Referat "Image Representation" im Proseminar "Computer Science Unplugged"
Dozent Prof. M. Hofmann
Referent Teja Philipp, 127906378
Datum 29.10.2004

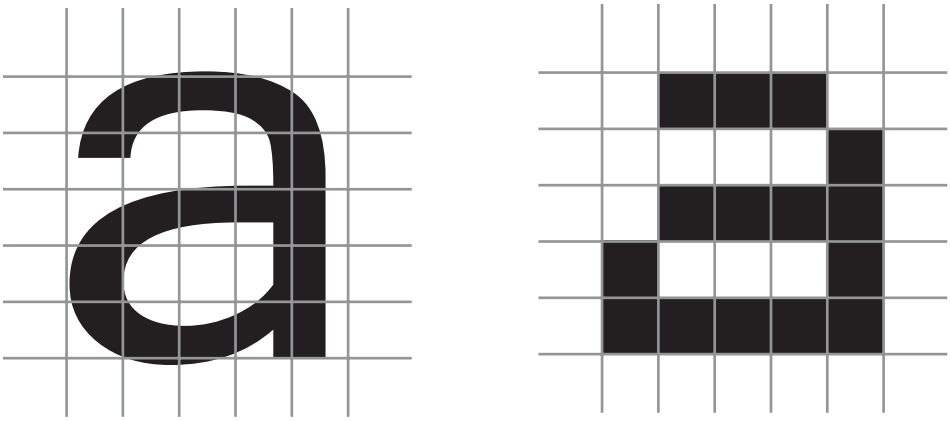
IMAGE REPRESENTATION

1. Digitalisierung des Bildmaterials

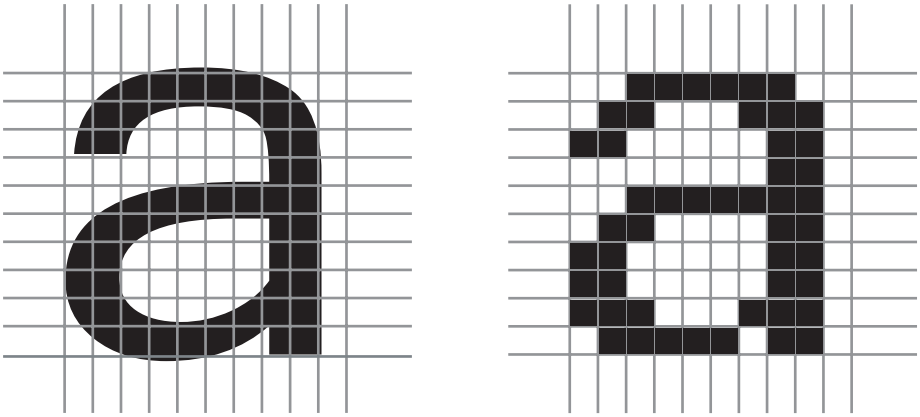
Um ein Bild, ein Foto oder eine Zeichnung mit Hilfe eines Computers bearbeiten zu können, muss es in eine dem Computer verständliche Form gebracht werden. Ebenso muss ein Fax bei dem Verschicken in elektrische Signale transferiert werden, bevor es den Weg durch die Telefonleitung zum Empfänger nimmt. Diesen Vorgang nennt man Digitalisierung.

1.1 Rasterung

Eine Schritt bei dieser Digitalisierung ist die Rasterung. Nahezu alle Geräte zur Bilderfassung wie Scanner, Faxgeräte oder auch Digitalkameras arbeiten mit dem Prinzip der Rasterung.
Das Ausgangsmaterial wird durch horizontale und vertikale Linien mit jeweils gleichem Abstand in Rechtecke aufgeteilt (sogenannte "picture elements - pixel"). Der Abstand der trennenden Linien wird als Auflösung bezeichnet und in Pixel pro Zoll ("dots per inch" - dpi) angegeben.



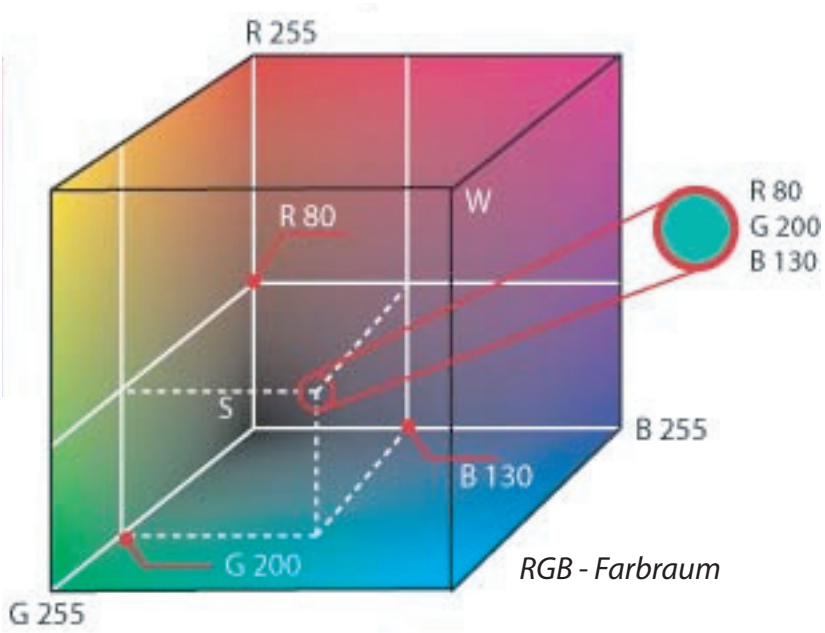
Im obigen Beispiel wird das Bild in ein 5x5 Raster aufgeteilt, wir erhalten 25 Pixel mit mit der Farbinformation schwarz oder weiss, je nachdem welche Farbe in dem jeweiligen Pixel überwiegt. Mehr Details zu Farbinformation eines Pixels in Abschnitt 1.2 .
Eine Erhöhung der Auflösung hat unmittelbar auch eine Erhöhung der resultierenden Datenmenge zur Folge.



