Kapitel 9

Farbe

9.1 Physik

Ein Teil des elektromagnetischen Spektrums wird vom Auge wahrgenommen:

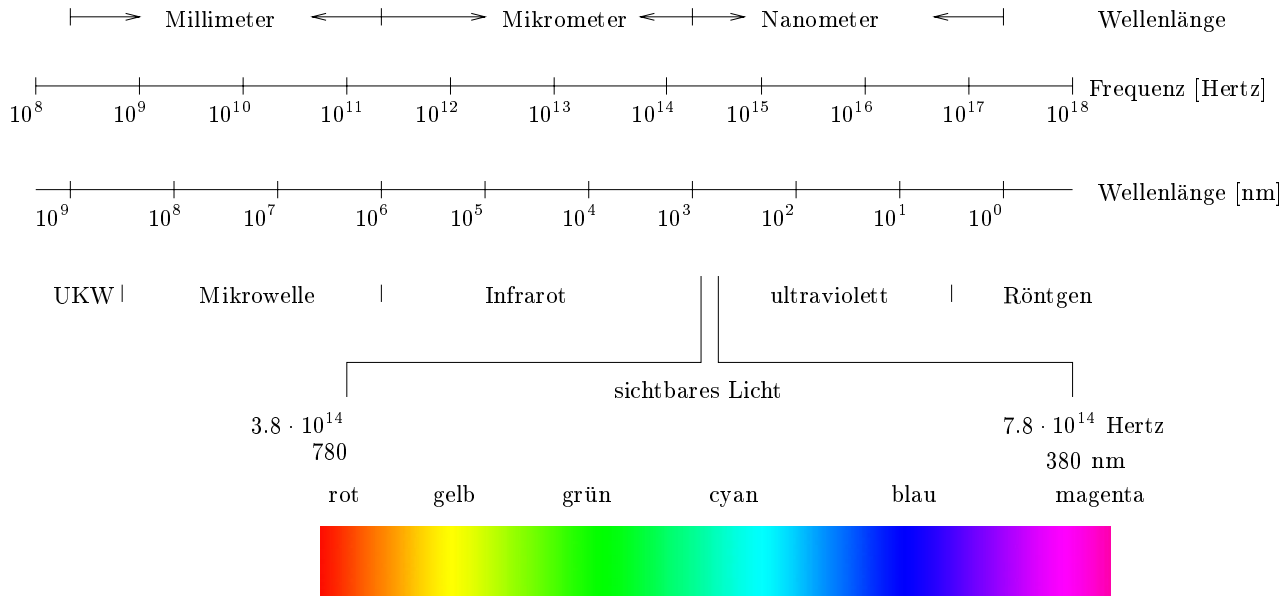


Abbildung 9.1: Elekromagnetisches Spektrum

Es gilt:

- Wellenlänge · Frequenz = Lichtgeschwindigkeit ($= 2.998 \cdot 10^8$ m/s).
- Spektralfarben bestehen aus Licht einer einzigen Wellenlänge.
- In der Natur vorkommende Farben bestehen aus Licht, das aus verschiedenen Wellenlängen zusammengesetzt ist.
- Die Verteilung der Wellenlängen bezeichnet man als Spektrum.

9.2 Dominante Wellenlänge

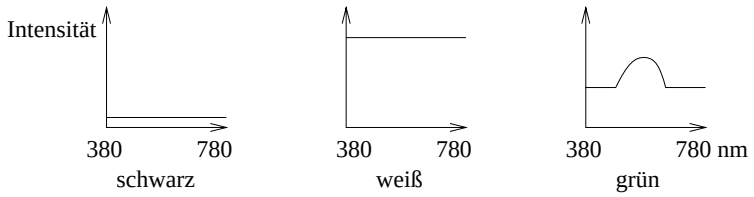


Abbildung 9.2: Spektren zu drei Farben

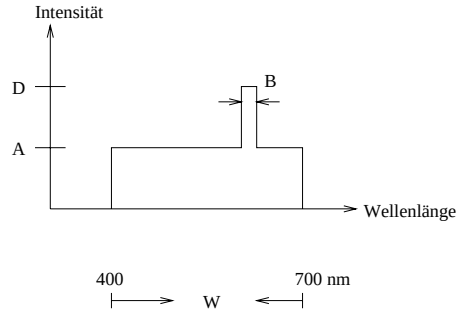


Abbildung 9.3: Dominante Wellenlänge

Eine mögliche Charakterisierung für den Farbeindruck lautet:

hue: Farbton, gegeben durch dominante Wellenlänge.

luminance: Helligkeit $= L = (D \Leftrightarrow A) \cdot B + A \cdot W$

saturation: Sättigung, Reinheit, gegeben durch das Verhältnis der Fläche im "Turm" zur Gesamtfläche $= \frac{(D-A)\cdot B}{L}$.

Je weiter A und D auseinanderliegen, desto größer ist die Sättigung, d.h. desto reiner ist die Farbe. Bei $A = 0$ liegt nur die dominante Wellenlänge vor. Bei $A = D$ liegt weißes Licht vor.

Der Mensch kann etwa 128 reine Farbtöne unterscheiden. Pro Farbton können etwa 20 Sättigungsgrade unterschieden werden.

9.3 Grundfarben

Beobachtung: Durch Mischen (= Addieren) von Farben entstehen neue Farben. Wähle 3 Grundfarben, z.B. Rot (R), Grün (G), Blau (B). Bei einer Normierung $R + G + B = 1$ läßt sich jede Kombination durch Angabe von zwei Parametern (siehe Abbildung 9.4) beschreiben (da z.B. B aus R und G folgt).

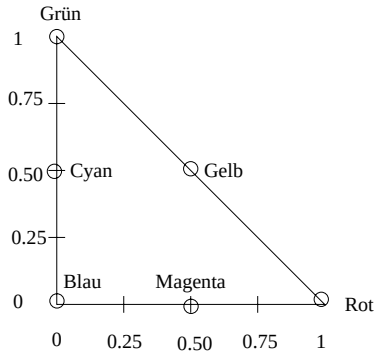


Abbildung 9.4: 2-dimensionale Beschreibung von Farbtönen

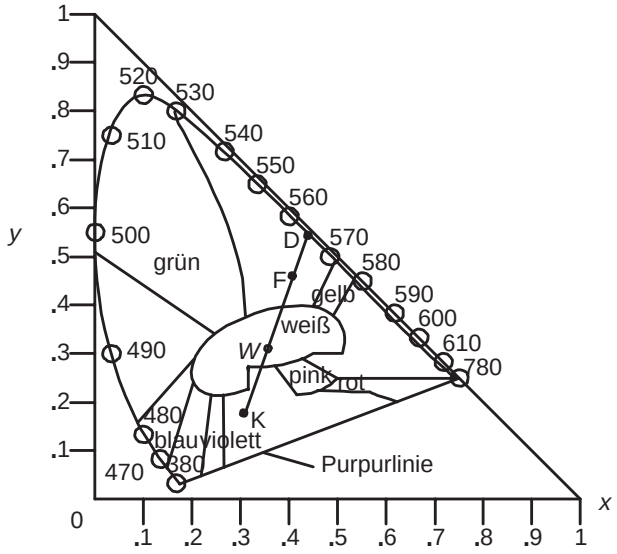


Abbildung 9.5: CIE-Farbdiagramm des sichtbaren Spektrum, Angaben zur Wellenlänge in nm

Abbildung 9.5 zeigt den im Jahre 1931 definierten CIE-Standard (*Commision Internationale l'Éclairage*), in dem drei (künstliche) Grundfarben festgelegt wurden, die alle sichtbaren Farben erzeugen können.

Die Grundfarben eines typischen Farbbildschirms haben die (x, y)-Koordinaten

$$\begin{aligned} \text{Rot} &= (0.628, 0.346), \\ \text{Grün} &= (0.268, 0.588), \\ \text{Blau} &= (0.150, 0.070). \end{aligned}$$

Die Reinheit der Farbe F ist der relative Abstand von F zu W, bezogen auf die Strecke $\overline{WD}$, wobei D den Schnittpunkt der Geraden durch W und F mit der Kurve bildet. D ist die dominante Wellenlänge in F. Das Komplement K der Farbe F ergibt sich durch Spiegelung von F an W mit entsprechender Skalierung.

9.4 RGB-Modell (Rot, Grün, Blau), (additiv)

Zur Ansteuerung der dreifarbigen Phosporschicht mit roten, grünen, blauen Phosphorpunkten bietet sich das RGB-Modell an. Es handelt sich um ein additives Farbmodell, da das von den Phosphorpunkten ausgehende Licht addiert wird.
Typische Darstellung durch Einheitswürfel:

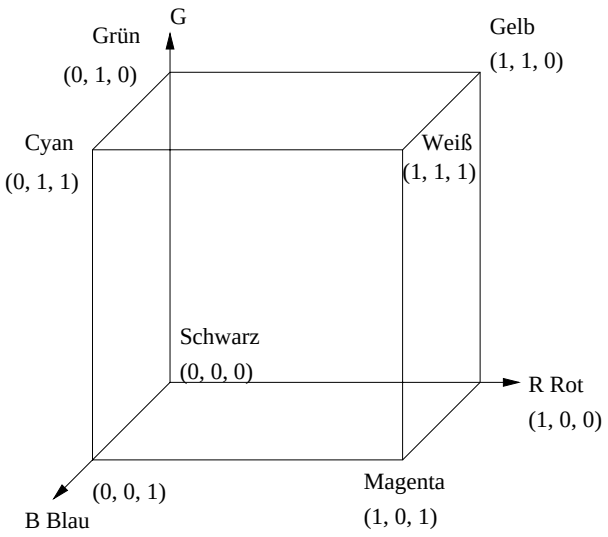


Abbildung 9.6: RGB-Würfel

Häufig werden die drei RGB-Werte statt im reellen Wertebereich $[0, 1]$ in 256 Abstufungen als ganze Zahlen im Bereich $\{0, 1, ..., 255\}$ angegeben, die in einem Byte kodiert werden. Dadurch ergibt sich die Darstellung einer Farbe in drei RGB-Bytes.

9.5 CMY-Modell (Cyan, Magenta, Yellow), (subtraktiv)

Bei Farbdrucken empfängt das Auge nur solche Anteile des weißen Lichts, die reflektiert werden. Es bietet sich daher ein subtraktives Modell an.
Ein CMY-Tripel beschreibt, wieviel von den Grundfarben Cyan, Magenta, Yellow reflektiert bzw. von den Grundfarben Rot, Grün, Blau absorbiert wird.

Es gilt:	$(0, 0, 0)$	absorbiert nichts	bleibt Weiß
	$(0, 0, 1)$	absorbiert Blau	bleibt Gelb
	$(0, 1, 0)$	absorbiert Grün	bleibt Magenta
	$(1, 0, 0)$	absorbiert Rot	bleibt Cyan
	$(0, 1, 1)$	absorbiert Cyan	bleibt Rot
	$(1, 0, 1)$	absorbiert Magenta	bleibt Grün
	$(1, 1, 0)$	absorbiert Gelb	bleibt Blau
	$(1, 1, 1)$	absorbiert alles	bleibt Schwarz

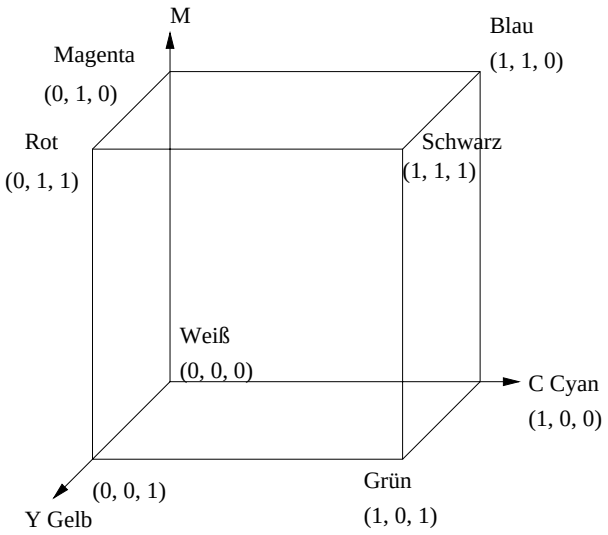


Abbildung 9.7: CMY-Würfel

Beispiel:	$(0,1,0)$ Magenta	gemischt mit	$(0,0,1)$ Gelb	ergibt	$(0,1,1)$ Rot
	$(1,0,0)$ Cyan	gemischt mit	$(0,0,1)$ Gelb	ergibt	$(1,0,1)$ Grün
	$(1,0,0)$ Cyan	gemischt mit	$(0,1,0)$ Magenta	ergibt	$(1,1,0)$ Blau

Die Umrechnung zwischen dem CMY- und dem RGB-Modell erfolgt in Vektorschreibweise über die Subtraktionen

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} S \\ S \\ S \end{pmatrix} \Leftrightarrow \begin{pmatrix} C \\ M \\ Y \end{pmatrix} \qquad \text{und} \qquad \begin{pmatrix} C \\ M \\ Y \end{pmatrix} = \begin{pmatrix} W \\ W \\ W \end{pmatrix} \Leftrightarrow \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

wobei die Vektoren $[S,S,S]$ im CMY-Modell und $[W,W,W]$ im RGB-Modell gleich $[1,1,1]$ sind.

9.6 Das YIQ-Modell

Beim 1953 in den USA eingeführten *NTSC-System* (National Television Standards Committee) werden die Farben durch die Farbparameter *YIQ* beschrieben, wobei Y die Helligkeit darstellt. Die Umrechnung zum RGB-Modell erfolgt durch die Formeln

$$\begin{pmatrix} Y \\ I \\ Q \end{pmatrix} = \begin{pmatrix} .299 & .587 & .114 \\ .596 & \Leftrightarrow.274 & \Leftrightarrow.322 \\ .211 & \Leftrightarrow.522 & .311 \end{pmatrix} \cdot \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1 & 0.956 & 0.623 \\ 1 & \Leftrightarrow 0.272 & \Leftrightarrow 0.648 \\ 1 & \Leftrightarrow 1.105 & 1.705 \end{pmatrix} \cdot \begin{pmatrix} Y \\ I \\ Q \end{pmatrix}$$

Beim europäischen *PAL-System* (Phase Alternating Line) werden statt der Parameter I und Q die Farbdifferenzen $R \Leftrightarrow Y$ und $B \Leftrightarrow Y$ übertragen.

