Seminar „Informatik für Biologen“

Datenkompression

Andreas Wieland

Universität Ulm WS2004/2005

Inhaltsverzeichnis:

1. Einleitung 1
2. Allgemeine, verlustfreie Verfahren..... 2
2.1 Wiederholungsbasierte Kompressionsverfahren 2
2.1.1 Word Coding..... 2
2.1.2 Lauflängen-Codierung (RunLengthEncoding = RLE) 3
2.1.3 LZ77-Verfahren (Lempel-Ziv-Verfahren von 1977)..... 4
2.2 Häufigkeitsbasierte Kompressionsverfahren 5
2.2.1 Huffmann-Codierung 5
2.3 Anwendungsspezifische, verlustfreie Verfahren 7
3. Anwendungsspezifische, verlustbehaftete Verfahren..... 7
3.1 JPEG-Verfahren 8
3.2 Fraktale Bildkompression 10
3.3 MP3 (MPEG-Audio-Layer 3)..... 10
3.4 Videokompression 12
4. Quellen 12

1. Einleitung

In Zeiten steigender Speicherkapazität (Festplatten mit über 250 GB) und immer schnellerer Datenübertragungsmethoden (DSL mit über 1Mbit/s) stellt sich die Frage, ob die zum Teil sehr rechenaufwendige und komplexe Kompression von Daten überhaupt noch notwendig ist. Betrachtet man jedoch die Größe unkomprimierter Daten wird schnell klar, dass trotz dieser steigenden Kapazitäten vor allem bei Audio- und Videodaten auf eine effektive Kompression nicht verzichtet werden kann. Das weit verbreitete MP3-Format verschaffte den Anwendern die Möglichkeit eine riesige Musiksammlung auf ihrem Rechner zu verwalten/speichern. Aufgrund der großen Kompressionseffektivität bei annehmbarer Qualität ist es möglich die 10-fache Menge an Musik bei gleichem Platzverbrauch zu speichern. Das Kompressionsformat erlangte auch die (negative) Aufmerksamkeit der breiten Öffentlichkeit durch die rasante Verbreitung dieser MP3-Files in den illegalen Tauschbörsen wie Napster, Kazaa, …. Analog zur illegalen Verbreitung von Musikdateien im Internet sind durch die verbesserten Videokompressionsverfahren (DivX, …) und die erhöhten Übertragungsraten durch Breitbandanschlüsse (DSL) in letzter Zeit auch ganze Filme in fast perfekter Qualität z.T. vor dem Kinostart erhältlich.

Die angewandten Kompressionsverfahren lassen sich aufgrund ihrer Verfahrensweise in 2 Hauptgruppen einteilen:

Allgemeine, verlustfreie Verfahren, die (mit wenigen Ausnahmen) unabhängig vom vorliegenden Datentyp angewendet werden können, arbeiten wie der Name schon sagt verlustfrei und können die Ausgangsdatei 1:1 wiederherstellen. Diese Verfahren werden in den bekannten Packprogrammen wie WinZip, 7Zip, WinRar, … verwendet. Allerdings ist es möglich, dass diese Verfahren manche Daten nicht komprimieren können und die durch den Kompressionsvorgang hinzugefügten Kompressionsinformationen die Dateigröße sogar vergrößern.

Anwendungsspezifische, verlustbehaftete Verfahren, die für bestimmte Datentypen (z.B. Video, Audio, Bilder) speziell entwickelt wurden, komprimieren nur die für sie bestimmten Datentypen sinnvoll, da bestimmte vom Menschen kaum wahrgenommene Details entfernt werden. Aus einer mit diesen Verfahren komprimierten Datei kann aufgrund der Entfernung „unwichtiger Information“ die Ausgangsdatei nicht mehr exakt wiederhergestellt werden.

2. Allgemeine, verlustfreie Verfahren

Die allgemeinen, verlustfreien Verfahren nutzen das Vorkommen von sog. Redundanzen, die in der Informatik als unnötige Informationen definiert werden, da sie in der „Nachricht“ vorher bereits genannt bzw. gespeichert wurden. Solche Redundanzen lassen sich auf verschiedenen Wegen erkennen und folglich die Datenmenge erheblich verringern. Die Methoden zur Suche nach Redundanzen unterteilt die allgemeinen, verlustfreien Verfahren in 2 Gruppen. Die erste Gruppe nutzt die Wiederholung von einzelnen Zeichen bzw. von ganzen Sequenzen (Zeichenkombinationen) in Dateien. Die zweite Gruppe ermittelt Redundanzen über die Häufigkeitsverteilung der einzelnen Zeichen einer Datei. Im Folgenden soll auf einige Verfahren aus beiden Gruppen eingegangen werden.

