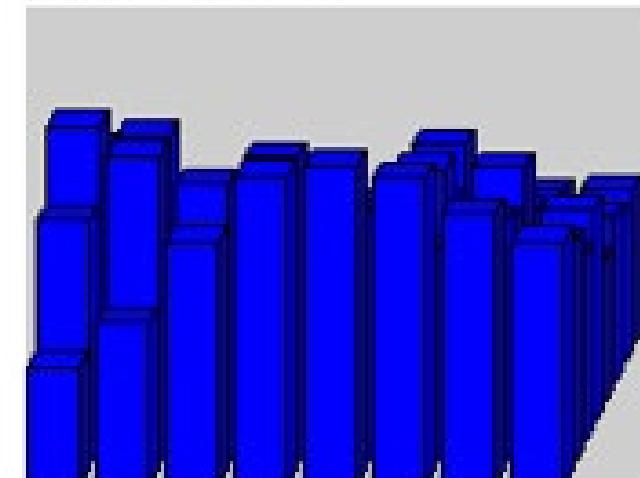
Originalwerte im 8x8 Block

114	115	117	114	109	117	113	115
120	114	117	117	121	113	120	110
118	119	121	114	117	112	119	126
163	149	141	115	130	193	143	106
204	163	133	155	141	163	191	159
236	228	178	163	186	204	163	138
180	228	208	227	186	157	179	163
86	119	179	228	236	227	200	179

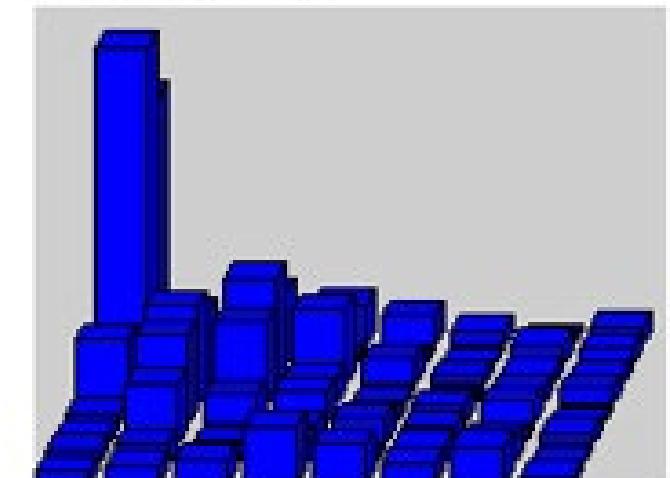
DCT-transformierte Werte

188	15	-22	32	-23	12	-6	-14
-239	10	55	4	3	-2	16	13
-11	-54	-81	-57	20	-8	-9	7
50	74	62	19	-3	2	-11	-12
-8	-67	-22	24	-4	15	17	-2
7	46	2	-14	-20	-13	1	-8
-11	-14	-3	16	-2	-13	18	-1
-9	-13	18	-43	29	15	-28	6

3D-Visualisierung



3D-Visualisierung



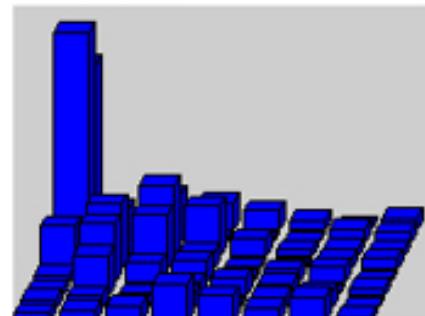
3. Schritt: Quantisierung



DCT - Matrix:

188	15	-22	32	-23	12	-6	-14
-239	10	55	4	3	-2	16	13
-11	-54	-81	-57	20	-8	-9	7
50	74	62	19	-3	2	-11	-12
-8	-67	-22	24	-4	15	17	-2
7	46	2	-14	-20	-13	1	-8
-11	-14	-3	16	-2	-13	18	-1
-9	-13	18	-43	29	15	-28	6

Visualisierung:



Niedrige Komprimierung

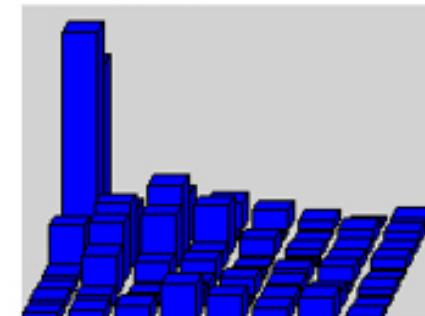
Quantisierungsmaatrix:

1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1
1	1	1	1	1	1	1	1

Quantisierte Matrix:

188	15	-22	32	-23	12	-6	-14
-239	10	55	4	3	-2	16	13
-11	-54	-81	-57	20	-8	-9	7
50	74	62	19	-3	2	-11	-12
-8	-67	-22	24	-4	15	17	-2
7	46	2	-14	-20	-13	1	-8
-11	-14	-3	16	-2	-13	18	-1
-9	-13	18	-43	29	15	-28	6

Visualisierung:



Mittlere Komprimierung

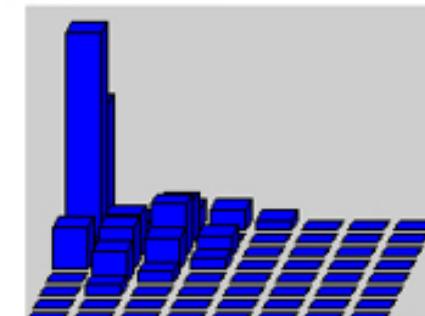
Quantisierungsmaatrix:

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

Quantisierte Matrix:

12	1	-2	2	-1	0	0	0
-20	1	4	0	0	0	0	0
-1	-4	-5	-2	0	0	0	0
4	4	3	1	0	0	0	0
0	-3	-1	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

Visualisierung:



Hohe Komprimierung

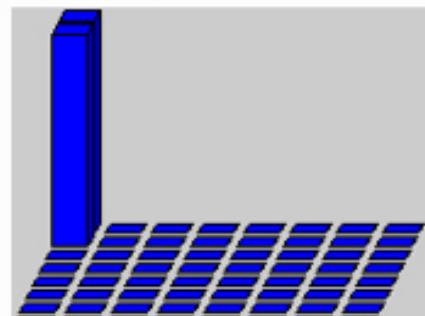
Quantisierungsmaatrix:

255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255

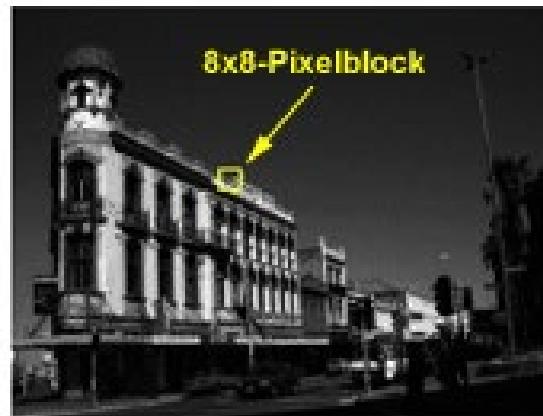
Quantisierte Matrix:

1	0	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	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

Visualisierung:



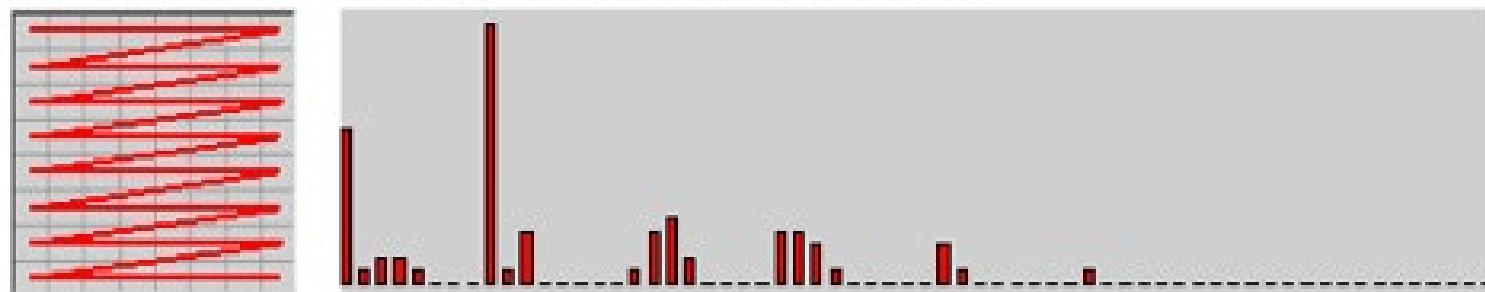
4. Schritt: Zick-Zack-Scan



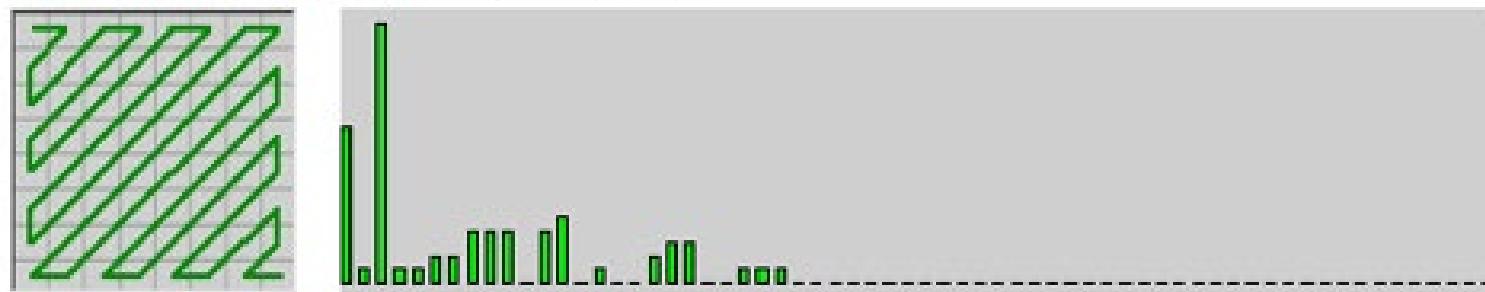
Quantisierte Matrix:

12	1	-2	2	-1	0	0	0
-20	1	4	0	0	0	0	0
-1	-4	-5	-2	0	0	0	0
4	4	3	1	0	0	0	0
0	-3	-1	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

Matrixwerte von links nach rechts und von oben nach unten:

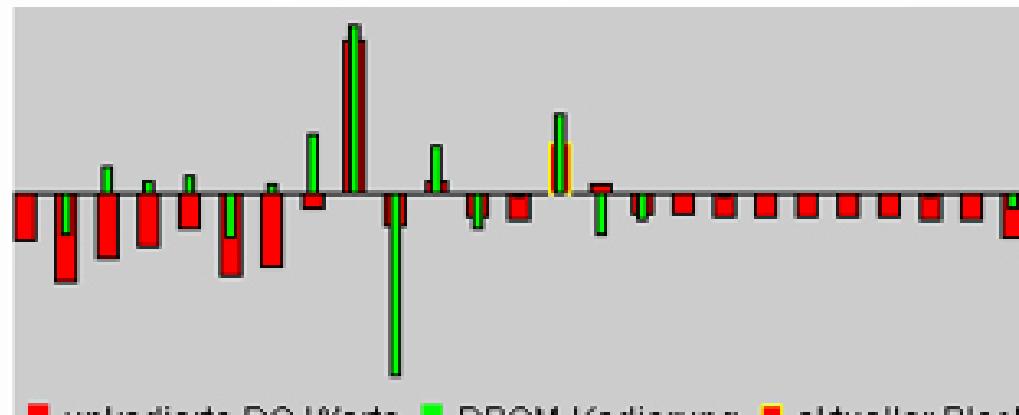


Matrixwerte mit Zick-Zack-Scan Methode:



5. Schritt: Datenreduktion RLC

DPCM-Kodierung des DC-Wertes jedes Blocks:



RLE- und Variable-Length-Integer-Kodierung der AC-Werte:

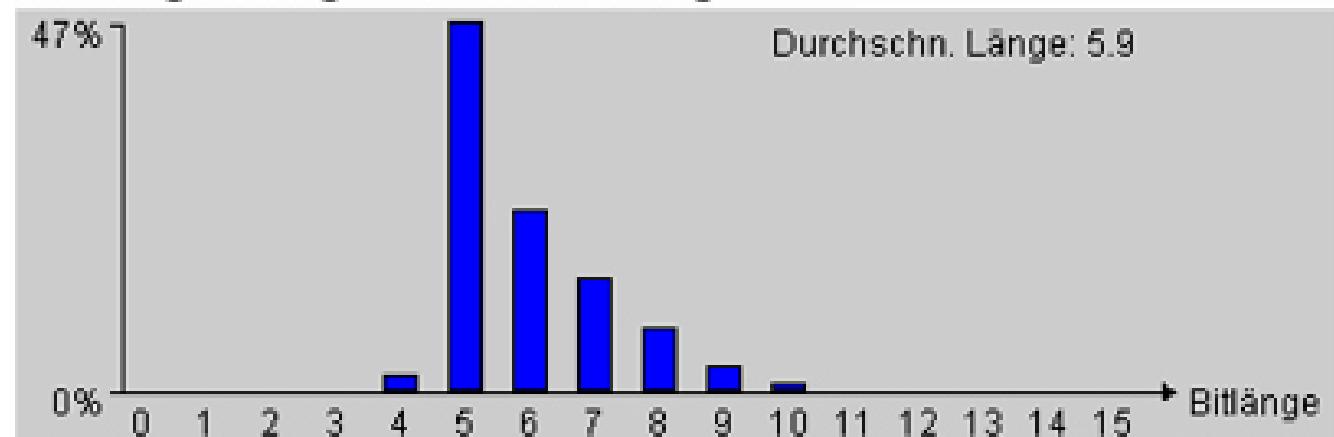
Vor Run Length-Encoding:

1 -20 -1 1 -2 2 4 -4 4 0 4 -5 0 -1 0 0 -2 3 -3 0 0 1 -1 1 0 0 0 0 0 0 0 0 0
0 0

Nach Run Length-Encoding:

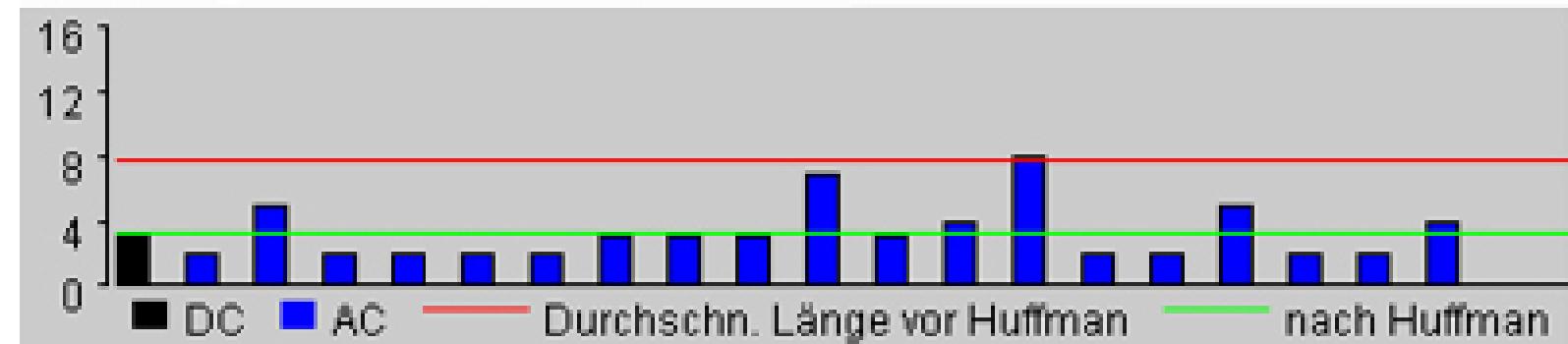
1 -20 -1 1 -2 2 4 -4 4 1 4 -5 1 -1 2 -2 3 -3 2 1 -1 1 39

Verteilung der Längen nach VLI Kodierung:

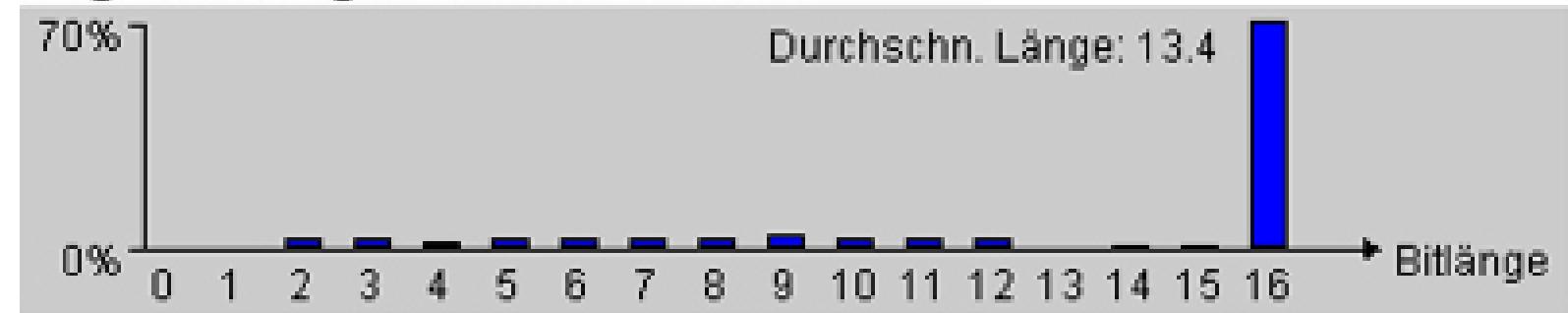


6. Schritt: Datenreduktion Huffman

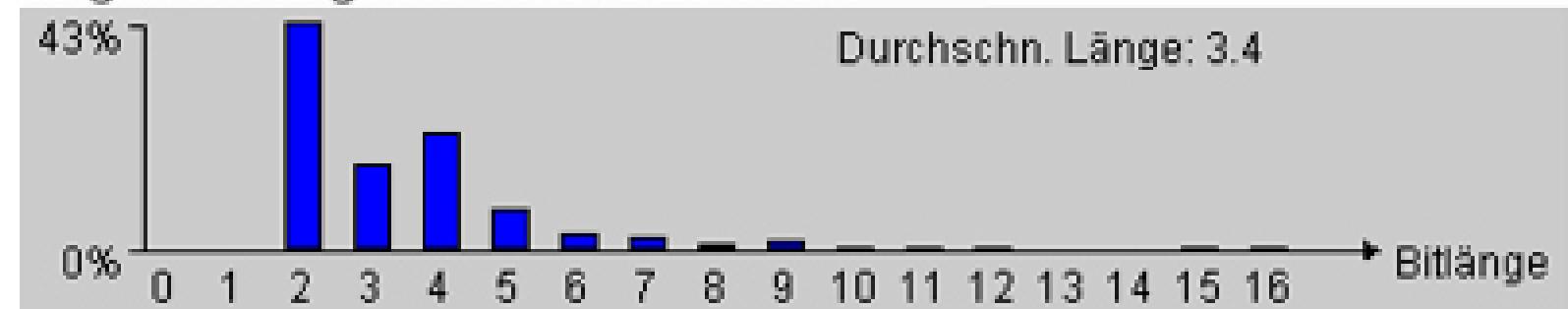
Darstellung des aktuellen Blocks:



Längenverteilung in der Huffman-Tabelle:



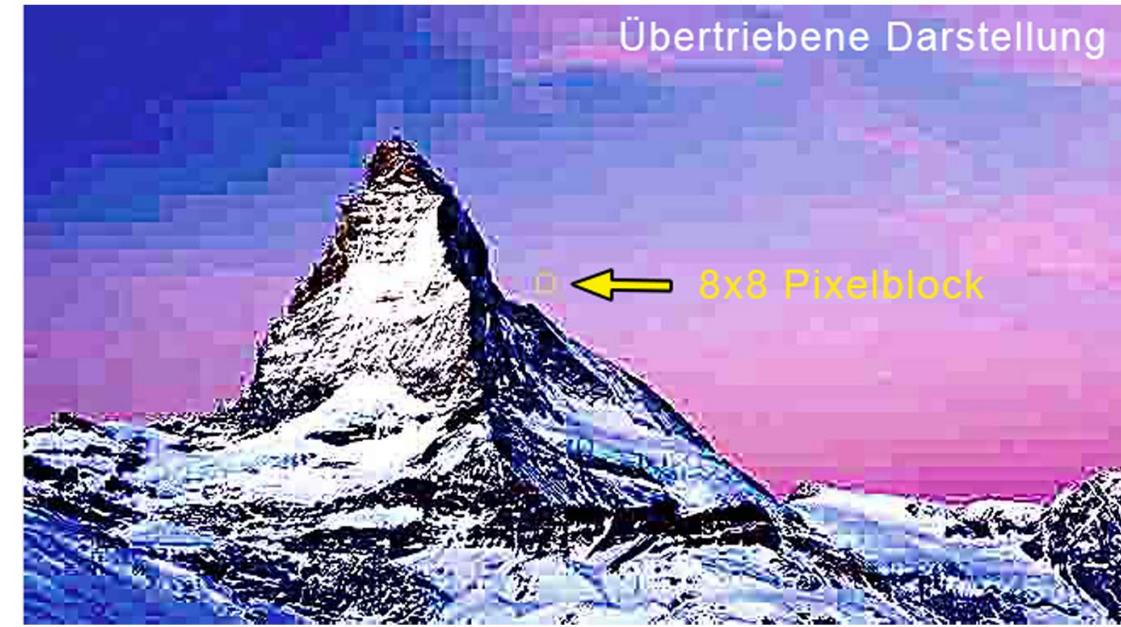
Längenverteilung im kodierten Datenstrom:



Risiko der Bildung von Artefakten



Eine starke JPG-Komprimierung führt zu unsauberen Textkonturen.
(Zur besseren Sichtbarmachung des Effekts wurde das Bild nachträglich gammakorrigiert)



Blockartefakte nach starker JPG-Komprimierung.
Führt von der 8x8-Blockbildung bei DCT.