Die Konvertierung der Farbinformation in Monochrom-Darstellung erfolgt in beiden Systemen durch Auswertung des Helligkeitsparameters Y.

9.7 YUV-Modell

Ein Farbwert wird beschrieben durch ein YUV-Tripel, wobei Y die Helligkeit (Luminanz) bezeichnet und U, V Farbdifferenzen (Chrominanz). Die Helligkeitsempfindung resultiert zu 59 % aus dem Grünanteil, zu 30 % aus dem Rotanteil und zu 11 % aus dem Blauanteil:

$$Y = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

In den Farbdifferenzen (mit historisch begründeten Normierungsfaktoren) ist die restliche Information codiert:

$$\begin{aligned} U &= 0.493 \cdot (B \Leftrightarrow Y) \\ V &= 0.877 \cdot (R \Leftrightarrow Y) \end{aligned}$$

Der Vorteil dieses Farbmodells liegt darin begründet, daß in der Y-Komponente das Bild als Matrix von Grauwerten vorliegt und ggf. separat von der Farbinformation weiterverarbeitet werden kann.

9.8 HSV-Modell

Zur intuitiven Beschreibung einer Farbe eignet sich das HSV-Modell, welches jede Farbe durch ein Tripel

- Hue = Farbton
- Saturation = Sättigung
- Value = Helligkeit

beschreibt.

Projiziere den RGB-Würfel längs der Weiß-Schwarz-Diagonale (Abbildung 9.8).

Dieses Sechseck bildet die Basis einer Pyramide. Die Wahl des Farbtons (hue) geschieht durch Angabe eines Winkels (0° = Rot).

Der Parameter V liegt zwischen 0 und 1 und bestimmt die Intensität der Farbe (dargestellt durch die Senkrechte). Der Parameter S liegt zwischen 0 und 1 und bestimmt die Reinheit der Farbe (Entfernung von der Senkrechten). Die Farbselektion kann erfolgen, indem z.B. zunächst eine reine Farbe ausgewählt wird ($H = \alpha, V = 1, S = 1$). Das Hinzumischen von Weiß zur Erzeugung von Pastellfarben erfolgt durch Reduktion von S bei konstantem V. Das Hinzumischen von Schwarz (zur Erzeugung von dunklen Farben) erfolgt durch Reduktion von V bei konstantem S.

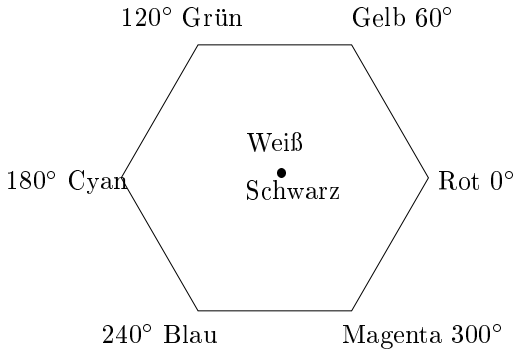


Abbildung 9.8: Gradeinteilung für Farbtöne im HSV-Modell

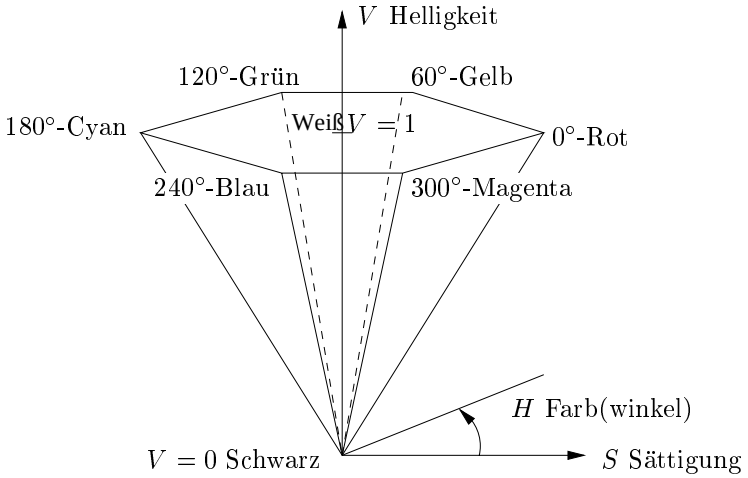


Abbildung 9.9: HSV-Modell

Umrechnung von RGB nach HSV

Die Achse V entspricht der Diagonalen im RGB-Würfel durch die Punkte Schwarz und Weiß, deshalb ist der Wert für V gleich dem Maximum der RGB-Intensitäten. Die Werte H und S können aus der Position des Punktes in jenem Sechseck berechnet werden, das durch Projektion des kleinsten, den RGB-Punkt beinhaltenden Würfels erzeugt wird.

Beispiel: Welche HSV-Darstellung haben die RGB-Bytes (64, 128, 32)?
Im RGB-Einheitswürfel entspricht dies

$$\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{8}\right) .$$

$$\begin{array}{lcccccl} v & = & \max(r,g,b) & = & \frac{1}{2} & = & 50\ \% \\ mi & := & \min(r,g,b) & = & \frac{1}{8} & & \\ s & = & \dfrac{v \Leftrightarrow mi}{v} & = & \frac{\frac{3}{8}}{\frac{1}{2}} = \frac{3}{4} & = & 75\ \% \end{array}$$

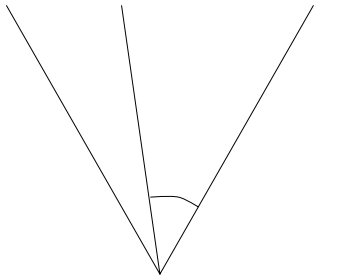
Die dominante Grundfarbe ist Grün, da $v = g$.
Am schwächsten ist Blau vertreten, da $mi = b$.
$\Rightarrow$ Farbe im Bereich Gelb ... Grün
$\Rightarrow h = 60° \ldots 120°$.

$$h = \left(1 + \frac{v \Leftrightarrow r}{v \Leftrightarrow mi}\right) \cdot 60° = \left(1 + \frac{\frac{1}{4}}{\frac{3}{8}}\right) \cdot 60° = \left(1 + \frac{2}{3}\right) \cdot 60° = 100°$$

Umrechnung von HSV nach RGB

Beispiel: Wie lauten die RGB-Bytes (Werte zwischen 0 und 255) für den Farbton 100° bei 75 % Sättigung und 50 % Helligkeit?

Farbton =	Grün	?	Gelb
$h =$	120°	100°	60°
$RGB =$	(0, 1, 0)	?	(1, 1, 0)



$$\begin{array}{lcccccc} f & = & \text{Winkel}/60 & - & \text{Winkel div } 60 & & \\ & = & \frac{5}{3} & - & 1 & = & \frac{2}{3} \end{array}$$

		R	G	B		R	G	B	
Farbton h	:	$1 \Leftrightarrow f$	1	0	$\Rightarrow$	$\frac{1}{3}$	1	0	für $f = \frac{2}{3}$
Sättigung s	:	$1 \Leftrightarrow s \cdot f$	1	$1 \Leftrightarrow s$	$\Rightarrow$	$1 \Leftrightarrow \frac{3}{4} \cdot \frac{2}{3} = \frac{1}{2}$	1	$\frac{1}{4}$	für $s = \frac{3}{4}$
Helligkeit v	:	$v \cdot (1 \Leftrightarrow s \cdot f)$	v	$v \cdot (1 \Leftrightarrow s)$	$\Rightarrow$	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{8}$	für $v = \frac{1}{2}$
Lösung:						64	128	32	

9.9 HLS-Modell

Das HLS-Modell (hue, lightness, saturation) ist dem HSV-Modell ähnlich. Die Farbe (hue) wird ebenfalls als Winkel in einem Farbenkreis angegeben. Die Reihenfolge der Farben auf diesem Kreis ist zwar identisch der beim HSV-Modell, jedoch liegt beim HLS-Modell die Farbe Blau bei 0°, d.h. um 120° verdreht.

Auch der Parameter S(aturation) entspricht dem HSV-Modell und liegt ebenfalls im Bereich $[0, 1]$. Der Wert $S = 1$ entspricht maximaler Reinheit, und für $S = 0$ erhält man die Grauskala.

Der Parameter L(ightness) entspricht in etwa dem Parameter V(alue), wobei allerdings aus der Pyramide durch "Hochziehen" des Punktes $S = 0$, $V = 1$ ein Doppelkegel entsteht. Die Werte $L = 0$ und $L = 1$ beschreiben die Farben Schwarz und Weiß, und die Farben maximaler Sättigung liegen auf der Ebene durch $L = 0.5$.

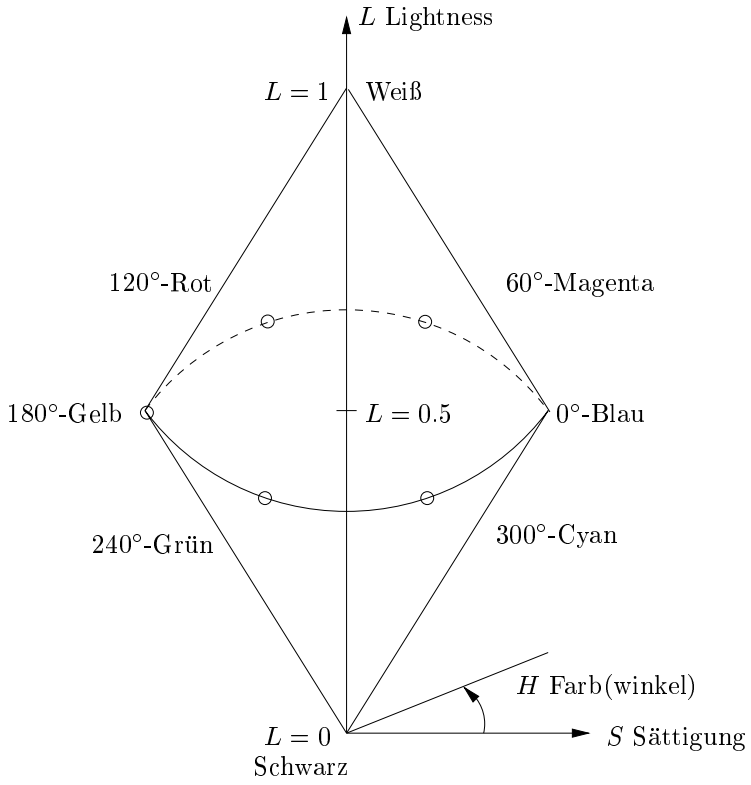


Abbildung 9.10: HLS-Modell

Bemerkung

Die lineare Interpolation zwischen zwei Farben geschieht beim RGB-, CMY-, YIQ- und YUV-Modell längs der Verbindungsstrecke zwischen diesen Farben im dreidimensionalen Farbraum. Im allgemeinen ist dies aber beim HSV- und HLS-Modell nicht der Fall.

9.10 Color Data Base

In einer Datenbank sind Byte-Tripel abgelegt zu einer Auswahl von Farbbeschreibungen, z.B.

```
205   92   92  indian red
124  252    0  lawn green
 25   25  112  midnight blue
210  105   30  chocolate
```

9.11 CNS

Zur verbalen Beschreibung einer Farbe eignet sich das CNS-Modell (*Color Name System*).

Zur Beschreibung des Farbtons verwende *red*, *orange*, *yellow*, *green*, *blue*, *purple*. Zur Beschreibung der Sättigung verwende *grayish*, *moderate*, *strong*, *vivid*. Zur Beschreibung der Helligkeit verwende *very dark*, *dark*, *medium*, *light*, *very light*.

Die achromatische Skala besteht aus den sieben Grautönen *black*, *very dark gray*, *dark gray*, *gray*, *light gray*, *very light gray*, *white*.

9.12 Color Table (Farbtabelle)

In einem Digitalrechner erwartet der Video-Controller die Farbinformation als Byte-Tripel. Aus Platzgründen können nicht immer für jedes Bildschirmpixel 3 Bytes gespeichert werden. Statt dessen speichert man pro Pixel einen Index, bestehend aus p Bits, mit dem man ein *RGB*-Byte-Tripel aus der Color Table referieren kann.

Beispiel: $p = 2$, d.h. $2^2 = 4$ Farben gleichzeitig darstellbar.

Index	R	G	B	
0	255	0	0	Rot
1	0	255	0	Grün
2	0	0	255	Blau
3	255	255	0	Gelb

Die Color Table kann verwendet werden, um mehrere Bilder dynamisch zu überlagern und wieder zu trennen, ohne die Bilder neu zu zeichnen.

Beispiel: In ein farbiges Bild mit den 8 Farben Schwarz, Grau, Rot, Gelb, Grün, Cyan, Blau, Magenta soll eine Grafik, bestehend aus weißen Linien, ein- und ausgeblendet werden. Pro Pixel seien 4 Bits als Index in der Color Table vorgesehen: Das vorderste Bit beschreibt die weiße Grafik, die hinteren 3 Bits beschreiben die Farbe.

Index	R	G	B	
0000	255	255	255	
0001	255	255	255	
0010	.	.	.	
0011	.	.	.	
0100	.	.	.	
0101	.	.	.	
0110	.	.	.	
0111	255	255	255	
1000	0	0	0	Schwarz
1001	150	150	150	Grau
1010	255	0	0	Rot
1011	255	255	0	Gelb
1100	0	255	0	Grün
1101	0	255	255	Cyan
1110	0	0	255	Blau
1111	255	0	255	Magenta

Offenbar werden alle Pixel mit vorderstem Bit= 0 weiß gezeichnet, alle Pixel mit vorderstem Bit= 1 erhalten die zugewiesene Farbe. Kopiert man nun die untere Hälfte der Color Table in die obere Hälfte, so verschwindet die weiße Grafik. Setzt man die obere Hälfte der Color Table auf Weiß, erscheint die Grafik wieder. Der Aufwand hängt nur von der Größe der Farbtabelle ab, nicht von der Größe der Bilder.
Die Color Table kann auch zur Animation verwendet werden.