2.1 Wiederholungsbasierte Kompressionsverfahren

2.1.1 Word Coding

Eines der einfachsten der wiederholungsbasierten Kompressionsverfahren stellt das sog. Word Coding dar. Dabei werden die einzelnen in einer Datei vorkommenden

„Worte“ in einer Liste gespeichert und die „Wörter“ in der Ursprungsdatei durch Verweise auf die entsprechenden Listenplätze gespeichert. Wie schon durch die oben verwendete Formulierung der „Datenstruktur“ des Ausgangsmaterials klar wird, funktioniert diese Verfahren am schnellsten und einfachsten bei Textdateien, da hier die für das Verfahren notwendige Existenz von einzelnen Worten durch die Leerzeichen bewerkstelligt wird. Das genaue Verfahren des Word Coding wird im folgenden Beispiel verdeutlicht.

Ausgangsdatei: to be or not to be

➔ Zuerst wird eine Liste mit den vorkommenden Worten angelegt.

Liste: 1: to
2: be
3: or
4: not

➔ Die vorliegende Datei besteht also effektiv nur aus 4 Worten.

Der Encoder würde nun lediglich die „Listennummern“ speichern und zusätzlich für die spätere Rekonstruktion der Datei die erstellte Liste Anhängen.

Komprimierte Datei: 1, 2, 3, 4, 1, 2 + erstellte Liste

Es zeigt sich, dass der verwendete Zahlencode wesentlich kürzer gespeichert werden kann. So sind für eine einzelne Zahl nur 2 Bit notwendig, wohingegen zur Speicherung des gesamten Satzes in Normalform schon pro Zeichen 8 Byte benötigt werden. Der Einwand, dass zudem auch die Wortliste gespeichert werden muss, ist bei diesem kurzen Beispiel zwar berechtigt, doch ist deutlich zu erkennen, dass dieses Verfahren bei längeren Texten, in denen viele Wortwiederholungen auftreten, äußerst effektiv ist. Allerdings muss die Einteilung zu den allgemeinen, verlustfreien Verfahren etwas eingeschränkt werden, da klar wird, dass dieses Verfahren fast nur bei Textdateien effektiv arbeiten kann.

2.1.2 Lauflängen-Codierung (RunLengthEncoding = RLE)

Das Verfahren der Lauflängen-Codierung (RLE) basiert auf der Suche nach Wiederholungen einzelner Zeichen. Dabei werden die Anzahl der Wiederholungen und der Zeichenwert als Wertepaar gespeichert. Das Ergebnis der RLE zeigt sich in folgendem Beispiel, das eine Zeile einer einfachen Schwarz-Weiß-Skizze darstellen soll.

Ausgangsdatei: WWWWSSSWWSSWWWWWWSWWW
Komprimierte Datei: 4x W; 3x S; 2x W; 2x S; 6x W; S; 3x W

Es ist zu erkennen, dass durch dieses Verfahren eine deutliche Reduktion der Datenmenge erreicht wurde. Zudem ist das Verfahren sehr einfach und folglich sehr schnell. Dieses Verfahren wird zum Beispiel für einfache Bildkompressionen eingesetzt. Auch die unter Windows verwendeten Bitmap-Bilder können RLE-komprimiert

werden. Allerdings ist dieses Verfahren aufgrund seiner Einfachheit bei komplexeren Bildern mit feinen Farbabstufungen recht uneffektiv, da es auf die genaue Übereinstimmung zweier nebeneinander liegender Pixel angewiesen ist. Bei Schwarz-Weiß-Skizzen ist dieses Verfahren sehr effektiv, da nur zwei Farbzustände vorkommen. Bei 8-Bit-Graustufenbildern kommen hingegen schon 256 Grautöne vor, wobei die Wahrscheinlichkeit, dass zwei oder mehr Nachbarpixel den gleichen Farbwert besitzen deutlich abnimmt. Zudem sind für eine wirkliche Kompression mindestens 3 identische Pixel notwendig, da die Information in einem Wertepaar gespeichert wird.
In diesem Zusammenhang soll schon einmal erwähnt werden, dass eine etwas abgewandelte Form der RLE einen Teil des JPEG-Verfahren bildet.