Das Beste zum Schluss, die Kontrollfragen: *(Dauer: 5 Min.)*

Komprimierung bei JPG mit DCT?

(Bei dieser Aufgabe kann Ihnen <https://www.youtube.com/watch?v=Kv1Hiv3ox8I> weiter helfen.)

1. Welches Reduktionsverfahren wird bei der JPG-Komprimierung zuerst ausgeführt?

2. Führt die DCT-Transformation bereits zu einer Datenreduktion?

3. Warum erhält man bei einer sehr starken Bildkomprimierung sogenannte Block-Artefakte?





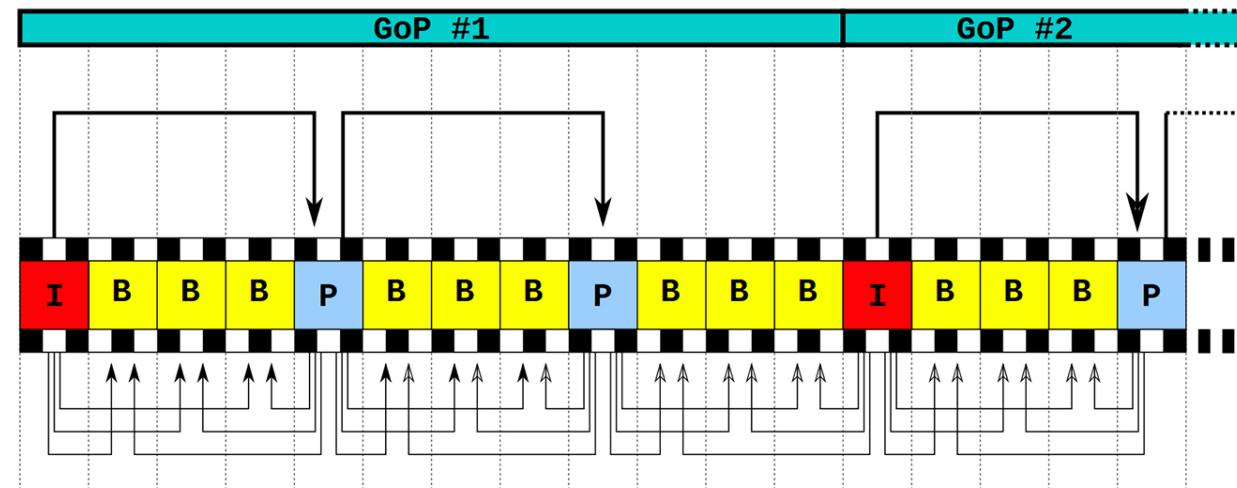
Fragen?

Interframekomprimierung mit GOP

Wenn der Bildinhalt zum Vorgänger nur wenig oder gar nicht ändert, muss nur die **Bilddifferenz** gespeichert werden.



GoP-Sequenz
Group-of-Pictures



Ähnliches Konzept beim Speicher-Backup (Inkrementell / Differenziell)

Tag1: Fullbackup - Tag2 bis Tag7: Inkrementelles Backup - Danach Wiederholung

Bei Datenverlust müssen das Fullbackup und sämtliche inkrementellen Backups bis zum Schadensvorfall neu eingespielt werden.

Und hier die letzten Aufgaben zum Thema:

Beantworten sie nun die folgenden Fragen: *(Dauer: 5 Min.)*

1. Was ist der Unterschied zwischen Intraframe- und Interframe-Komprimierung?

2. Bei welcher 30 Sekunden-Videosequenz bietet die Interframekomprimierung mehr Potential?
"Faultier auf Nahrungssuche" oder "Formel-1-Rennenwagen bei Zieleinfahrt"?

3. Gibt es Parallelen zwischen Datenbackupkonzepten und Interframe-Komprimierung?

4. Was versteht man unter GOP25?



Damit sind sie jetzt Multimediacompetent!



Noch Fragen?