Beispiel: Ein (beliebig komplexes) Objekt der Farbe Grün soll sich zyklisch über 15 disjunkte Positionen bewegen. Beschreibe die Szene wie folgt:
Alle Pixel des Objekts in Lage i erhalten den Index i.
Alle Pixel des Hintergrunds erhalten den Index 15.
Die Color Table ist für alle Einträge mit der Hintergrundfarbe geladen. Das Objekt taucht bei Position i auf durch `color_table[i] :=` Objektfarbe. Es verschwindet von Position i durch `color_table[i] :=` Hintergrundfarbe.

Bei sich überlagernden Objekten wird für jede Position eine Bit-Plane benötigt: Alle Pixel des Objekts in Lage i setzen das i-te Bit in ihrem Index auf 1. Das Objekt wird bei Position i sichtbar, indem für Indizes mit i-tem Bit $= 1$ die Objektfarbe gesetzt wird.

Die Anzahl der Bit-Planes entspricht der Anzahl der Bits im Index.

Darstellung eines True-Color-Bildes auf einem 8-Bit-Farbschirm

Es wird eine Farbtabelle initialisiert, die einem $6 \times 6 \times 6$ RGB-Würfel entspricht. D.h. auf insgesamt 216 Einträge verteilt befindet sich das Farbspektrum mit je 6 verschiedenen Rot-, Grün- und Blau-Abstufungen. Jedem RGB-Tripel des True-Color-Bildes wird der Index des nächstgelegenen Farbeintrags zugeordnet. Hierzu wird der Bereich $0 \ldots 255$ in 6 Intervalle partitioniert, denen ein quantisierter Wert zugeordnet ist:

x	0 ... 25	26 ... 76	77... 127	128 ... 178	179 ... 229	230 ... 255
$q(x)$	0	51	102	153	204	255

$$q(x) := \left\lfloor \frac{x+25}{51} \right\rfloor \cdot 51$$

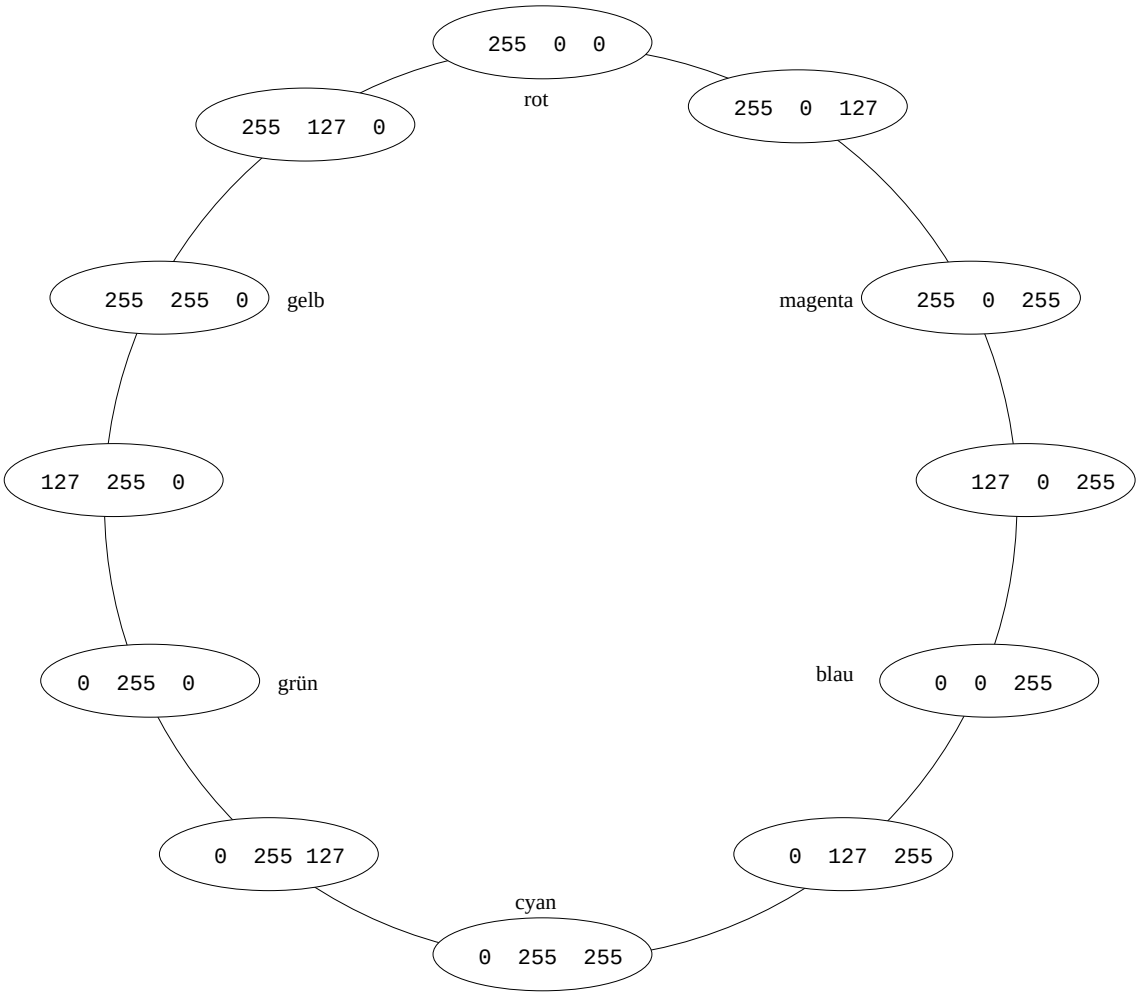


Abbildung 9.11: 12 typische Eintrage einer Farbtabelle fur einen Bildschirm mit 4 Bit Farbtiefe

Hi-Color-Modus

Beim Hi-Color-Modus werden die Rot-, Grün-, Blauanteile eines Pixels durch den Grafikkartentreiber auf $5+6+5 = 16$ Bit gerundet, d.h., die letzten 3 bzw. 2 Bit werden abgeschnitten. Der resultierende 2 Byte lange Index wird im Bildspeicher abgelegt. Beim Auslesen läßt sich daraus unmittelbar ein Spannungstripel erzeugen, eine Farbtabelle wird nicht benötigt. Es sind also $2^{16} = 65536$ Farben gleichzeitig darstellbar.

Beim Einlesen eines Palettenbildes wird jedem Bildpunkt der 2-Byte Index zugeordnet, der sich nach Rundung des referierten Farbtabelleneintrags ergibt. Danach wird die Farbtabelle nicht mehr benötigt.

9.13 Beispiel-Applet zu Farbe

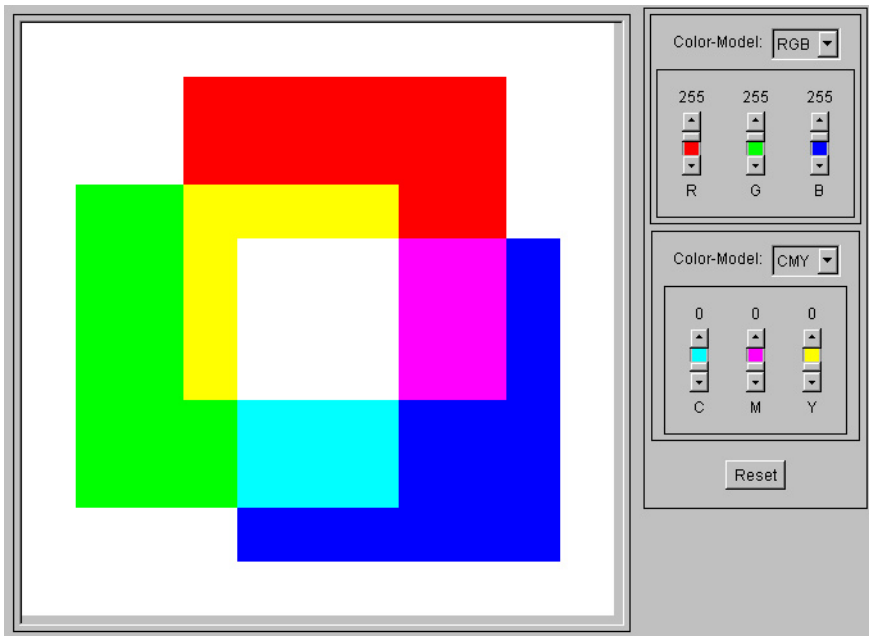


Abbildung 9.12: Screenshot vom Farben-Applet

Kapitel 10

Pixeldateiformate

10.1 TIF

Eines der häufig verwendeten Dateiformate zur Speicherung von Pixelbildern wurde von Aldus Corporation, Seattle, USA, entwickelt: *Tag Image File Format*, abgekürzt TIF.

Dateikopf:	1. Wort	`4d4d` Motorola (high order first) `4949` Intel (low order first)
	2. Wort	`002a` Versionsnummer (konstant)
	3. + 4. Wort	Offset des ersten IFD (Image File Directory)
IFD:	1. Byte	Anzahl der Tags im IFD, jedes Tag besteht aus 12 Bytes
	Aufbau eines Tag:	
	1. Wort	Identifikation des Tags (es gibt 45 Tags)

	2. Wort	Typ der Daten
		`1`: Byte
		`2`: 0-terminierter String von ASCII-Zeichen
		`3`: short = 16-Bit unsigned integer `(0000-ffff)`
		`4`: long = 32-Bit unsigned integer `(00000000-ffffffff)`
		`5`: Rational = Bruch aus 2 long-Werten
		`11`: float = im IEEE-Format, einfache Genauigkeit
		`12`: double = im IEEE-Format, doppelte Genauigkeit

	3. + 4. Wort	Anzahl der Daten für dieses Tag
	5. + 6. Wort	Tag-Daten falls mit $\leq$ 4 Bytes darstellbar : von links nach rechts füllen falls mehr als 4 Bytes benötigt: 32-Bit Startadresse für Daten
Datenblock:		RGB-Werte oder Indizes (ggf. komprimiert) für die Farbtabelle

Offset	Bedeutung	Wert
0100	ImageWidth	z.B. 814
0101	ImageLength	z.B. 517
0102	Bits per Sample	z.B. 8, ggf. Zeiger auf 3 shorts
0103	Compression	1 keine Komprimierung 2 CCITT Gruppe 3 (nur bei Schwarz/Weiß) 3 CCITT Gruppe T4 (nur bei Schwarz-Weiß) 4 CCITT Gruppe T6 (nur bei Schwarz/Weiß) 5 LZW 6 JPEG 32773 Packbit mit Lauflängen-Kodierung
0106	Photometric Interpretation	0 S/W mit 0=Weiß, wachsende Nummern gegen Schwarz 1 S/W mit 0=Schwarz, wachsende Nummern gegen Weiß 2 RGB-Farbbild, pro Pixel drei Farbwerte, 0,0,0=Schwarz 3 Palettenbild, pro Pixel ein Index in Palette 4 Transparenzmaske für ein weiteres Bild 5 CMYK-Farbsystem, pro Pixel 4 Farbwerte 6 YUV-Farbsystem, pro Pixel Luminanz + 2 Chrominanz
0111	Strip Offset	Startadresse des Datenblocks (meistens $0008)
0112	Orientation	1 horizontal, beginnend oben links
0115	Samples per Pixel	3 bei RGB-Bildern 1 sonst
0116	Rows per Strip	wenn 1 Block: Bildhöhe wenn > 1 Block: Anzahl Zeilen pro Block
0117	Strip Byte Counts	wenn 1 Block: Anzahl der Bytes im Block wenn > 1 Block: Zeiger auf Feld mit Streifenlängen
011a	X Resolution	Zeiger auf 2 long Werte, zB. 300 1
011b	Y Resolution	Zeiger auf 2 long Werte, zB. 300 1
0128	Resolution unit	1 keine Einheit 2 Inch 3 Zentimeter
011c	Planar Configuration	1 RGBRGBRGB ... 2 RRRR... GGGG.... BBBB....
0140	Color map	Anzahl der Einträge + Startadresse Jeder Farbwert $0 \leq w = 255$ wird als Wort ww abgelegt

Tag-Art : True Color Bild

						Position 0:
4d4d						Motorola
002a						Versionsnr. (konstant)
0013	43ba					Offset des IFD $001343ba = 1262522
						Position 8:
ebe9	e7e2	e4c0	b3bd	9a9c	a387	Beginn der RGB-Werte, insgesamt 814*517*3 Bytes
9178	5294	724b	8d6f	4b8e	7453	
a79e	6da8	a06f	a19b	759d	a486	
a6ac	8aa2	aa92	a7ac	9ba5	a995	
...						
...						
...						
						Position 1262522: Beginn des Image File Directory
000b						$000b = 11 Tags zu je 12 Bytes
0100	0003	0000	0001	032e	0000	Breite, 1 short, Wert 3*256 + 2*16 + 14 = 814
0101	0003	0000	0001	0205	0000	Höhe, 1 short, Wert 2*256 + 0*16 + 5 = 517
0102	0003	0000	0003	0013	4444	Bits per sample, 3 short, Adresse $134444=1262660
0103	0003	0000	0001	0001	0000	Compression, 1 short, Wert=1 (keine Kompression)
0106	0003	0000	0001	0002	0000	Photometric, 1 short, Wert=2 (RGB-Farbbild)
0111	0004	0000	0001	0000	0008	StripOffsets,1 long, Wert=8 (Startadresse)
0112	0003	0000	0001	0001	0000	Orientierung,1 short, Wert = 1 (zeilenweise links/oben)
0115	0003	0000	0001	0003	0000	Samples per Pixel, 1 short, Wert=3 (RGB)
0116	0004	0000	0001	0000	0205	Rows per Strip, 1 long, Wert $205 = 517 (Zeilen)
0117	0004	0000	0001	0013	43b2	StripByteCounts, 1 long, Wert = 1262514 (Bytes)
011c	0003	0000	0001	0001	0000	Planar Konfiguration, 1 short, Wert=1 (RGBRGB...)
0000	0000					Ende IFD, kein weiteres IFD
						Position 1262660:
0008	0008	0008				jeweils 8 Bit pro Farbwert

kommentierter Hex-Dump zu einem True-Color-Bild im tif-Format

Vorspann	8 Bytes
814 * 517 RGB-Tripel	1262514 Bytes
IFD 11 * 12 + 12	144 Bytes
→ Dateilänge	1262666 Bytes