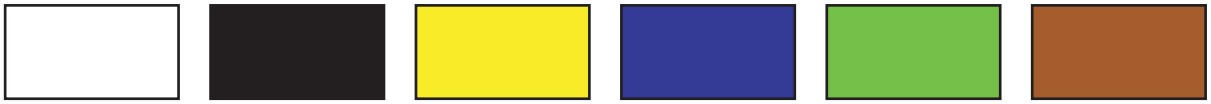
Eine Verdoppelung der Auflösung wie im zweiten Beispiel (10 x 10 Raster) ergibt 100 Pixel, also das Vierfache verglichen mit dem ersten Beispiel. Natürlich ist damit auch eine wesentlich feinere Repräsentation des Originals verbunden.
Bemerkung: Horizontale und vertikale Auflösungen können auch voneinander abweichen, beispielsweise bei Videomaterial (4:3 PAR "pixel aspect ratio").

1.2 Farberfassung

Die einfachen Beispielbilder hatten bis jetzt nur Schwarz und Weiss als Farben, die Farbinformation liesse sich also mit einem Bit pro Pixel beschreiben, also 25 Bit im ersten Beispiel, 100 Bit im zweiten Beispiel.
Die Darstellung einer Farbe geschieht mit Hilfe des additiven RGB-Farbraums. Diesen Farbraum kann man sich als einen dreidimensionalen Raum vorstellen, dessen Koordinaten jeweils den Farbanteil der Grundfarben Rot, Grün und Blau darstellen. Mit diesem Prinzip kann man nahezu alle für das menschliche Auge darstellbaren Farben erzeugen. Auf weitere Details dieses Farbraums möchte ich hier nicht eingehen, relevant für uns ist hier nur die Darstellung einer Farbe mittels der Grundfarbanteile.



RGB - Farbraum



(255,255,255) (0,0,0) (255,255,0) (0,0,255) (107,186,38) (159,62,14)

Einige Farben und die dazugehörigen RGB-Werte

Bei der Digitalisierung wird jeder Farbkanal mit einer Auflösung von 8 Bit quantisiert, damit sind 2^{24} verschiedene Farben realisierbar, man spricht von einer Farbtiefe von 24 Bit pro Pixel (zur Verdeutlichung der Farbmischung in einem RGB-Farbraum: http://www.wackerart.de/mixer.html). Der RGB-Farbraum umfasst mehr Farben als das menschliche Auge zu unterscheiden vermag. Der Informationsverlust bei der Quantisierung der Farbkanäle ist somit irrelevant.

Zusammenfassung:
Ist das Bildmaterial digitalisiert, liegt es in Rohdaten vor. Im obigen Beispielbild haben wir jetzt zu jedem Pixel einen X- und Y-Wert und den dazugehörigen Farbwert, im Beispiel schwarz oder weiss.
Weiterhin haben wir Informationen über die Anzahl vertikaler und horizontaler Pixel, eventuell auch über die Auflösung. Die Farbinformation liegt zu jedem Pixel als Zahlentripel vor, dass die Rot-, Grün- und Blauanteile des Farbraums wiederspiegelt.
Die Farbdarstellung von einem Bit pro Pixel in der linken Tabelle ist im Grunde

Breite: 5 pixel
Höhe: 5 pixel
Auflösung: ? dpi

X	Y	Farbe
0	0	0
1	0	1
2	0	1
3	0	1
4	0	0
0	1	0
1	1	0
2	1	0
3	1	0
...	...	...

Wertetabelle mit Rohdaten aus dem grob gerasterten Bild des ersten Beispiels.

X	Y	Farbe
0	0	0 0 0
1	0	127 0 0
2	0	127 0 0
3	0	127 0 0
4	0	0 0 0
0	1	0 0 0
1	1	0 0 0
2	1	0 0 0
3	1	0 0 0
...	...	...

Wertetabelle mit RGB Informationen anstelle von Farbindexen

2. Codierung der digitalisierten Daten

Diese Daten gilt es nun geschickt anzuordnen, um die Handhabung des digitalen Bildmaterials dem Anwendungszweck anzupassen. Ein Programm zur Fotoretusche beispielsweise braucht möglicherweise schnellen Zugriff auf die Farbwerte eines jeden Pixels ohne dabei unnötige Positionsberechnungen auszuführen. Beim Speichern eines Bildes in einer Digitalkamera hingegen möchte man möglichst viele Bilder auf dem Speichermedium unterbringen, hier spielt die Dateigrösse eine wesentlich bedeutendere Rolle. Die verschiedenen Möglichkeiten der Anordnung dieser Rohdaten bezeichnet man als Codierung.