2.1.3 LZ77-Verfahren (Lempel-Ziv-Verfahren von 1977)

Das LZ77-Verfahren stellt ein weiteres wiederholungsbasiertes Verfahren dar, das im Gegensatz zu dem vorgestellten RLE-Verfahren nicht nur die Wiederholung einzelner Zeichen, sondern die Wiederholung ganzer Sequenzen (Zeichenkombinationen) verarbeiten kann. Im folgenden kurzen Beispiel soll dieser Vorteil erläutert werden:

Ausgangsdatei:	ABABABABABAB	
RLE-kodiert:	ABABABABABAB	(➔keine Wiederholung von Zeichen)
LZ-kodiert:	6x AB	(Computerverfahren leicht abgeändert)

Das LZ77-Verfahren und das später entwickelte LZ78-Verfahren basieren auf dem Gedanken, dass die Wiederholung einfach durch einen Verweis auf die schon im Datenstrom gespeicherte Sequenz ersetzt wird.
Eine einfache Implementation bildet das LZSS-Verfahren. Dabei besitzt der Encoder einen dynamischen Lesepuffer, in dem eine gewisse Anzahl der eben eingelesenen Zeichen gespeichert wird und mit dem aktuellen „Zeichenfenster“ verglichen wird. Falls eine Übereinstimmung gefunden werden sollte, wird statt der Wiederholung einfach ein Wertepaar aus Entfernung der Ursprungssequenz zur Wiederholung und Länge des übereinstimmenden Bereichs gespeichert. Folgendes klassisches Beispiel zeigt deutlich die Arbeitsweise des Encoders.

Ausgangsdatei: MISSISSIPPI

➔ M wird eingelesen, in den Puffer geschrieben und gespeichert!
➔ I wird gelesen, mit dem Puffer verglichen, hineingeschrieben und gespeichert!
➔ S wird gelesen, mit dem Puffer verglichen, hineingeschrieben und gespeichert!
➔ S wird gelesen, mit dem Puffer verglichen
  ➔ Übereinstimmung von einem Zeichen
  ➔ Überprüfung des nächsten Zeichens liefert keine Übereinstimmung
  ➔ Schreiben in den Puffer und Speicherung

(S wird nicht ersetzt, da die Ersetzung durch einen Verweis aufwendiger wäre als die Speicherung des Ausgangszeichen)

➔ I wird eingelesen, mit dem Puffer verglichen

➔ Übereinstimmung mit Puffer

➔ Überprüfung des nächsten Zeichens ➔ Übereinstimmung

➔ Überprüfung des nächsten Zeichens ➔ Übereinstimmung

➔ „leerer Puffer“

➔ zweites „ISS“ wird durch Verweis (3,3) ersetzt

➔ folgende Zeichen werden ebenfalls mit Puffer verglichen, aber keine Übereinstimmungen gefunden

LZ-komprimierte Datei: MISS(3;3)IPPI

Das LZ77-Verfahren liefert bei den meisten Dateien eine gute Kompressionsleistung und wird in Kombination mit anderen Verfahren bei vielen klassischen Packprogrammen wie WinZip, PKZip, … verwendet. Die Weiterentwicklung LZ78 wird zum Beispiel beim TIFF-Format, das für die verlustfreie Kompression von Bildern verwendet wird, eingesetzt.

2.2 Häufigkeitsbasierte Kompressionsverfahren

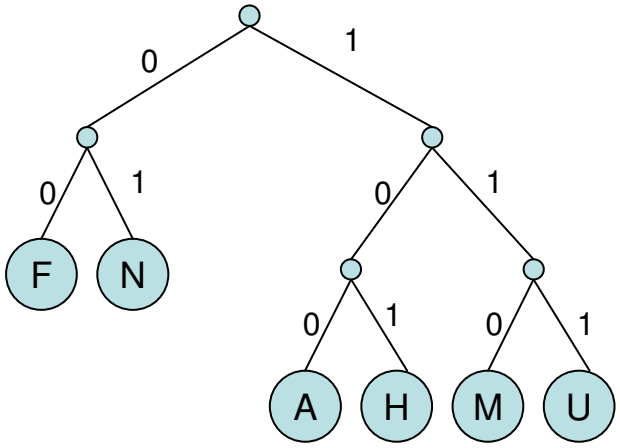
Die Häufigkeitsbasierten Kompressionsverfahren arbeiten mit der Zeichenhäufigkeit eines Zeichens in einer Datei. Untersucht man beispielsweise einen deutschen Text, so zeigt sich, dass die Buchstaben x, y und q recht selten vorkommen. Häufigkeitsbasierte Verfahren speichern nach der Ermittlung der Zeichenhäufigkeit nicht alle Zeichen einer Datei mit der gleichen Länge. So werden häufig benutzte Zeichen mit nur 2 Zeichen kodiert, wobei seltenere Zeichen mehrere Zeichen benötigen. Diese Idee ist zudem auch die Grundlage des Morse-Codes, der aber 3 Zustände (lang, kurz und Pause) kennt. Die folgende Huffmann-Codierung löst dieses Problem bezüglich des PC-Speichers, der nur 2 Zustände kennt.