Tag-Art : Palettenbild

						Position 0:
4d4d						Motorola
002a						Versionsnr. (konstant)
0006	6be					Offset des IFD $00066bee = 420846
						Position 8:
fe68	bd52	6f6f	6f6f	0101	01f4	Beginn der Indizes, insgesamt 814*517 Bytes
5252	e952	2abd	522a	5252	b252	
0000	016f	076f	01bd	e952	3877	
ea3a	3838	38a1	0101	3a38	01be	
.						
.						
						Position 420846: Beginn des Image File Directory
000c						$000c = 12 Tags zu je 12 Bytes
0100	0003	0000	0001	032e	0000	Breite, 1 short, Wert $32e = 814
0101	0003	0000	0001	0205	0000	Höhe, 1 short, Wert $205 = 517
0102	0003	0000	0001	0008	0000	Bits per sample, 1 short, Wert = 8 (Adresse)
0103	0003	0000	0001	0001	0000	Compression, 1 short, Wert=1 (keine Kompression)
0106	0003	0000	0001	0003	0000	Photometric, 1 short, Wert=3 (Palette)
0111	0004	0000	0001	0000	0008	StripOffsets,1 long, Wert=8 (Startadresse)
0112	0003	0000	0001	0001	0000	Orientierung,1 short, Wert=1 (zeilenweise l.o.)
0115	0003	0000	0001	0001	0000	Samples per Pixel, 1 short, Wert=1 (Palette)
0116	0004	0000	0001	0000	0205	Rows per Strip, 1 long, Wert $205 = 517 (Zeilen)
0117	0004	0000	0001	0006	6be6	StripByteCounts,1 long, Wert = 420838 (Bytes)
011c	0003	0000	0001	0001	0000	Planar Konfiguration, 1 short, Wert=1 (RGBRGB..)
0140	0003	0000	0300	0006	6c84	Colormap, $300 = 768 short, beginnend bei $66c84
0000	0000					Ende IFD, kein weiteres IFD
						Position: $66c84 = 420996
						Palette mit 768 Doppelbytes
b8b8	a8a8	1010	3c3c	0808	2424	
b4b4	a8a8	1c1c	9898	6868	3838	
3c3c	5858	2c2c	7070	6464	8080	
.						
.						
3030	1414	1010	3030	2020	3c3c	
7070	2828	d8d8	1010	2828	2020	
7474	1414	acac	a8a8	e4e4	e4e4	

kommentierter Hex-Dump zu einem Palettenbild im tif-Format

Vorspann	8 Bytes
814 * 517 Indizes	420838 Bytes
IFD 12 * 12 + 6	150 Bytes
Palette 3 * 256 * 2	1536 Bytes
→ Dateilänge	422532 Bytes

10.2 PBM, PGM, PNM und PPM

Die *Portable Bitmap Utilities* (PBM, auch pbmplus oder netpbm) sind eine Sammlung freierhältlicher Programme, die von Jef Poskanzer gepflegt werden. Es handelt sich eigentlich um drei Programmpakete, die sich mit unterschiedlichen Bildarten und File-Formaten befassen:

- Die Portable Bitmap Utilities (PBM) manipulieren monochrome Bilder.
- Die Portable Greymap Utilities (PGM) manipulieren Grauwert-Bilder.
- Die Portable Pixmap Utilities (PPM) manipulieren Farbbilder.

Die Portable Anymap Utilities (PNM) arbeiten auf allen von den drei Programmpaketen erzeugten Bilddateien. Für PNM gibt es kein eigenes Dateiformat.

Die PBM-, PGM- und PPM-File-Formate sind so einfach wie möglich gehalten. Sie starten jeweils mit einem Header, dem die Bildinformation unmittelbar folgt. Der Header ist immer in ASCII geschrieben, wobei die einzelnen Einträge durch White Spaces getrennt werden. Die Bildinformation kann entweder im ASCII- oder Binärformat sein.

Es gibt jeweils zwei Header-Versionen, von denen eine für ASCII- und eine für binäre Bildinformationen benutzt wird.

PBM-Header

Ein PBM-Header besteht aus folgenden Einträgen, jeweils durch White Spaces getrennt:

Magic Value	`P1` für ASCII-, `P4` für binäre Bildinformation
Image Width	Breite des Bildes in Pixeln (ASCII-Dezimalwert)
Image Height	Höhe des Bildes in Pixeln (ASCII-Dezimalwert)

PGM-Header

Ein PGM-Header besteht aus folgenden Einträgen, jeweils durch White Spaces getrennt:

Magic Value	`P2` für ASCII-, `P5` für binäre Bildinformation
Image Width	Breite des Bildes in Pixeln (ASCII-Dezimalwert)
Image Height	Höhe des Bildes in Pixeln (ASCII-Dezimalwert)
Max Grey	Maximaler Grauwert (ASCII-Dezimalwert)

PPM-Header

Ein PPM-Header besteht aus folgenden Einträgen, jeweils durch White Spaces getrennt:

Magic Value	`P3` für ASCII-, `P6` für binäre Bildinformation
Image Width	Breite des Bildes in Pixeln (ASCII-Dezimalwert)
Image Height	Höhe des Bildes in Pixeln (ASCII-Dezimalwert)
Max Color	Maximaler Farbwert (ASCII-Dezimalwert)

Bildinformation in ASCII

Nach dem Header folgt die Beschreibung der Breite $\times$ Höhe vielen Pixel, beginnend in der linken oberen Bildecke zeilenweise von links nach rechts.

Bei PPM besteht jedes Pixel aus drei ASCII-Dezimalwerten zwischen 0 und dem angegebenen maximalen Farbwert, die die Rot-, Grün- und Blauwerte des Pixels darstellen.

Bei PBM und PGM gibt es nur einen ASCII-Dezimalwert pro Pixel. Bei PBM ist der maximale Wert implizit 1; die Werte müssen nicht durch White Spaces getrennt werden.

Beispiel:

Das folgende Beispiel stellt ein 4×4-Pixmap im ASCII-Format dar:

```
P3
# feep.ppm
4 4
15
 0  0  0    0  0  0    0  0  0   15  0 15
 0  0  0    0 15  7    0  0  0    0  0  0
 0  0  0    0  0  0    0 15  7    0  0  0
15  0 15    0  0  0    0  0  0    0  0  0
```

Wie zu sehen, können auch Kommentare in ein PNM-File eingefügt werden. Zeichen ab einem `#` bis zum Zeilenende werden ignoriert.

Binäre Bildinformation

Deutlich kleinere Dateien, die schneller gelesen und geschrieben werden können, werden mit Hilfe der binären Bilddarstellung erreicht.

Die Pixelwerte werden dabei wie folgt abgelegt:

Format	Magic Value	binäre Darstellung
PBM	`P4`	acht Pixel pro Byte
PGM	`P5`	ein Pixel pro Byte
PPM	`P6`	drei Bytes pro Pixel

Beim binären Format ist nach dem Header nur noch ein einziger White Space (typischerweise Newline) erlaubt. Innerhalb der binären Bilddaten sind keine White Spaces erlaubt. Die Bitorder innerhalb der Bytes ist "most significant bit first".

Beachtenswert ist, daß für das Binärformat nur maximale Werte bis 255 benutzt werden können.

Konvertierungsroutinen

Für die Umwandlung vom bzw. in das PNM-Format steht eine ganze Reihe von Routinen zur Verfügung, z.B.:

`anytopnm`, `asciitopgm`, `bmptoppm`, `giftopnm`, `pbmtopgm`, `pgmtopbm`, `pgmtoppm`, `ppmtopgm`, `pstopnm`, `rgb3toppm`, `tifftopnm`, `xbmtopbm`, `xwdtopnm`, `pbmtoascii`, `pbmtolps`, `pbmtopgm`, `pbmtoxbm`, `pgmtopbm`, `pgmtoppm`, `pnmtops`, `pnmtotiff`, `pnmtoxwd`, `ppmtobmp`, `ppmtogif`, `ppmtopgm`, `ppmtorgb3`.

10.3 Photo-CD

Die von Kodak entwickelte Photo-CD weist eine Verzeichnisstruktur auf, die zum Ansteuern eines Photo-CD-Players ausgelegt ist und außerdem von diversen Bildverarbeitungsprogrammen gelesen werden kann. Hierfür stellt Kodak eine Programmierbibliothek zur Verfügung; das Erstellen einer Photo-CD ist nur speziellen Labors vorbehalten. Typischer Preis: 16,00 DM für CD, 5,00 DM für Session, 1,20 DM pro Bild. Es können 100 Photos, ggf. in mehreren Sessions, aufgebracht werden. Da sich dadurch das Verzeichnis an verschiedenen Stellen der CD befinden kann, muß das CD-Laufwerk "multisessionfähig" sein.

Jedes Photo wird in 5 verschiedenen Auflösungen in einem (patentrechtlich geschützten) Image-Pac abgelegt.

Bezeichnung	Auflösung	unkomprimierter Platzbedarf
Base/16	192×128	72 KBytes
Base/4	384×256	288 KBytes
Base	768×512	1152 KBytes
Base$*$4	1536×1024	4608 KBytes
Base$*$16	3072×2048	18432 KBytes

Beim Einscannen werden die RGB-Werte für jeden Bildpunkt mit 12 Bit Genauigkeit pro Farbanteil abgetastet und anschließend in das Kodak-eigene YCC-Format konvertiert (8 Bit für Helligkeit, je 8 Bit für zwei Chrominanzwerte). Durch 4:1:1 Subsampling wird die Chrominanz über je 4 Pixel gemittelt. Pro Image-Pac werden die ersten 3 Auflösungen explizit gespeichert, für Base$*$4 und Base$*$16 nur die Differenzen. Dadurch reichen etwa 4.5 MByte aus, also etwa 20 % des aufsummierten Platzbedarfs.

10.4 Auflösung

Für die Qualität eines Bildes sind die am Erstellungsprozeß beteiligten Auflösungen verantwortlich. Die Auflösung wird meistens gemessen als *Dots per Inch* (dpi).

Scanner-Auflösung:	Gegeben durch Anzahl der CCD-Elemente in einer Sensorzeile (z.B. 300 dpi).
Scan-Auflösung:	Gewählte Auflösung beim Scanner.
Bildauflösung:	Entspricht zunächst der Scan-Auflösung, kann später ggf. runtergerechnet werden (durch Mittelung) oder hochgerechnet (durch Interpolation).
Bildschirmauflösung:	Ein 17-Zoll Monitor mit Seiten-Verhältnis 4:3 hat eine Breite von $(17 \cdot 4)/5 = 13.6$ inch. Bei 1024×768 Pixeln ergibt das $1024/13.6 \approx 75$ dpi.
Druckerauflösung:	Gegeben durch die Anzahl der Druckerpunkte, die der Druckkopf nebeneinander setzen kann (z.B. 300 dpi). Da bei Tintenstrahldruckern nur 3 Grundfarben (ggf. zusätzlich Schwarz) zur Verfügung stehen, wird jedes Farbpixel durch eine Rasterzelle mit 16×16 Druckerpunkten dargestellt.
Druckauflösung:	Druckerauflösung/Rasterzellengröße, d.h. ein 400 dpi-Belichter mit 16×16 Rasterzellen kann nur $400/16 = 25$ dpi erzeugen.

Thermosublimationsdrucker lösen durch Heizelemente Farbpigmente aus einer Folie, die anschließend in ein Spezialpapier eindringen. Jedes True-Color-Pixel wird daher als ein Farbpunkt gedruckt (*Continuous Tone-Druck*). Ein 300 dpi Thermosublimationsdrucker ermöglicht daher eine 300 dpi Druckauflösung. Ein 100 ASA-Kleinbilddia, welches eine reale Auflösung von 2000 dpi in sich birgt, kann daher mit einem 300 dpi-Thermosublimationsdrucker in idealer Weise auf das $2000/300 = 6.66$-fache vergrößert werden, welches eine Kantenlänge von $6.66 * 36$ mm $= 24$ cm bedeutet (etwa DIN-A4). Allerdings ist der im analogen Dia enthaltene Helligkeitsumfang etwa um den Faktor 10 größer als ein Farbdruck oder Monitorbild darstellen kann.

Auflösung KB-Dia

Die Projektion eines 24×36 mm Kleinbilddias auf eine 1.80 m breite Leinwand stellt eine 50-fache Vergrößerung dar. Sind dann auf 1 cm Leinwand 8 Linien zu unterscheiden, entspricht das 16 dots per cm Leinwand= 800 dots per cm Dia ≈ 2000 dpi.

24 x 36 mm Dia, eingescannt mit 2168 dpi, ergibt 3072 x 2048 Pixel,
skaliert auf 75 %, ergibt 2304 x 1536 Pixel
gedruckt mit 300 dpi Ink Jet, ergibt 19.5 x 13.0 cm.

10.5 LZW-Komprimierung (Lempel/Ziv/Welch, 1984)

Ein wichtiger Bestandteil des GIF-Formates (siehe nächste Sektion) ist die LZW-Komprimierung für Zeichenketten.