2.1 Rohdaten bzw. unkomprimiert

Nimmt man die Wertetabelle aus dem obigen Beispiel und reiht die Pixeldaten einfach aneinander, so erhält man ein sehr einfaches aber auch umständliches und speicherintensives Bildformat.
Die resultierende Datei könnte so aussehen:

```
0,0,0  1,0,1  2,0,1  3,0,1  4,0,0
0,1,0  1,1,0  2,1,0  3,1,0  4,1,1
0,2,0  1,2,1  2,2,1  3,2,1  4,2,1
0,3,1  1,3,0  2,3,0  3,3,0  4,3,1
0,4,1  1,4,1  2,4,1  3,4,1  4,4,1
```

Simple Datencodierung, leicht zu "lesen", aber speicherintensiv durch redundante Daten. Informationen (Breite, Höhe, Farbtiefe) müssen aus der kompletten Datei extrahiert werden

Wie man leicht sieht, könnte man die X und Y Positionen auch wegfallen lassen, da diese durch die Position des Datentripels innerhalb einer Zeile und der Zeilennummer dargestellt wird.

Beispiel: Das Public Bitmap Format (PBM)

Das PBM Format gibt es in drei Grundformaten, die sich in der Farbtiefe unterscheiden.
- Bitmap (PBM) schwarz/weiss 1 Bit
- Greymap (PGM) Graustufen bis 8 Bit
- Pixmap (PPM) TrueColor bis 24 Bit

Die Formate werden standardmässig in ASCII codiert, es besteht jedoch zusätzlich die Möglichkeit einer binären Codierung (Raw PBM).

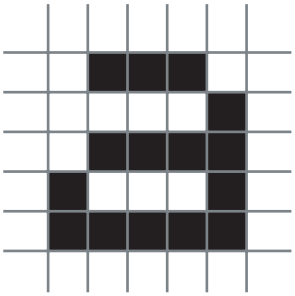
Datenstruktur vom PBM-Format:

Zeile	Inhalt
1	Text 'P1' | 'P2' | 'P3' | 'P4' | 'P5' | 'P6' (gibt Farbtiefe und Codierung an)
2	Breite, Höhe in Pixel
3	Farbtiefe in Bit (entfällt bei 'P1' / 'P4')
ab 4	Daten in ASCII bei P1 ... P3, Binär bei P4 ... P6

Kommentare sind erlaubt und werden mit '#' eingeleitet, Kommentarzeilen werden nicht gezählt.

Beispiel für eine PBM Datei:

```
# PBM-Typ: 1 Bit Farbtiefe ASCII
P1
# Breite und Hoehe
5 5
# Buchstabe a grob gerastert
0 1 1 1 0
0 0 0 0 1
0 1 1 1 1
1 0 0 0 1
1 1 1 1 1
```



Beispiel für eine PPM Datei:

```
# PBM-Typ: 24 Bit Farbtiefe ASCII
P3
# Breite und Hoehe
5 5
# Farbtiefe
24
# Pixeldaten, jeweils ein Zahlentripel pro Pixel
255 255 255  29  14 130  15  33 139  12  65 154 255 255 255
255 255 255 255 255 255 255 255 255 255 255 255   8  96 168
255 255 255 127 195  28 194 227  14 255 255   0   3 128 183
 64 166  41 255 255 255 255 255 255 255 255 255   0 160 198
  0 136 155  16 146  82   0 146 112   0 150 141   0 155 169
```

Die PBM Datei aus dem Beispiel hat eine Grösse von 53 Byte (der Dateikopf wurde weggelassen, denn mit steigender Pixelzahl wird der Anteil an der Gesamtgrösse vernachlässigbar gering), oder aber 16,96 Bit / Pixel.
Die PPM Datei (303 Byte ohne Kopf) erreicht hier einen Wert von 96,96 Bit / Pixel. Gegenüber der Rohdatenmenge (1 Bit / Pixel; 24 Bit / Pixel farbig) eine deutliche Verschlechterung.

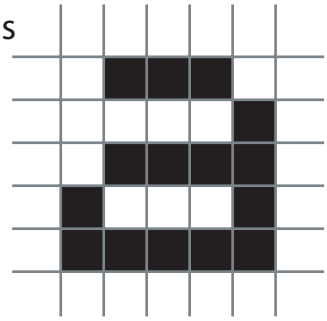
2.2 Datenkompression

Um das Datenvolumen der Rohinformationen zu reduzieren benutzt man verschiedene Kompressionsmethoden. Grundsätzlich muss zwischen zwei Methoden unterschieden werden. Bei der verlustfreien Datenkompression wird die Datenmenge durch geschickte Codierung der Daten vermindert, ohne dass ein Informationsverlust entsteht, das Bild ist beispielsweise nach der Dekompression identisch mit der digitalisierten Vorlage. Bei der verlustbehafteten Kompression ist dies nicht der Fall. Farbabweichungen und Unschärfe treten je nach Kompressionsgrad und Methode mehr oder weniger deutlich auf.

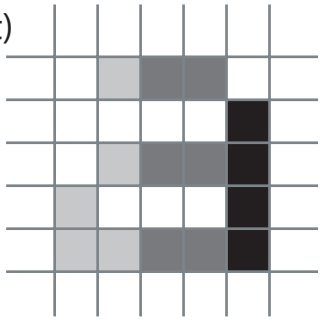
2.2.1 Verlustfreie Kompression mit RLE (Run Length Encoding)

Nach dem Prinzip der "Run Length Encoding"-Kompression schreibt man einzelne Pixel mit gleicher Farbe in der selben Zeile nicht separat, sondern gibt zuerst die Anzahl der aufeinander folgenden identischen Pixel, dann die Farbinformation dieser an.
Bei nur zwei Farben entfällt die Farbinformation, da logischerweise die Farben immer alternieren. Man geht davon aus, dass die erste Zahl in einer Zeile die weissen Pixel angibt. Die führende Null in Zeile 4 und 5 ist also notwendig, wenn die Zeile mit einen schwarzen Pixel beginnen soll.

Schwarz / weiss
1 3 1
4 1
1 4
0 1 3 1
0 5



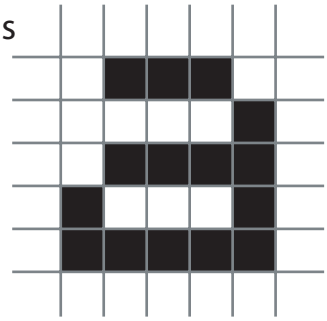
Graustufen (2 Bit)
1 3 1 2 2 1 1 3
4 3 1 0
1 3 1 2 2 1 1 0
1 2 3 3 1 0
2 2 2 1 1 0



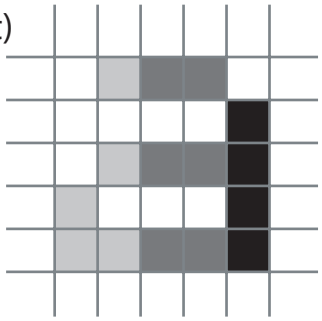
Die Kompressionsrate beim RLE hängt sowohl vom Ausgangsmaterial ab, als auch von der gewählten maximalen Länge von Pixelblöcken. In unserem Beispielbild von 5 x 5 Pixeln können maximal 5 gleiche Pixel aufeinander folgen, würde man dies als maximalen Zähler nehmen, müssten wir 3 Bit dafür reservieren. Natürlich wäre dann der maximale Zähler nicht 5 sondern 7 (Bei

dem Graustufenbild wären es maximal 8 Pixel, hier benötigen wir den Fall 0 Pixel nicht, da für jeden Block eine Farbinformation angegeben ist).
Das schwarz / weisse Bild könnten wir somit in 39 Bit (13 Pixelblöcke * 3 Bit Zähler) codieren, das Graustufenbild mit 80 Bit
(16 Pixelblöcke * (3 Bit Zähler + 2 Bit Farbinformation)).
Verringert man den maximalen Zähler auf 2 Bit, also maximal 3 aufeinander folgende Pixel (1 Bit / 2 aufeinander folgende Pixel beim Graustufenbild) wäre

Schwarz / weiss
1 3 1
3 0 1
1 3 0 1
0 1 3 1
0 3 0 2



Graustufen (2 Bit)
1 3 1 2 2 1 1 3
2 3 2 3 1 0
1 3 1 2 2 1 1 0
1 2 2 3 1 3 1 0
2 2 2 1 1 0



das Ergebnis wie folgt:

36 Bit für das schwarz / weisse Bild (18 Blöcke * 2 Bit), 54 Bit bei dem Graustufenbild (18 Blöcke * (1 Bit Zähler + 2 Bit Farbinformation)).
Vergleicht man dieses Beispiel mit dem Rohdatenvolumen (25 Bit schwarz / weiss, bzw. 50 Bit Graustufen), ist man zunächst vielleicht ein wenig enttäuscht, man muss sich allerdings vor Augen halten, dass mit steigender Pixelzahl das Verfahren effizienter wird.
Ein Faxgerät liest eine DIN A4 Seite mit 1728 x 1143 Pixel (horizontal x vertikal) mit einem Bit Farbtiefe, die Rohdatenmenge ist ca. 241kByte. Nach RLE-Codierung ergibt sich eine Datenmenge von ca. 40 - 120 kByte, abhängig von dem Bildinhalt.

In unseren Beispielen werden Pixelanzahl und Farbwert in Zahlenpaaren gespeichert. Die eigentlichen RGB-Werte werden per Farbpalette zugeordnet. Farbpaletten müssen nicht unbedingt in der Datei selbst gespeichert werden, sie können auch vom Betriebssystem zur Verfügung gestellt werden. Bei 24 Bit Farbtiefe liegen allerdings die Rot-, Grün- und Blauwerte mit je 8 Bit direkt vor, dass keine Farbtabelle mehr nötig ist.

Beispiel: Das Format BMP

Bei dem BMP Format handelt es sich um ein einfaches Bildformat, das in der Windowswelt viel Verbreitung findet.