Starte mit Stringtabelle, gefüllt mit `characters`, und fülle sie mit Verweisen auf bereits gelesene Substrings.

```
x := get_char();
w := x;
repeat
   x := get_char();
   if wx in Tabelle
      then w := wx
      else put_string(code(w));
           trage wx in Tabelle ein
           w := x
   endif
until x = EOF
```

w	Input	Output	String	Code
			a	1
			b	2
			c	3
	a			
a	b	1	ab	4
b	a	2	ba	5
a	b			
ab	c	4	abc	6
c	b	3	cb	7
b	a			
ba	b	5	bab	8
b	a			
ba	b			
bab	a	8	baba	9
a	a	1	aa	10
a	a			
aa	a	10	aaa	11
a	a			
aa	a			
aaa	a	11	aaaa	12

Entwicklung von Output und Tabelle für den Input `ababcbababaaaaaaa`

String Präfix-String-Index	Erweiterungs-character	Code
	a	1
	b	2
	c	3
1	b	4
2	a	5
4	c	6
3	b	7
5	b	8
8	a	9
1	a	10
10	a	11
11	a	12

Implementation der Stringtabelle

10.6 GIF

Das *Graphics Interchange Format* (GIF) wurde von CompuServe eingeführt und wird benutzt, um ggf. mehrere Bilder (Animation) in einer einzigen Datei zu speichern und zwischen verschiedenen Rechnersystemen auszutauschen. Bezogen auf die Anzahl der existierenden Dateien ist GIF eines der weitverbreitetsten Formate zum Speichern von Bilddaten.

Im Gegensatz zu vielen anderen Dateiformaten basiert GIF auf einem Strom der Daten. Das Format besteht aus einer Reihe von Datenpaketen, Blöcke genannt, kombiniert mit zusätzlicher Protokollinformation.

Das GIF-Format ist in der Lage, Bilddaten mit einer Farbtiefe von 1 bis 8 Bits zu speichern. Die Bilder werden immer im RGB-Modell unter Benutzung einer Farbtabelle gespeichert.

Bei den auf WWW-Seiten verwendeten Grafikformaten nimmt GIF den Spitzenplatz in punkto Häufigkeit ein. Insbesondere bei künstlich erzeugten Bildern mit einheitlich gefärbten Farbflächen ist es an Kompaktheit nicht zu schlagen.

Abbildung 10.1: Statisches GIF, Dateigröße: 2K

Zwei Kompressionsideen tragen zur Datenreduktion bei:

Farbpalette: Statt in jedem Pixel das komplette RGB-Tripel mit 3 Byte = 24 Bit Farbinformation zu speichern, werden für geeignetes p die 2^p wichtigsten Farben in der Farbpalette, gehalten und über einen p-Bit langen Index referiert. Für $p = 8$ schrumpft der Platzbedarf daher auf ein Drittel.

LZW: Das von *Lempel*, *Ziv* und *Welch* entwickelte und als Patent geschützte Verfahren zur Kompression beliebiger Zeichenfolgen basiert auf der Idee, in einer sogenannten Präfix-Tabelle die Anfangsstücke bereits gelesener Strings zu speichern und wiederholtes Auftauchen derselben Strings durch Verweise in die Tabelle zu kodieren.

Beide Ansätze zahlen sich insbesondere dann aus, wenn die Vorlage weite Bereiche mit identischer Information enthält, wie es bei computergenerierten Grafiken, wie z.B. Logos, der Fall ist. Zum einen enthält das Bild dann gar nicht die theoretisch verfügbare Zahl von $256^3 \approx 16$ Millionen Farben, sondern nur wenige Dutzend, und kann daher völlig verlustfrei durch eine Palette mit 256 Einträgen dargestellt werden. Zum anderen führen Folgen von identischen Pixelindizes zu kompakten Einträgen in der Präfixtabelle.

10.7 Erzeugung einer bildbezogenen Farbtabelle

Zunächst wird für jede Farbe die Häufigkeit ihres Auftretens ermittelt. Ggf. muß "vorquantisiert" werden von 24 Bit auf 15 Bit (je 5 Bit für Rot, Grün, Blau); hierdurch erhält das Histogramm maximal 32768 Einträge. Sei $d(a,b)$ der Abstand zweier Farbtupel a, b, z.B. $\sqrt{(a_{Rot} \Leftrightarrow b_{Rot})^2 + (a_{Gruen} \Leftrightarrow b_{Gruen})^2 + (a_{Blau} \Leftrightarrow b_{Blau})^2}$.

Optimaler Algorithmus

Gegeben sei eine Menge F von beobachteten Farben. Gesucht ist eine Menge M von Repräsentanten, so daß der Ausdruck

$$\Delta := \max_{x \in F} \min_{p \in M} d(p,x)$$

minimiert wird, d.h., Δ ist der maximale Abstand, den eine Farbe zu ihrem nächsten Repräsentanten hat.

Da die exakte Bestimmung von M eine exponentielle Laufzeit verursacht, begnügt man sich mit einer Näherungslösung.

Popularity-Algorithmus (1978)

```
Wähle die K häufigsten Farben.
```

Nachteil: Selten vorkommende Farben werden schlecht repräsentiert.

Diversity-Algorithmus (*xv*, John Bradley, 1989)

```
Initialisiere M mit der häufigsten Farbe
for i := 2 to K do
       erweitere M um die Farbe des Histogramms,
       die zu allen Farben aus M den größten Abstand hat
end
```

Median-Cut (Heckbert, MIT 1980)

```
Initialisiere RGB-Wuerfel mit Häufigkeiten der beobachteten Farbtupel
Initialisiere Wurzel des Schnittbaums mit Gesamtzahl der Pixel
While noch_nicht_genügend_Blätter do
       Wähle Blatt mit der größten Pixelzahl
       Bestimme umschliessende Box
       Bestimme Achse mit größtem Wertebereich
       Durchlaufe Box längs dieser Achse
       Teile am Median in zwei Hälften
       Trage Hälften als Soehne ein
end
Für jedes Blatt wähle den Mittelwert aller in ihm liegenden Farben.
```

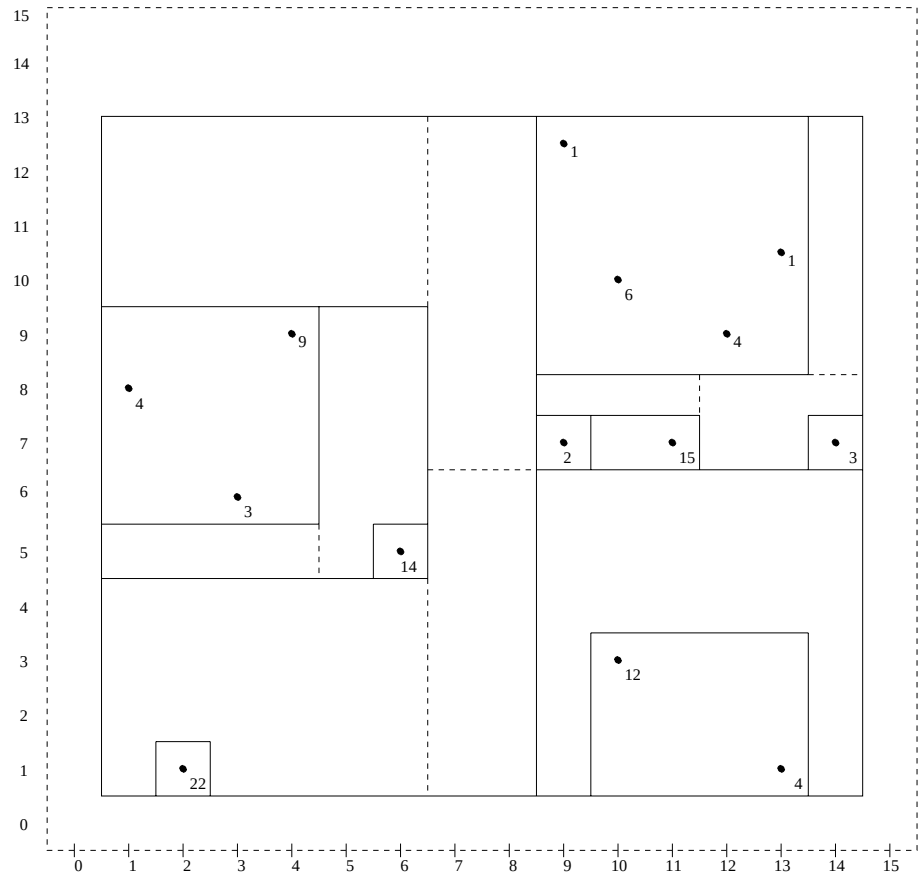


Abbildung 10.2: Partitionierung der Ebene beim Median-Cut-Algorithmus

Abbildung 10.2 zeigt die Anwendung des Median-Cut-Algorithmus anhand eines Beispiels. Zur einfachen Darstellung ist der Farbbaum zweidimensional gewählt. Gezeigt wird die Verteilung der Farbtupel aus dem Bereich $[0 \ldots 15] \times [0 \ldots 15]$ in einem Bild mit $10 \times 10 = 100$ Pixeln. Eingetragen sind an der Position x/y die Häufigkeit des Farbtupels x, y. Schnittlinien sind gestrichelt, umschließende Boxen durchgezogen gezeichnet.

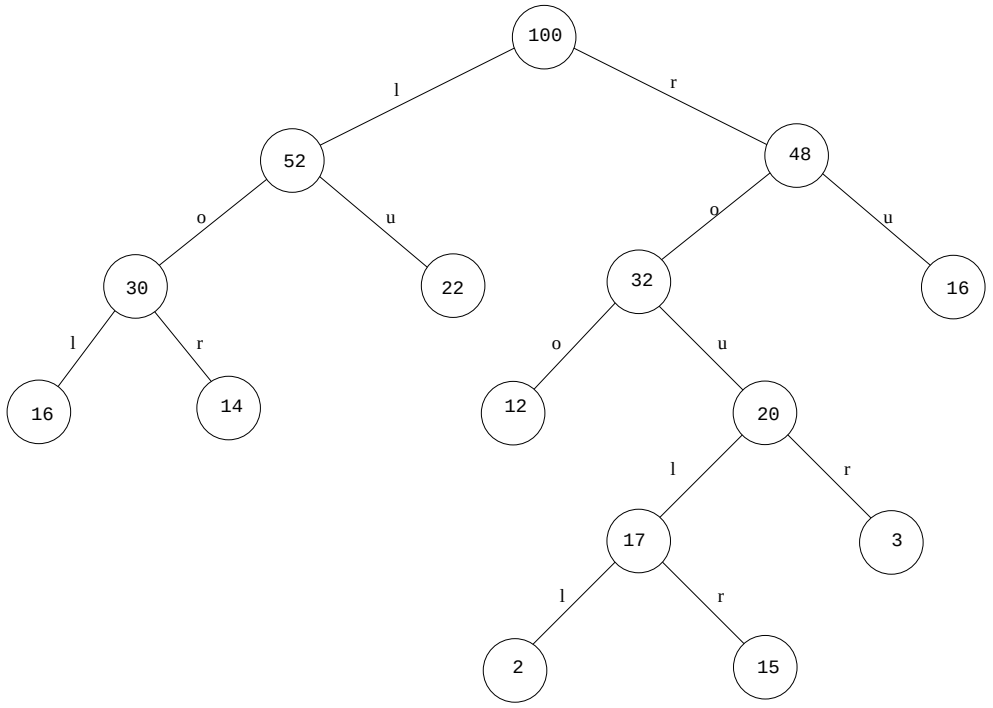
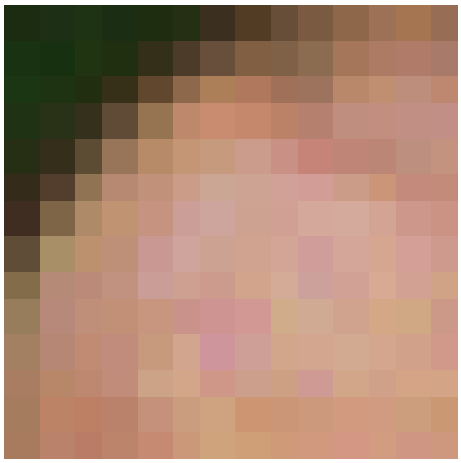


Abbildung 10.3: Schnittbaum beim Median-Cut-Algorithmus

Abbildung 10.3 zeigt für das vorliegende Beispiel den Schnittbaum nach Anwendung des Median-Cut-Algorithmus. Die Knoten sind markiert mit der Anzahl der noch zu quantisierenden Pixel, an den Kanten ist vermerkt, ob es sich um einen Links/Rechts- oder um einen Oben/Unten-Schnitt handelt.

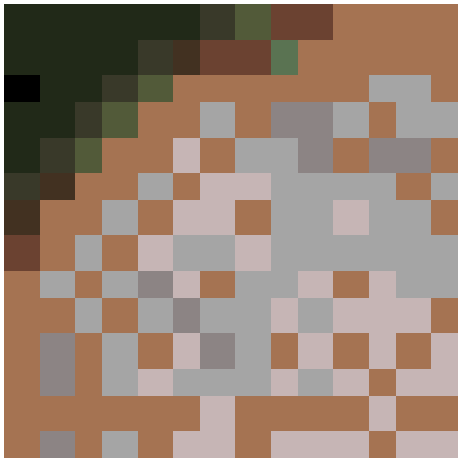
Floyd-Steinberg-Dithering (1975)
Der bei Verwendung einer Farbtabelle verursachte Fehler beim Quantisieren eines Pixels wird auf die Nachbarpixel verteilt (bevor diese quantisiert werden).

```
for (i = 0; i < M; i++)
for (j = 0; j < N; j++)
{
   x = f[i][j];   /* hole Original-Farbe */
   k = p(x);      /* finde naechsten Repraesentant */
   q[i][j] = k;   /* zeichne quantisiertes Pixel */
   e = d(x, k);   /* bestimme Fehlerabstand */
   f [i][j + 1]     = f[i][j + 1]     + e * 3.0/8.0;
   f [i + 1][j]     = f[i + 1][j]     + e * 3.0/8.0;
   f [i + 1][j + 1] = f[i + 1][j + 1] + e * 1.0/4.0;
}
```

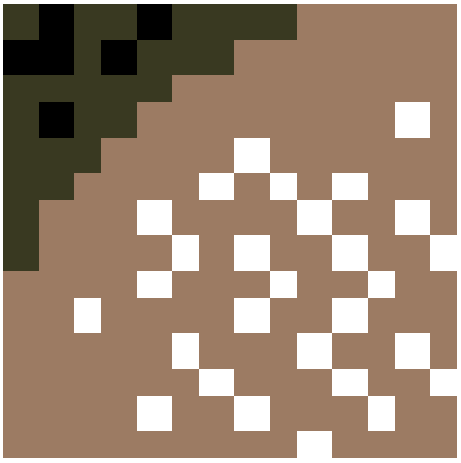


24 Bit pro Pixel,
16 Mill. Farben

8 Bit pro Pixel,
256 Farben



4 Bit pro Pixel,
16 Farben



2 Bit pro Pixel,
4 Farben

Abbildung 10.4: Auswirkung des Median-Cut-Algorithmus

Abbildung 10.4 zeigt Original und quantisierte Versionen einer 14×14 Ausschnittvergrößerung, erstellt von einem 814×517 True-Color-Bild. Die Zahl der Farben bezieht sich jeweils auf das Gesamtbild. Verwendet wurde der Median-Cut-Algorithmus mit anschließendem Floyd-Steinberg-Dithering.

10.8 Kompression nach JPEG

Die **J**oint **P**hotographic **E**xpert **G**roup bildete sich aus Mitgliedern der Standardisierungsgremien CCITT (Consultative Committee for International Telephone and Telegraph) und ISO (International Standardization Organization). JPEG wurde schließlich zum Namen für den Standard selbst.

Zunächst wird das RGB-Bild in den YUV-Raum transformiert. Da das Auge für Helligkeitssprünge sensitiver ist als für Farbdifferenzen, kann man nun die Y-Matrix in der vollen Auflösung belassen und in den U, V-Matrizen jeweils 4 Pixel mitteln (4:1:1 Subsampling).

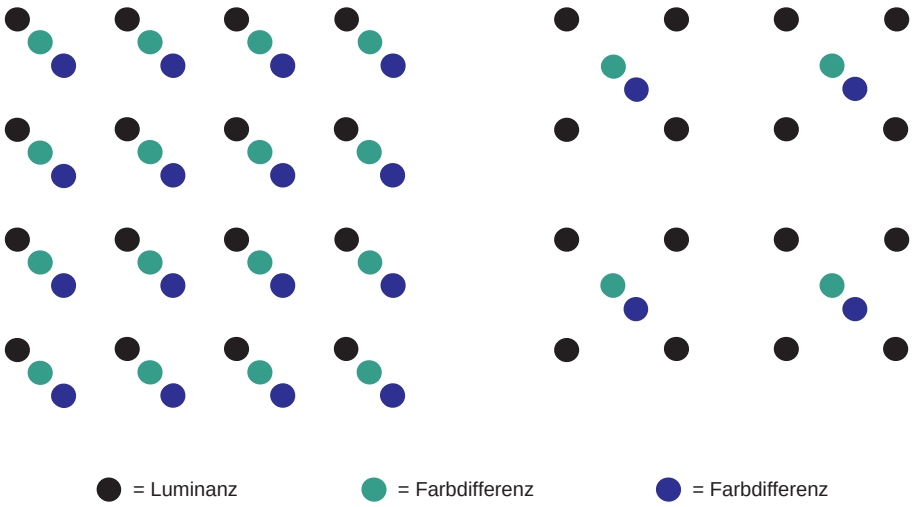


Abbildung 10.5: 4:1:1 Subsampling

Für je 4 Originalpixel mit insgesamt 12 Bytes werden nun $4 + 1 + 1 = 6$ Bytes benötigt (pro Bildpunkt also $6 \cdot 8/4 = 12$ Bit). Die Reduktion beträgt 50 %.

Beim JPEG-Verfahren werden nun die drei Matrizen in Blöcke mit 8×8 Abtastwerten aufgeteilt. Anschließend durchlaufen die Blöcke folgende Schritte:

1. Diskrete Cosinus Transformation
2. Rundung der Frequenzkoeffizienten
3. Lauflängenkodierung der quantisierten Werte
4. Huffmancodierung der Lauflängenbeschreibung.

Um aus dem komprimierten Bild das Original zu rekonstruieren, werden die Schritte in umgekehrter Reihenfolge und inverser Funktionalität durchlaufen.

Durch die Wahl der Rundungstabelle läßt sich der Tradeoff zwischen Qualität und Kompression beliebig steuern. Ein typisches Farbbild läßt sich ohne für das Auge sichtbare Artefakte auf 10 % seiner Originalgröße reduzieren. Eine Reduktion auf 5 % verursacht oft nur leichte, kaum wahrnehmbare Verzerrungen.

DCT (Diskrete Cosinus Transformation)
Die diskrete Cosinus-Transformation ist ein Spezialfall der Fouriertransformation. Es wird ausgenutzt, daß für "gerade" Funktionen (d.h. $f(\Leftrightarrow x) = f(x)$) der Sinus-Term mit seinem imaginären Anteil wegfällt. Ein zweidimensionaler Bildbereich läßt sich durch Spiegelung an der y-Achse künstlich gerade machen.

$$s[u,v] := \frac{1}{4} \cdot c_u \cdot c_v \cdot \sum_{x=0}^{7} \sum_{y=0}^{7} f[x,y] \cdot \cos\frac{(2x+1)\cdot u \cdot \pi}{16} \cdot \cos\frac{(2y+1)\cdot v \cdot \pi}{16}$$

$$c_u, c_v := \begin{cases} \frac{1}{\sqrt{2}} & \text{für } u,v = 0 \\ 1 & \text{sonst} \end{cases}$$

Hierdurch wird eine 8×8 Ortsmatrix in eine 8×8 Frequenzmatrix transformiert.

Mit Hilfe der inversen DCT lassen sich die Originalwerte rekonstruieren.

$$f[x,y] := \frac{1}{4} \sum_{u=0}^{7} \sum_{v=0}^{7} c_u \cdot c_v \cdot s[u,v] \cdot \cos\frac{(2x+1)\cdot u \cdot \pi}{16} \cdot \cos\frac{(2y+1)\cdot v \cdot \pi}{16}$$

Für die Bildverarbeitung wird der Pixelwertebereich 0..255 in das symmetrische Intervall $\Leftrightarrow$128..127 verschoben. Der Wertebereich von s liegt dann im Intervall $\Leftrightarrow$1024.. + 1023. Der errechnete Koeffizient s_{00} entspricht dem Anteil der Frequenz null in beiden Achsen und wird *DC*-Koeffizient (Gleichspannungsanteil) bezeichnet. Die übrigen Koeffizienten werden *AC*-Koeffizienten (Wechselspannungsanteil) genannt. Z.B. bezeichnet s_{77} die höchste, in beiden Richtungen auftretende Frequenz.

Die Eingangsmatrix M läßt sich auch durch $T \cdot M \cdot T'$ transformieren mit Hilfe der Matrix T:

```
0.353553    0.353553    0.353553    0.353553    0.353553    0.353553    0.353553    0.353553
0.490393    0.415735    0.277785    0.097545   -0.097545   -0.277785   -0.415735   -0.490393
0.461940    0.191342   -0.191342   -0.461940   -0.461940   -0.191342    0.191342    0.461940
0.415735   -0.097545   -0.490393   -0.277785    0.277785    0.490393    0.097545   -0.415735
0.353553   -0.353553   -0.353553    0.353553    0.353553   -0.353553   -0.353553    0.353553
0.277785   -0.490393    0.097545    0.415735   -0.415735   -0.097545    0.490393   -0.277785
0.191342   -0.461940    0.461940   -0.191342   -0.191342    0.461940   -0.461940    0.191342
0.097545   -0.277785    0.415735   -0.490393    0.490393   -0.415735    0.277785   -0.097545
```

Quantisierung
Die errechnete Matrix hat von links oben nach rechts unten Werte abnehmender Größe. Da die Werte rechts unten den hohen, eher unwichtigen Frequenzen entsprechen, werden alle Einträge mit Faktoren zunehmender Größe dividiert.

$$r[u,v] := \frac{s[u,v]}{q[u,v]}$$

95	88	87	95	88	95	95	95
143	144	151	151	153	170	183	181
153	151	162	166	162	151	126	117
143	144	133	130	143	153	159	175
123	112	116	130	143	147	162	189
133	151	162	166	170	188	166	128
160	168	166	159	135	101	93	98
154	155	153	144	126	106	118	133

Bildmatrix

↓

98	95	91	89	90	95	101	106
140	143	149	156	163	167	168	167
146	149	154	159	159	151	137	126
149	142	136	137	145	156	163	166
119	117	118	125	140	157	170	176
137	147	160	170	172	166	157	150
166	167	164	152	132	112	99	93
151	153	150	139	125	118	119	123

rekonstruierte Bildmatrix

↑

93	2	-8	-7	3	1	1	-2
-38	-58	11	17	-3	5	5	-3
-84	63	-1	-17	2	7	-4	-0
-51	-37	-10	13	-10	5	-1	-4
-85	-42	50	-8	18	-5	-1	1
-63	66	-13	-1	2	-6	-2	-2
-16	14	-37	18	-12	4	3	-3
-53	31	-7	-10	23	-0	2	2

DCT-Koeffizienten

→

31	0	-1	0	0	0	0	0
-7	-8	1	1	0	0	0	0
-12	7	0	-1	0	0	0	0
-5	-3	0	0	0	0	0	0
-7	-3	3	0	0	0	0	0
-4	4	0	0	0	0	0	0
-1	0	-1	0	0	0	0	0
-3	1	0	0	0	0	0	0

quantisierte DCT-Koeffizienten

3	5	7	9	11	13	15	17
5	7	9	11	13	15	17	19
7	9	11	13	15	17	19	21
9	11	13	15	17	19	21	23
11	13	15	17	19	21	23	25
13	15	17	19	21	23	25	27
15	17	19	21	23	25	27	29
17	19	21	23	25	27	29	31

Quantisierungsmatrix

Entropiekodierung
Die *DC*-Koeffizienten benachbarter Blöcke unterscheiden sich nur wenig und werden daher als Differenz zum Vorgängerblock übertragen.

Die *AC*-Koeffizienten werden zunächst in eine Zick-Zack-Sequenz umgeordnet:

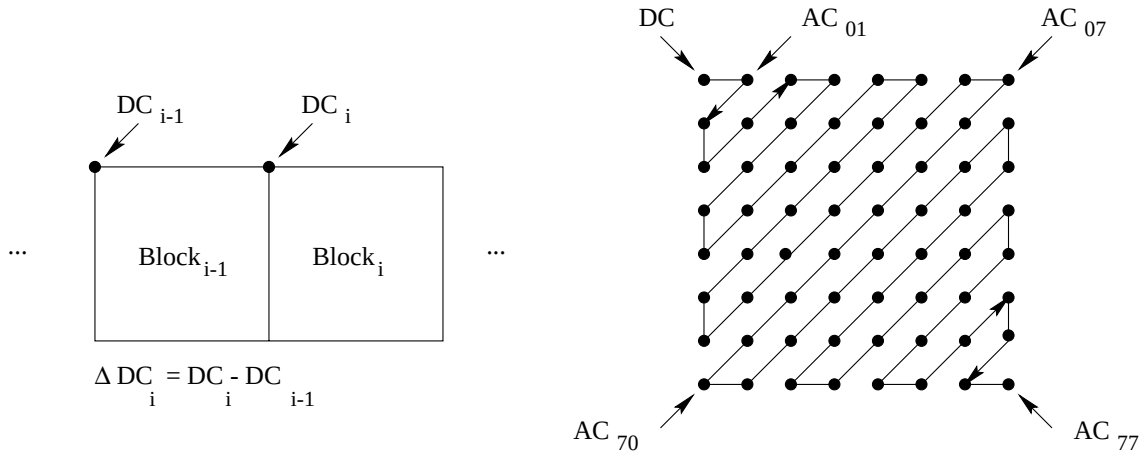


Abbildung 10.6: Durchlaufsequenz pro Macroblock

Die AC-Koeffizienten längs dieses Weges bis zum letzten Eintrag ungleich Null werden als Folge von Paaren beschrieben:

Symbol 1: Länge der ununterbrochenen Folge von Nullen vor diesem Wert (Runlength)
Anzahl der Bits, die zur Darstellung des Wertes erforderlich sind.
Symbol 2: der Wert selbst

Der quantisierte DCT-Koeffizient AC_{70} aus vorigem Beispiel wird beschrieben als Tupel $<(11,2),\Leftrightarrow 3>$, denn vor ihm in der Zickzacksequenz stehen 11 Nullen, sein Wert kann mit 2 Bit codiert werden, und sein Wert beträgt $\Leftrightarrow 3$.

Für das Symbol 1 gibt es Häufigkeitsverteilungen, aus denen sich ein Huffman-Code konstruieren läßt. Für das Symbol 2 wählt man ein 2-er Komplement, welches berücksichtigt, daß bei angekündigten N Bits nur solche Zahlen kodiert werden müssen, für die $N \Leftrightarrow 1$ Bits nicht ausreichen.