Eine BMP Datei besteht aus drei Teilen:
- dem Dateikopf (Header)
- der Bitmap Information mit Angaben zur Dimension, Farbtiefe, Kompression ...
- dem Datenblock

Übersicht Dateikopf und Bitmapinfo einer BMP Datei

Grösse	Name	Inhalt
HEADER		
2 Byte	bfType	"BM"
4 Byte	bfSize	Dateilänge in Byte
2 x 2 Byte		Reserviert, 0
4 Byte		Offset Datenblock
BITMAP INFO		
4 Byte	biSize	Länge der Info
2 x 4 Byte	biWidth, biHeight	Dimension in Pixel
2 Byte	biPlanes	Farbtiefe
2 Byte	biBitCnt	Bit / Pixel
4 Byte	biCompr	Kompressionstyp
4 Byte	biSize	Bildgrösse in Byte
2 x 4 Byte	biXPels, biYPels	Auflösung horizontal / vertikal
2 x 4 Byte	biClrUsed, biClrImp	Zahl benutzter / wichtiger Farben
DATEN n * 4 Byte, ein Byte als Zähler, dann jeweils 1 Byte für Rot, Grün, Blau (Eine von drei möglichen Datenformaten, abhängig von der Kompressionsmethode und Farbtiefe)		

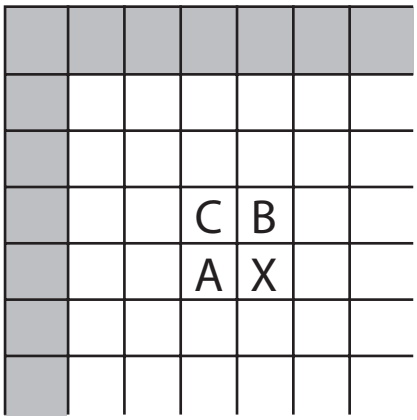
```
42 4D 26 04 00 00 00 00  00 00 36 00 00 00 28 00
00 00 12 00 00 00 12 00  00 00 01 00 18 00 00 00
00 00 F0 03 00 00 EB 0A  00 00 EB 0A 00 00 00 00
00 00 00 00 00 00 0A FF  FF FF 08 FF FF 00 00 00
09 FF FF FF 09 FF FF 00  00 00 08 FF FF FF 0A FF
AA 04 00 00 .. .. .. ..
```

Anfang einer BMP- Datei. Ab '0A FF FF FF' in Zeile 4 beginnen die Pixelinformationen in RLE Codierung

Bemerkung:
Nicht alle Bilder eignen sich zur RLE-Kompression.
Ein "Schachbrett" mit einer Feldgrösse von einem Pixel ist denkbar ungünstig zu komprimieren, da es faktisch keine aufeinanderfolgenden Pixel mit selbem Farbwert gibt.

2.2.2 Verlustfreie Kompression mit "Color Prediction"

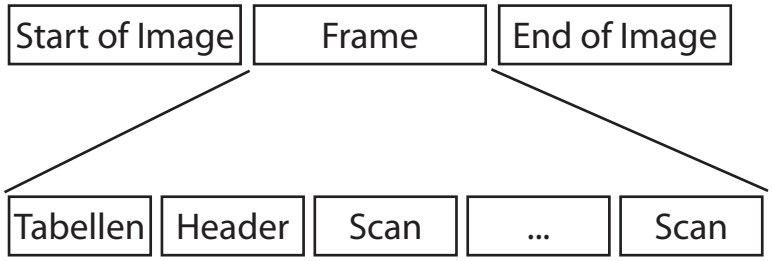
Dieses Verfahren wurde 1991 - 1993 von der Joint Photographers Expert Group entwickelt. Das Verfahren beruht auf der Annahme, dass benachbarte Pixel eines Bildes ähnliche Farbwerte besitzen. Im Gegensatz zu der RLE Kompression können hier nicht nur das linke Pixel in der selben Zeile mit gleichem Farbwert als Informationsbasis genutzt werden, sondern auch ähnlichfarbige in der Zeile darüber. Mit einem von 7 möglichen Algorithmen wird der Farbwert des Pixels aus den Farbwerten der benachbarten Pixel vorhergesagt, für das Pixel wird die Nummer des Algorithmus ("Predictor"), der dem tatsächlichen Farbwert am nächsten kommt und die Abweichung von diesem Farbwert gespeichert.
(Das linke obere Pixel enthält immer den Predictor 0, da hier keine Vorhersage des Farbwerts möglich ist. Für den linken und den oberen Rand kommt prinzipbedingt auch nur Predictor 2 bzw. 1 in Betracht.)



Predictor / Nr	Algorithmus
0	---
1	= A
2	= B
3	= C
4	= A +B - C
5	= A + (B - C) / 2
6	= B + (A - C) / 2
7	= (A + B) / 2

Die Flexibilität der 7 Algorithmen sind die meisten Farbabweichungen vor der Vorhersage durch die Algorithmen nahe Null, sie werden deshalb mit Hilfe der Huffman-Codierung oder einer arithmetischen Codierung sehr effizient komprimiert. Jeder Farbkanal wird separat codiert.
Die verlustfreie Kompressionsmethode ermöglicht Einsparungen beim Datenvolumen von bis ca 50%.

Beispiel: Das JPG-Format (verlustfrei)



Dateistruktur einer JPG-Datei mit verlustfreier Kompression

Die einzelnen Farbkanäle werden beim JPG-Format als Scan bezeichnet und einfach hintereinandergereiht. Im Headerbereich stehen Informationen über Dimension, Farbmodell, Auflösung, weiterhin Angaben zu Copyright oder Kommentare. Im Tabellenbereich steht im wesentlichen die Tabelle mit Huffman Codes (Huffman ist ein Verfahren zur Datenkompression mittels einer tabellarischen Zuordnung). Jeder Scan kann auch seine eigene Huffman-Tabelle haben.
Innerhalb des Scans finden sich dann die Pixeldaten (Nummer des Algorithmus und Abweichung) Huffman-codiert wieder.

2.2.3 Kompression mit Farbtiefenverlust

Eine andere Möglichkeit der Kompression ist die Anzahl der möglichen Farben zu veringern. Besonders bei Bildern mit einer dominierenden Farbe, beispielsweise Fotos mit Sepiatönung, bietet sich diese Methode sich an.
In einer Farbtabelle werden die einzelnen Farben mit ihren RGB-Werten aufgelistet und mit einem Index versehen. Bei einer maximalen anzahl von 32 Farben (aus dem kompletten RGB Farbraum!) im Bild werden pro Pixel nur 5 Bit für den Farbindex gebraucht.

links: Original, 971kByte ; mitte: GIF Kompression 8 Farben, 50 kByte ; rechts: GIF Kompression 2 Farben, 11 kByte

Eine mögliche Codierung wäre die Folgende

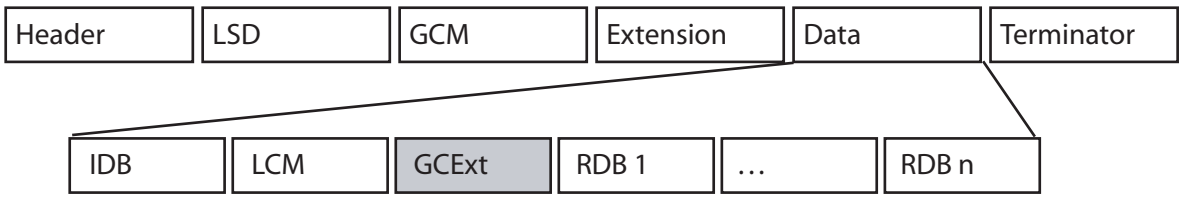
```
#Farbtabelle (Farbindex Rot Grün Blau)
 0 FF FF FF
 1 99 00 00
 2 A9 00 00
 3 B9 00 00
 ...
#Bilddaten mit RLE (Zähler Farbindex)
 1 0 1 1 2 2 1 0
 4 0 1 3
 1 0 1 1 2 2 1 3
 ...
```

Beispiel: Das GIF Format (Graphics Interchange Format)

Seit der Aufnahme in die HTML-Spezifikation hat das von UNISYS und Compuserve 1987 definierte GIF Format sich stark verbreitet. Grundsätzlich gibt es zwei Varianten, GIF87a und GIF89a. Letztere besitzt die Möglichkeit mehrere Einzelbilder als Animation zu speichern.

Im GIF Format kommen eine oder mehrere Farbtabellen zum Einsatz, eine Pixel/Pixelblockfarbe wird mit einem Index aus der Farbtabelle angegeben. Das GIF Format kann maximal 256 Farben per Farbtabelle verwalten, folglich tritt eine Farbtiefenreduktion auf. Statt 24 Bit / Pixel wird hier nur ein Farbindex von 8 Bit pro Pixel gespeichert. Weil bei dieser Kompression die Bildschärfe keine Verluste erleidet, eignet sich das GIF Format besonders zum Speichern von typografischen Elementen und Zeichnungen.

Weiterhin kennt das GIF Format einen Interlace Modus, in dem die einzelnen Zeilen nach Art eines binären Baums angeordnet werden. Das Resultat ist ein Bild, das bei der Anzeige zuerst grob gerastert, dann immer feiner dargestellt wird.