Länge/Bitzahl	Codierung
0/0 (EOB)	1010
0/1	00
0/2	01
0/3	100
0/4	1011
0/5	11010
0/6	1111000
0/7	11111000
0/8	1111110110
0/9	1111111110000010
0/10	1111111110000011
1/1	1100
1/2	11011
1/3	1111001
⋮	⋮
2/1	11100
2/2	11111001
2/3	1111110111
⋮	⋮
3/1	111010
3/2	111110111
3/3	111111110101
⋮	⋮
11/1	1111111001
11/2	1111111111010000
⋮	⋮
15/10	1111111111111110

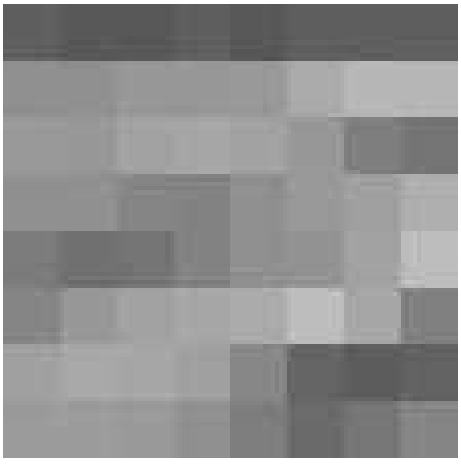
Huffman Codierung
für Symbol 1

Wert	Codierung
15	1111
14	1110
13	1101
12	1100
11	1011
10	1010
9	1001
8	1000
7	111
6	110
5	101
4	100
3	11
2	01
1	1
-1	0
-2	10
-3	00
-4	011
-5	010
-6	001
-7	000
-8	0111
-9	0110
-10	0101
-11	0100
-12	0011
-13	0010
-14	0001
-15	0000

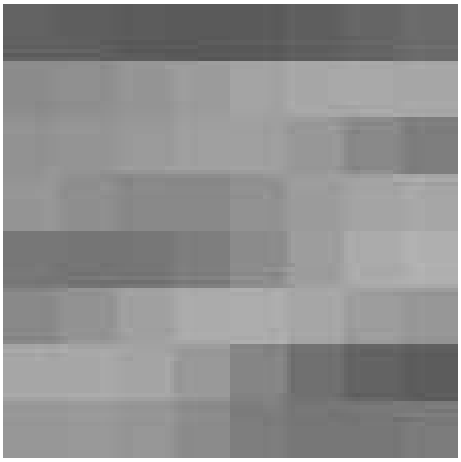
Komplement-Codierung
für Symbol 2

Symbol 1	Symbol 2	Huffman für Symbol 1	2-er Komplement für Symbol 2
(1, 3)	-7	1111001	000
(0, 4)	-12	1011	0011
(0, 4)	-8	1011	0111
(0, 1)	-1	00	0
(1, 1)	1	1100	1
(0, 3)	7	100	111
(0, 3)	-5	100	010
(0, 3)	-7	100	000
(0, 2)	-3	01	00
(1, 1)	1	1100	1
(3, 1)	-1	111010	0
(1, 2)	-3	11011	00
(0, 3)	-4	100	011
(0, 1)	-1	00	0
(0, 3)	4	100	100
(0, 2)	3	01	11
(11, 2)	-3	1111111111010000	00
(0, 1)	1	00	1
(0, 1)	-1	00	0
EOB		1010	

Symbole und ihre Kodierung für die im Beispiel vorgestellten quantisierten AC-Koeffizienten



Ausgangsbildmatrix



rekonstruierte Bildmatrix

Abbildung 10.7: 8 x 8 Matrix vor und nach der JPG-Kompression.

95	88	87	95	88	95	95	95
143	144	151	151	153	170	183	181
153	151	162	166	162	151	126	117
143	144	133	130	143	153	159	175
123	112	116	130	143	147	162	189
133	151	162	166	170	188	166	128
160	168	166	159	135	101	93	98
154	155	153	144	126	106	118	133

Dezimaldarstellung der Bildbereich-Matrix

01011111	01011000	01010111	01011111	01011000	01011111	01011111	01011111
10001111	10010000	10010111	10010111	10011001	10101010	10110111	10110101
10011001	10010111	10100010	10100110	10100010	10010111	01111110	01110101
10001111	10010000	10000101	10000010	10001111	10011001	10011111	10101111
01111011	01110000	01110100	10000010	10001111	10010011	10100010	10111101
10000101	10010111	10100010	10100110	10101010	10111100	10100110	10000000
10100000	10101000	10100110	10011111	10000111	01100101	01011101	01100010
10011010	10011011	10011001	10010000	01111110	01101010	01110110	10000101

8-Bit-Codierung für Bildbereich-Matrix

00011111

quantisierter DC-Koeffizient

111100100010110011101101110001100110011110001010000001001100111100100010110011101101110001100110011110001010000001001100111101001100100011000100100011111111111111010000000010001010

Huffman-Codierung für quantisierte AC-Koeffizienten

motel.tif
100%

motel80.jpg
9%

motel40.jpg
5%

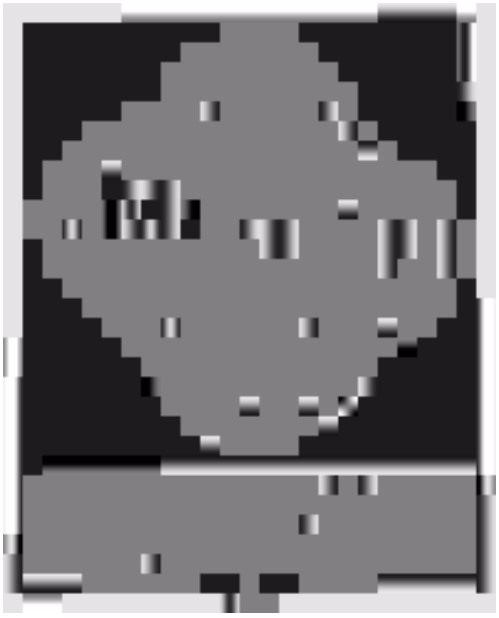
motel20.jpg
4%

Abbildung 10.8: Kompression nach JPEG. Platzbedarf in Prozent bezogen auf tif-Datei

motel10.jpg
3%

motel05.jpg
2%

motel02.jpg
1.5%



motel01.jpg
1%

Abbildung 10.9: Kompression nach JPEG. Platzbedarf in Prozent bezogen auf tif-Datei

Kapitel 11

3D-Grundlagen

Für 3-dimensionale Objekte gibt es mehrere Möglichkeiten der Repräsentation (d.h. Definition des Objekts) und der Darstellung (d.h. Projektion des Objekts auf den Bildschirm).

11.1 Repräsentation und Darstellung

Repräsentation

- Elementarobjekt mit Definitionspunkten,
- Drahtmodell mit Kantenliste,
- Flächenmodell mit Flächenliste,
- Flächenmodell mit Halbkantendarstellung,
- CSG (constructive solid geometry) mit mengentheoretischer Verknüpfung von Elementarobjekten.

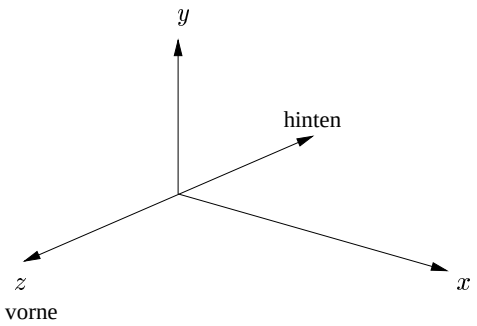
Darstellung

- Drahtmodell mit sämtlichen Kanten,
- Drahtmodell mit Entfernung verdeckter Kanten,
- Flächenmodell mit Schattierung, ohne abgewandte Flächen,
- Flächenmodell mit Berechnung von Lichtreflektion, ohne verdeckte Teile von Flächen,
- Körpermodell mit Berechnung von Schattenbildung, Spiegelungen und Brechungen.

11.2 3D-Koordinatensystem

Weit verbreitet ist das kartesische Koordinaten-System, z.B. in der rechtshändigen Form.

Bei gespreizten Fingern der rechten Hand zeigt der Zeigefinger in x-Richtung, der Mittelfinger in y-Richtung, der Daumen in z-Richtung.



11.3 Länge und Kreuzprodukt

Gegeben sei ein 3D-Vektor

$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix}$$

Seine **Länge** ist definiert als

$$|v| := \sqrt{v_1^2 + v_2^2 + v_3^2}$$

Gegeben seien zwei 3D-Vektoren

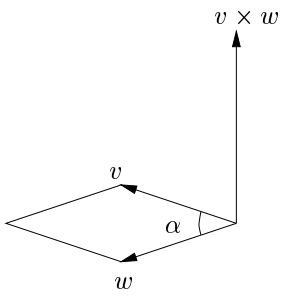
$$v = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \end{pmatrix} \text{ und } w = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \end{pmatrix}$$

Das **Kreuzprodukt** von v und w ist definiert als

$$v \times w = \begin{pmatrix} v_2 \cdot w_3 \Leftrightarrow v_3 \cdot w_2 \\ v_3 \cdot w_1 \Leftrightarrow v_1 \cdot w_3 \\ v_1 \cdot w_2 \Leftrightarrow v_2 \cdot w_1 \end{pmatrix}$$

Der Vektor $v \times w$ steht senkrecht auf v und steht senkrecht auf w. Seine Länge entspricht der Fläche des durch v und w aufgespannten Parallelogramms, d.h.

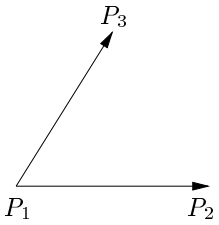
$$|v \times w| = |v| \cdot |w| \cdot \sin(\alpha)$$



In einem rechtshändigen Koordinatensystem entspricht bei gespreizten Fingern der Zeigefinger v, der Mittelfinger w, der Daumen $v \times w$.

Anwendung des Kreuzprodukts

Gegeben sei eine Fläche durch 3 nicht kollineare Punkte P_1, P_2, P_3.



Der Vektor $(P_2 \Leftrightarrow P_1) \times (P_3 \Leftrightarrow P_1)$ bildet den Normalenvektor zur Fläche.
Ist eine Ebene durch ihre Ebenengleichung $Ax + By + Cz + D = 0$ gegeben, so ergibt sich der Normalenvektor als (A, B, C).
Ist ein Normalenvektor (A, B, C) gegeben, so errechnet sich D durch Einsetzen eines beliebigen Punktes der Ebene.
Ein Punkt (x, y, z) liegt

oberhalb der Ebene,	falls	$Ax + By + Cz + D > 0$
unterhalb der Ebene,	falls	$Ax + By + Cz + D < 0$
in der Ebene,	falls	$Ax + By + Cz + D = 0$

11.4 Skalarprodukt

Gegeben seien zwei n-dimensionale Vektoren v, w.
Das **Skalarprodukt** lautet:

$$v \cdot w := \sum_{i=1}^{n} v_i \cdot w_i$$

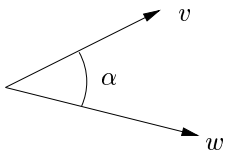
Für Vektoren a, b, c und $s \in \mathbb{R}$ gilt:

$a \cdot b$	$=$	$b \cdot a$	(Symmetrie)
$(a + c) \cdot b$	$=$	$a \cdot b + c \cdot b$	(Linearität)
$(sa) \cdot b$	$=$	$s(a \cdot b)$	(Homogenität)
$\lvert b \rvert$	$=$	$\sqrt{b \cdot b}$	(euklidische Norm)

Anwendungen des Skalarprodukts:

Gegeben zwei Vektoren v, w. Für den Winkel α zwischen v und w gilt:

$$\cos(\alpha) = \frac{v \cdot w}{|v| \cdot |w|}.$$



Es gilt:
$v \cdot w < 0 \quad \Leftrightarrow \quad v$ und w sind mehr als 90° auseinander
$v \cdot w = 0 \quad \Leftrightarrow \quad v$ steht senkrecht auf w
$v \cdot w > 0 \quad \Leftrightarrow \quad v$ und w sind weniger als 90° auseinander

Es gilt: $n \cdot r = D$ beschreibt eine Ebene für $D \in \mathbb{R}$ und Normalenvektor n. Alle Lösungsvektoren r liegen (als Punkte aufgefaßt) auf der Ebene. Das Skalarprodukt aller Ebenenpunkte (als Vektoren geschrieben) mit dem Normalenvektor ist konstant.

11.5 Matrixinversion

Analog zum zweidimensionalen Fall werden die dreidimensionalen Transformationen durch Verknüpfung homogener Koordinaten mit 4×4-Transformationsmatrizen dargestellt.

Sei $A = (a_{ik})$, $1 \leq i, k \leq 4$, eine 4×4-Matrix:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{41} & a_{42} & a_{43} & a_{44} \end{pmatrix}$$

Ist $D = det\ A = |a_{ik}|$ die *Determinante* von A, so bezeichnet man als *Unterdeterminante* des Elementes a_{ik} diejenige 3-reihige Determinante, die aus D durch Streichen der i-ten Zeile und der k-ten Spalte hervorgeht. Unter der *Adjunkten* A_{ik} des Elementes a_{ik} versteht man die mit dem Faktor $(\Leftrightarrow 1)^{i+k}$ versehene Unterdeterminante von a_{ik}.

Beispiel:

$$A_{23} = \Leftrightarrow \begin{vmatrix} a_{11} & a_{12} & a_{14} \\ a_{31} & a_{32} & a_{34} \\ a_{41} & a_{42} & a_{44} \end{vmatrix} = \Leftrightarrow [a_{11}(a_{32}a_{44} \Leftrightarrow a_{34}a_{42}) \Leftrightarrow a_{12}(a_{31}a_{44} \Leftrightarrow a_{34}a_{41}) + a_{14}(a_{31}a_{42} \Leftrightarrow a_{32}a_{41})]$$

Die Adjunkten sind nützlich zur Berechnung der Determinanten von A sowie der *inversen Matrix* A^{-1} :

$$\det A = \sum_{k=1}^{4} a_{1k} A_{1k} \qquad A^{-1} = \frac{1}{\det A} \begin{pmatrix} A_{11} & A_{21} & A_{31} & A_{41} \\ A_{12} & A_{22} & A_{32} & A_{42} \\ A_{13} & A_{23} & A_{33} & A_{43} \\ A_{14} & A_{24} & A_{34} & A_{44} \end{pmatrix}$$

Kapitel 12

3D-Repräsentation

Die Repräsentation dreidimensionaler Objekte in der Computergrafik teilt sich in verschiedene Repräsentationsklassen, die hierarchisch aufgebaut sind.

12.1 Elementarobjekte

Für den Benutzer sollte die Beschreibung einer Szene durch *Elementarobjekte* erfolgen. Diese können unterschiedlich kompliziert sein, sollten aber durch wenige Parameter beschrieben werden können. Eine Kugel z.B. ist bereits durch Mittelpunkt und Radius eindeutig im Raum plaziert. Es ist sinnvoll, jedes Objekt in seinem eigenen, lokalen Modellkoordinatensystem zu definieren. Dessen Ursprung wird im Inneren des Objekts gewählt und das gesamte Objekt in ein Einheitsvolumen (z.B. $\Leftrightarrow 1 \leq x, y, z \leq +1$) eingeschlossen. Für Orts- und Größenveränderungen sind Transformationen zuständig.

12.2 Drahtmodell

In der nächsten Repräsentationsklasse wird das Elementarobjekt als *Drahtmodell* durch eine Liste von Kanten repräsentiert. Jede Kante besteht aus zwei Punkten im kartesischen Koordinatensystem.