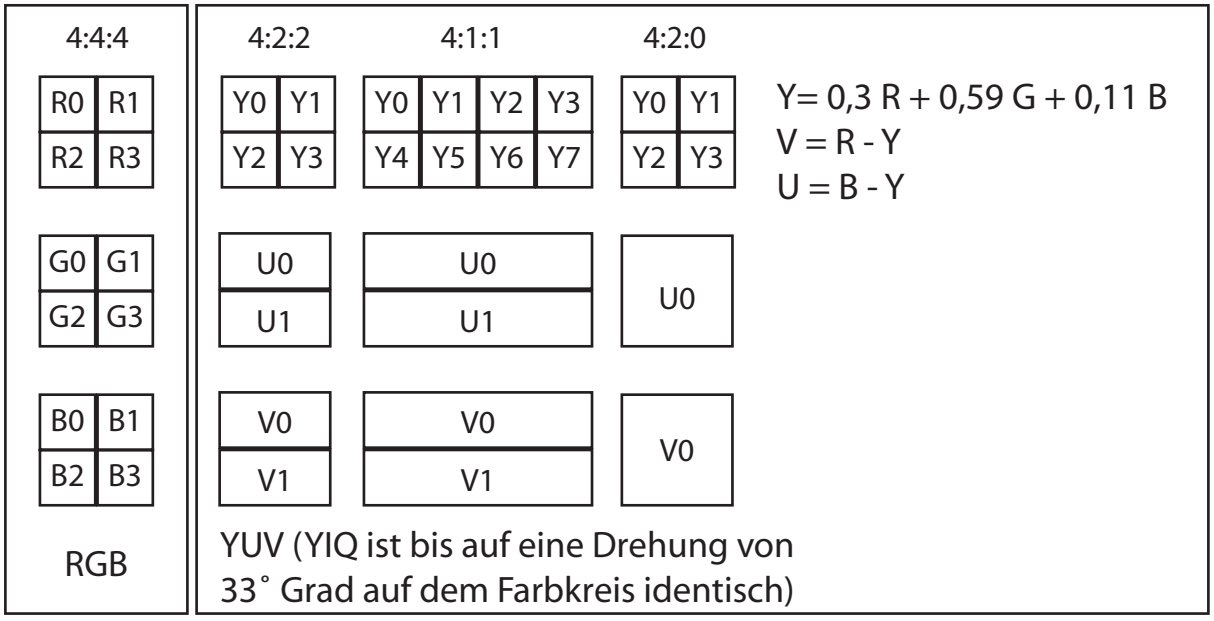
Grösse	Name	Inhalt
HEADER		
3 Byte	Signatur	"GIF"
3 Byte	Version	"87a" | "89a"
1 Byte	Terminator	0x3B
LSD (Logical Screen Descriptor)		
2 x 2 Byte	Logical Screen Width and Height	Breite und Höhe in Pixel
1 Byte	Resolution Flag	Bit 0..2 : Bit/Pixel Bit 3 : Sort Flag (89a) Bit 4..6 : Fartiefe Bit 7 : Global Color Map Flag
1 Byte	Background Color Index	Index für die Hintergrundfarbe
1 Byte	Pixel Aspect Ratio	Pixelseiten-verhältnis (87a)

Datenstruktur und Blockübersicht einer GIF Datei

Grösse	Name	Inhalt
GCM (Global Color Map)		
3 Byte * n		max. 256 True Color Farben
Extension		
1 Byte	Header	0x21
1 Byte	Function Code	"87a" | "89a"
n * 1 + x + 1 Byte	Length + Data + Terminator	Wiederholbar: 1 Byte Länge in Byte, x Datenbytes, 0x00 als Terminator
IDB (Image Descriptor Block)		
1 Byte	Separator Header	0x2C
2 + 2 Byte	Left + Top Alignment	Koordinate des folgenden Teilbildes
1 Byte	Flags	Bit 0..2 : Pixel Size Bit 5 : Sorted Flag Bit 6 : Interlace Flag Bit 7 : Local Color Map Flag
LCM (Local Color Map) wie GCM		
GCExt (Graphics Control Extension Block)		
8 Byte	Verschiedene Einstellungen für Animation in GIF 89a	
RDB (Raster Data Block)		
1 Byte	Code Size	Codelänge für LZW
1 Byte	Length	Datenlänge in Byte
n Byte	Data	Indizes in Farbtabelle
1 Byte	Block Terminator	0x00

2.2.4 Verlustbehaftete Kompression mit Fouriertransformation

Darstellung am Beispiel des JPEG Formats.
Bei dieser Kompressionsmethode geht ein optionaler Wechsel des Farbmodells von RGB nach YIQ ("Luminance InPhase Quadrature") voran. Da das menschliche Auge empfindlicher in der Helligkeitswahrnehmung als in der Farbwahrnehmung ist, verbessert man durch diese Trennung von Helligkeit und Farbe die Kompression ohne dass die subjektive Bildqualität darunter leidet.
Die Helligkeit (Luminanz = Y) wird für jedes Pixel gespeichert, die beiden Differenzen (Farbton und Sättigung) nur in Blöcken zu vier Pixeln (Farb-Subsampling nach Methode 4:2:0). Im weiteren Prozess werden die Farbkanäle separat behandelt.



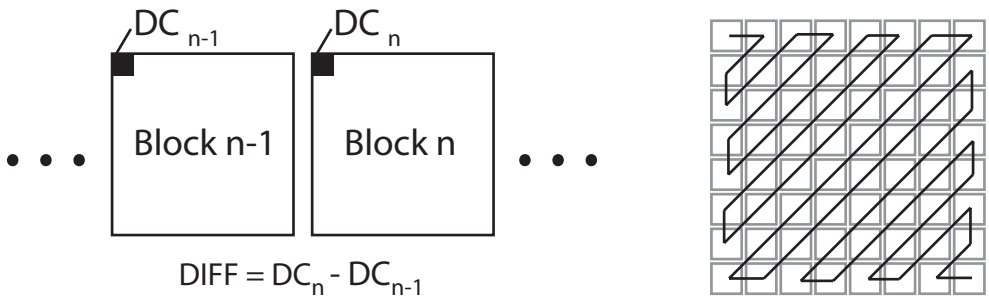
Verschiedene Möglichkeiten beim Farb-Subsampling

Nach dieser Farbreduzierung wird das Bild in sogenannte Makroblöcke zu je 8x8 Pixel eingeteilt. Für jeden Makroblock wird die Luminanz mit Hilfe einer Fourier Transformation (Discrete Cosine Transformation / DCT) in den Frequenzraum übertragen. Man erhält einen Satz von 8 x 8 Koeffizienten je Makroblock.
Bis hier ist abgesehen vom Farb-Subsampling noch keine Komprimierung eingetreten, die Daten wurden auf die Komprimierung vorbereitet.
Im eigentlich verlustbehafteten Prozess der Quantisierung werden die 64 DCT-Koeffizienten nun dividiert und das Ergebnis auf eine ganze Zahl gerundet. Die Division geschieht anhand einer Quantisierungstabelle, die die Farb- und Helligkeitsempfindlichkeit des Auges berücksichtigt. Das menschliche Auge reagiert auf niedrige Frequenzen viel empfindlicher als auf hohe. Die

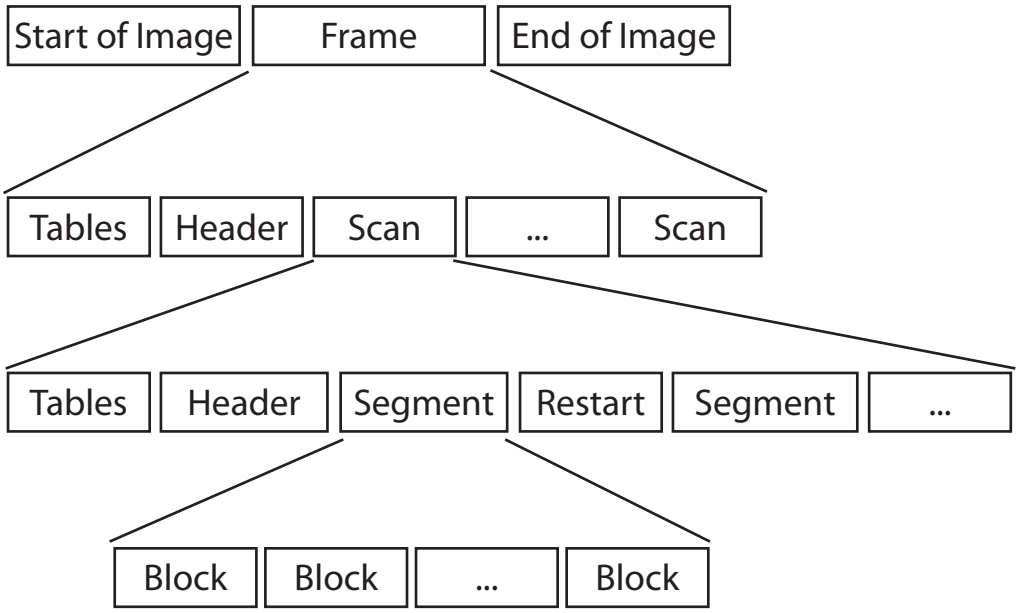
Zunehmend sichtbare Informationsverluste bei der Quantisierung der DCT-Koeffizienten

Koeffizienten höherer Frequenzen werden deshalb durch höhere Zahlen geteilt als die der niedrigeren. Hiermit erreicht man, dass sich der Wertebereich verkleinert. Viele Koeffizienten werden auch zu Null, was sich sehr günstig auf die später folgende RLE-Codierung auswirkt.
Die Art der Quantisierung ist nicht fest vorgeschrieben, es ist auch möglich anstelle der Standard-Quantisierungstabelle eine eigene anzulegen, die dann im Header eines jeden Scans gespeichert wird.

Der obere linke Koeffizient eines Makroblocks (DC - Koeffizient) gibt die mittlere Helligkeit des Makroblocks bezüglich des Farbkanals wieder. Da diese Werte sehr gross sind wird nur die Differenz zum vorhergehenden Makroblock gespeichert. Die weiteren Koeffizienten werden in einer Zick-Zack-Serialisierung gespeichert, dabei erfolgt eine RLE-Codierung. Die erhaltenen Werte werden letztendlich nach Huffman-Codierung gespeichert.



Die Differenz der DC-Koeffizienten ist Beginn der Zick-Zack-Serialisierung aller Koeffizienten eines Makroblocks



Dateistruktur einer JPG-Datei

Typisch für das JPG-Format sind Kompressionsfaktoren von 10 - 50, abhängig vom Bildmaterial. Besonders gut lassen sich Bilder mit weichen Farbübergängen komprimieren, dort fällt der bedingte Schärfeverlust nicht auf.

Verwendete Literatur:

Peter A. Henning - Taschenbuch Multimedia
Fachbuchverlag Leipzig
ISBN 3-446-21274-4

http://www.daubnet.com/formats/index.html

http://www.w3.org/Graphics/GIF/spec-gif89a.txt

http://de.wikipedia.org/wiki/Fax

http://de.wikipedia.org/wiki/RGB-Farbraum

http://de.wikipedia.org/wiki/Diskrete_Kosinustransformation

http://goethe.ira.uka.de/seminare/redundanz/vortrag11/