Beim Würfel verbinden die Kanten die Eckpunkte, bei einer Kugel werden die Längen- und Breitenkreise durch n-Ecke angenähert, wobei mit n die Güte der Approximation steigt. Das Drahtmodell skizziert nur die Umrisse eines Objekts und enthält keine zusätzlichen Flächen- oder Volumeninformationen.

12.3 Flächenmodell

Beim *Flächenmodell* werden Objekte durch approximierte oder analytische Flächen, z.B. durch eine Liste von konvexen Polygonen, repräsentiert. Ein solches Polygon wird durch seine Eckpunkte beschrieben, die durch Kanten verbunden sind.

Punkteliste	Kantenliste	Flächenliste
$P_1 : (x_1, y_1, z_1)$	$k_1 : P_1, P_2$	$F_1 : k_1, k_2, k_3$
$P_2 : (x_2, y_2, z_2)$	$k_2 : P_2, P_3$	$F_2 : k_1, k_6, k_4$
$P_3 : (x_3, y_3, z_3)$	$k_3 : P_3, P_1$	$F_3 : k_2, k_6, k_5$
$P_4 : (x_4, y_4, z_4)$	$k_4 : P_1, P_4$	$F_4 : k_3, k_4, k_5$
	$k_5 : P_4, P_3$	
	$k_6 : P_4, P_2$	

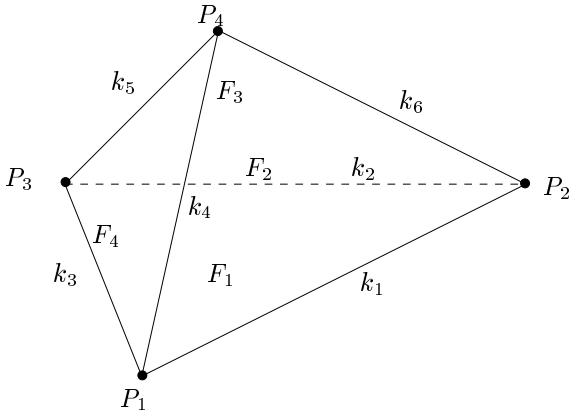


Abbildung 12.1: Tetraeder als Flächenmodell

Zur vollständigen Beschreibung einer Fläche gehört noch die Angabe, welche Seite "innen" und welche Seite "außen" liegt. Dies geschieht durch Angabe des Normalenvektors: Er steht senkrecht auf der Fläche und zeigt von innen nach außen. Für die Approximation gekrümmter Flächen wird häufig pro Eckpunkt eine Normale verwendet.

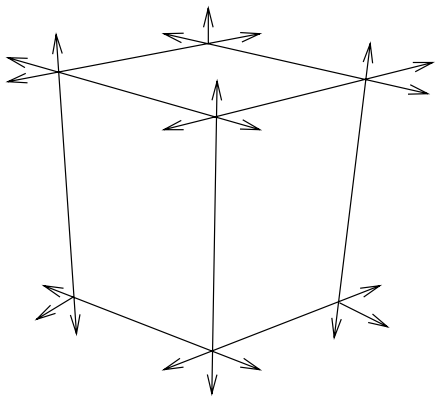


Abbildung 12.2: Würfel mit Normalenvektoren

12.4 CSG (constructive solid geometry)

Jedes Objekt wird beschrieben durch einen binären Baum, dessen Blätter beschriftet sind mit Elementarobjekten und dessen innere Knoten beschriftet sind mit den Mengenoperationen $\cup$ (Vereinigung), $\cap$ (Durchschnitt), $\setminus$ (Differenz).

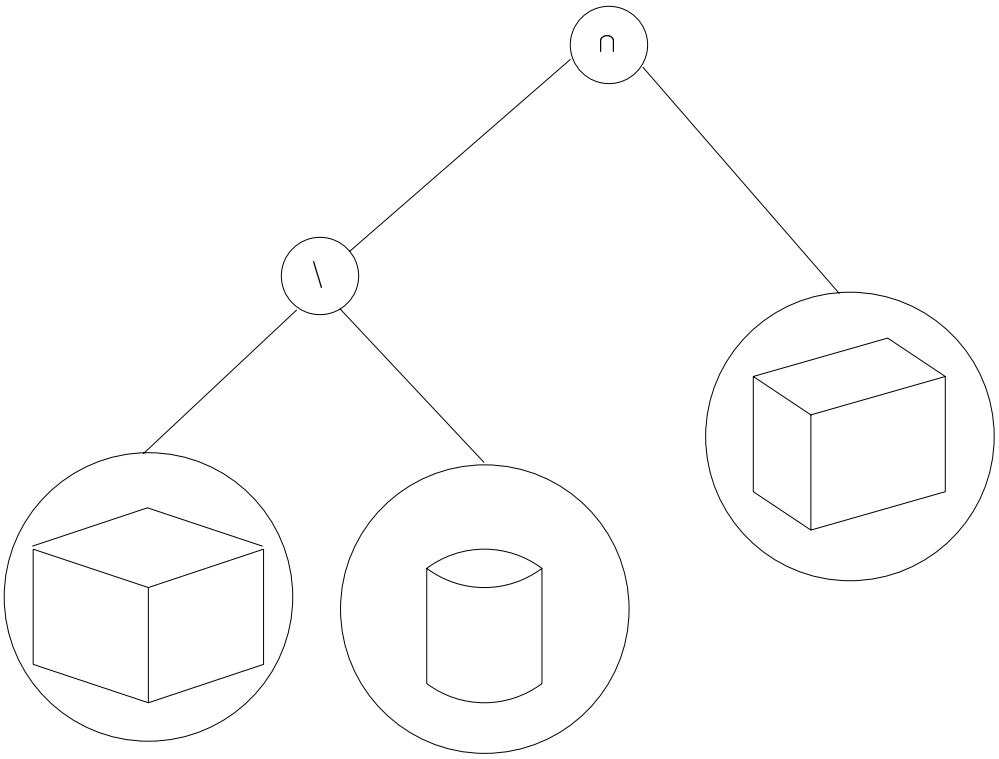


Abbildung 12.3: CSG-Repräsentation eines Körpers

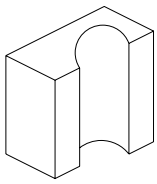


Abbildung 12.4: Darstellung des resultierenden Körpers

12.5 Flächenmodell mit Halbkantendarstellung

Jedes Objekt enthält eine Liste von Flächen, die von Halbkanten begrenzt werden. Die Halbkanten sind von außen betrachtet im Uhrzeigersinn orientiert und zeigen auf ihre jeweils linke Nachbarfläche sowie ihre Anfangs- und Endpunkte.

Die Halbkantendarstellung eignet sich zur effizienten Entfernung von verdeckten Kanten und Flächen. Eine Kante zwischen Punkt P_1 und Punkt P_2, welche die Flächen F_1 und F_2 trennt, taucht einmal als Halbkante (P_1, P_2) in der Kantenliste zu F_2 auf mit einem Verweis auf die Nachbarfläche F_1 und ein weiteres Mal als (P_2, P_1) in der Kantenliste zu F_1 mit einem Verweis auf die Nachbarfläche F_2. Werden nun alle Halbkanten einer Fläche F_1 bearbeitet, so regelt die Sichtbarkeit von F_1 und die Sichtbarkeit der jeweils anstoßenden Fläche die Sichtbarkeit der jeweiligen Halbkante. Das doppelte Zeichnen einer Kante läßt sich vermeiden, indem bei jeder Fläche vermerkt wird, ob ihre Halbkanten bereits bearbeitet wurden.

Auch die Flächennormalen können in der Datenstruktur gespeichert werden.

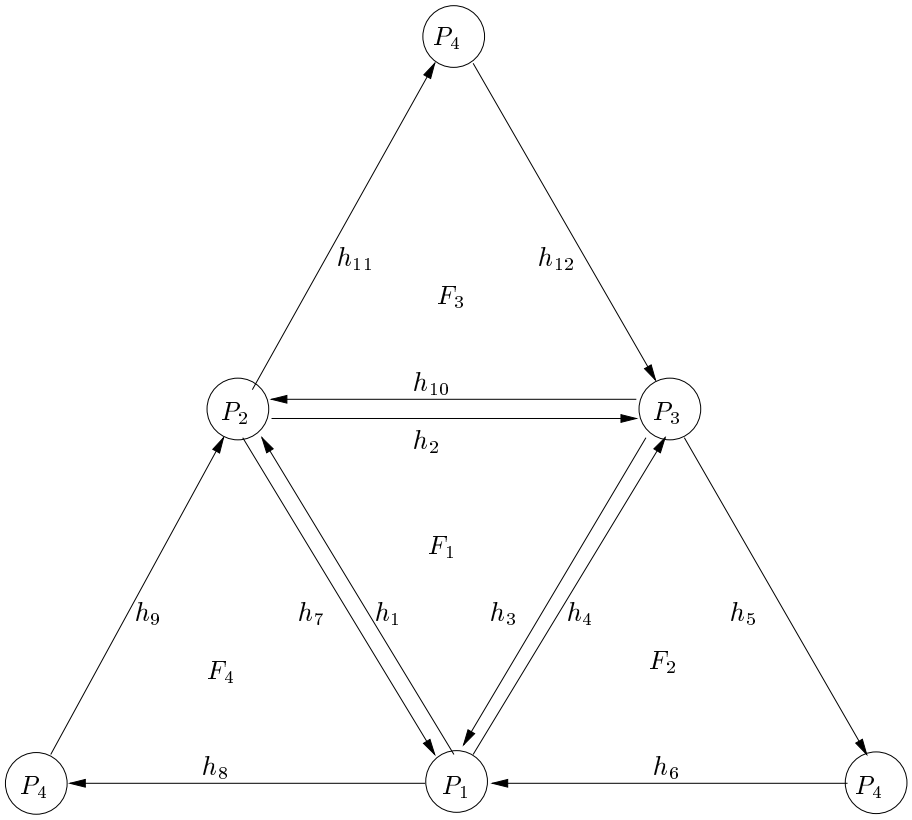


Abbildung 12.5: Tetraeder mit 4 Knoten, 12 Halbkanten, 4 Flächen

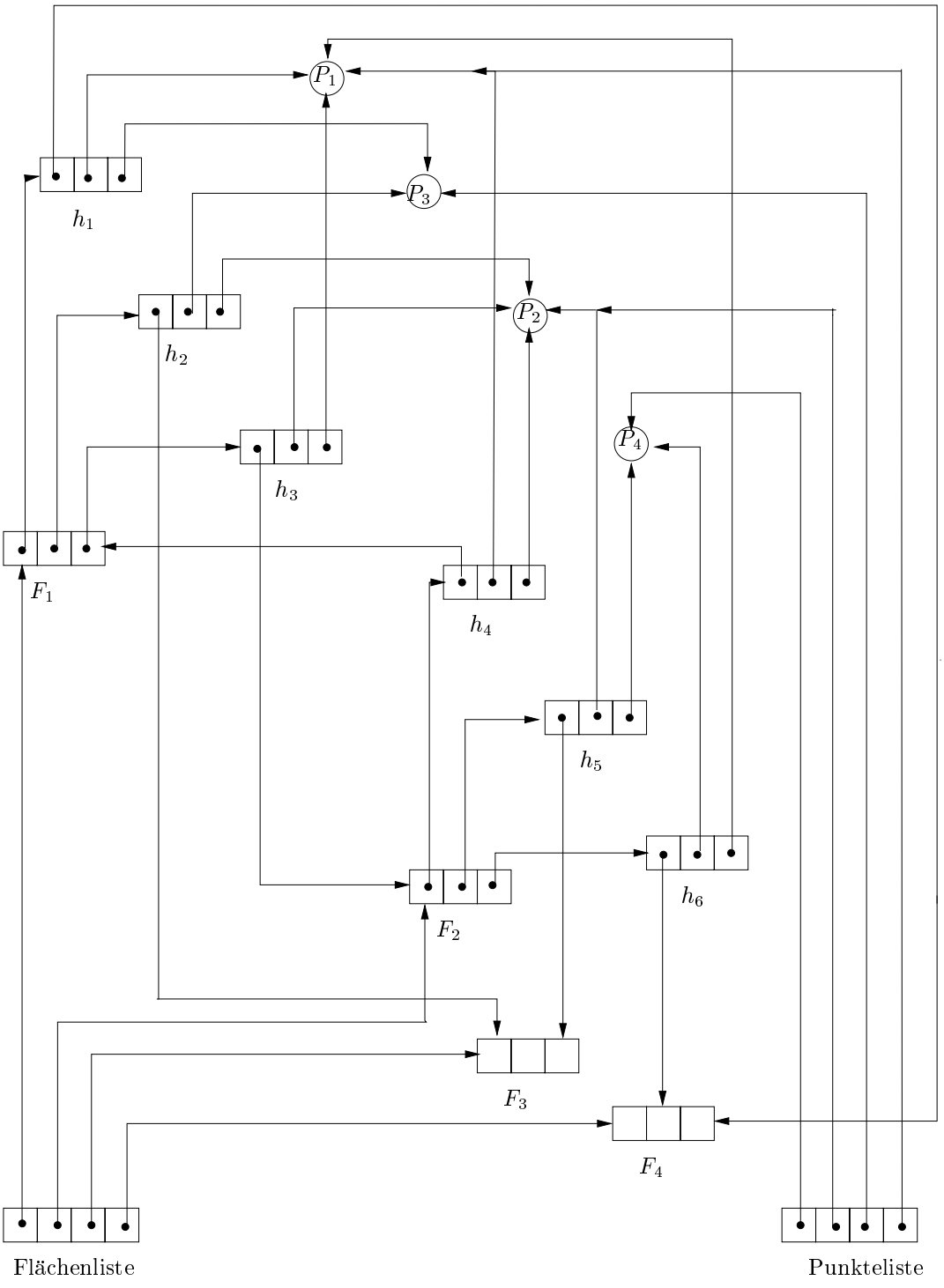


Abbildung 12.6: Objektstruktur für Tetraeder. Nicht gezeichnet sind Halbkanten für F_3 und F_4