2.2.1 Huffmann-Codierung

Die Huffmann-Codierung kann wie schon erwähnt als eine Weiterentwicklung und Übertragung des Morse-Codes auf „Bit-Strukturen (2 Zustände) betrachtet werden. Zu Beginn ermittelt der Encoder die Zeichenhäufigkeiten aller in der Datei vorkommenden Zeichen. Aufgrund dieser Daten erfolgt dann die Erstellung eines Binärbaums, wobei immer die beiden Knoten mit den niedrigsten Zeichenhäufigkeiten an einen gemeinsamen Vaterknoten gehängt werden, der als Wert die Summe der Zeichenhäufigkeiten seiner Tochterknoten trägt. Es wird bei diesem Verfahren mit allen Knoten analog verfahren bis sich alle Knoten in einem Baum mit einer Wurzel befin-

den, deren Zeichenhäufigkeit 100% ist. Der entstandene Baum kann nur zur Codierung der Datei verwendet werden. Die linken Äste des Baumes werden mit 0, die rechten Äste mit 1 bezeichnet, wodurch alle vorkommenden Zeichen durch eine bestimmte Sequenz von 0 und 1 codiert werden. Ein animiertes Beispiel zur Erstellung des folgenden Binärbaums ist in der zugehörigen Powerpointpräsentation zu finden. Im Folgenden soll nur kurz auf die ablaufende Codierung eingegangen werden.

Ausgangsdatei: Huffmann (8 Buchstaben ➔ 64 Bit)



Codierungstabelle:
F: 00
N: 01
A: 100
H: 101
M: 110
U: 110

Huffmann: ➔ 10111000001101000101

Zur Codierung des „häufig“ auftretenden Zeichens F benötigt der Encoder lediglich 2 Bit, da die Zeichen F und N aufgrund ihrer Häufigkeit sich in der ersten Ebene des Baumes befinden. Es zeigt sich, dass Zeichen, die mit gleicher Häufigkeit auftreten, sich in einer gemeinsamen Ebene befinden. Würde man nun das Wort Huffmann mit dem folgenden Baum codieren so würde der Name zu Speicherung nur 20 Bit belegen.

Durch diese Huffmann-Codierung konnte die Datenmenge im gewählten Beispiel auf ein Drittel der Ursprungsmenge reduziert werden. Dabei darf jedoch nicht vernachlässigt werden, dass der Baum zur anschließenden Decodierung ebenfalls mitgespeichert werden muss. Es gibt zudem die Möglichkeit, dass nur die Häufigkeitsverteilung mit der codierten Datei gespeichert wird, wobei allerdings sichergestellt werden muss, dass der Encoder und der Decoder aus dieser Häufigkeitsverteilung den gleichen Baum erzeugen.

Es wurde deutlich, dass die Huffmann-Codierung eine einfache und effektive Kompressionsmethode ist, die jedoch auf eine günstige Häufigkeitsverteilung der Zeichen angewiesen ist. Die optimale Kompression würde nur erreicht werden, wenn die Zeichenhäufigkeiten Zweierpotenzen wären und jedes Zeichen unterschiedlich oft auftreten würde. Dennoch wird das Huffmann-Verfahren in fast jedem klassischen Packprogramm als sog. Postprocessor verwendet und sogar beim JPEG-Verfahren wird Huffmann als abschließender Kompressionsschritt eingesetzt.

Eine Verbesserung der Effektivität im Hinblick auf die „Unabhängigkeit“ der vorkommenden Zeichenwahrscheinlichkeiten bietet die Arithmetische Codierung.

2.3 Anwendungsspezifische, verlustfreie Verfahren

In diesem Zusammenhang soll kurz auf die Möglichkeit der anwendungsspezifischen, verlustfreien Verfahren eingegangen werden. Ein Großteil dieser Verfahren basiert auf dem Prinzip der sog. Differential-Codierung, wobei kein absoluter Wert gespeichert wird, sondern lediglich die Differenz zum Vorherigen. Dieses Vorgehen soll an der Speicherung eines kleinen Bildes exemplarisch dargestellt werden.

16	18	19
15	17	18

Es ist deutlich, dass sich die Farbwerte der gezeigten Pixel nur geringfügig unterscheiden. Die Differential-Codierung würde anstatt der einzelnen Werte nur die Differenz zum vorherigen Wert speichern. Dies hätte zur Folge, dass der Großteil des „Bildes“ aus kleinen relativ ähnlichen Werten bestehen würde mit denen im Gegensatz zur Ausgangsdatenmenge eine effektive Huffmann-Codierung möglich wäre. Die Differential-Codierung selbst erbringt also kaum bzw. keine Reduktion des Datenmaterials. Erst durch die Verwendung eines geeigneten Postprocessor gelingt eine signifikante Reduktion des Datenmaterials.

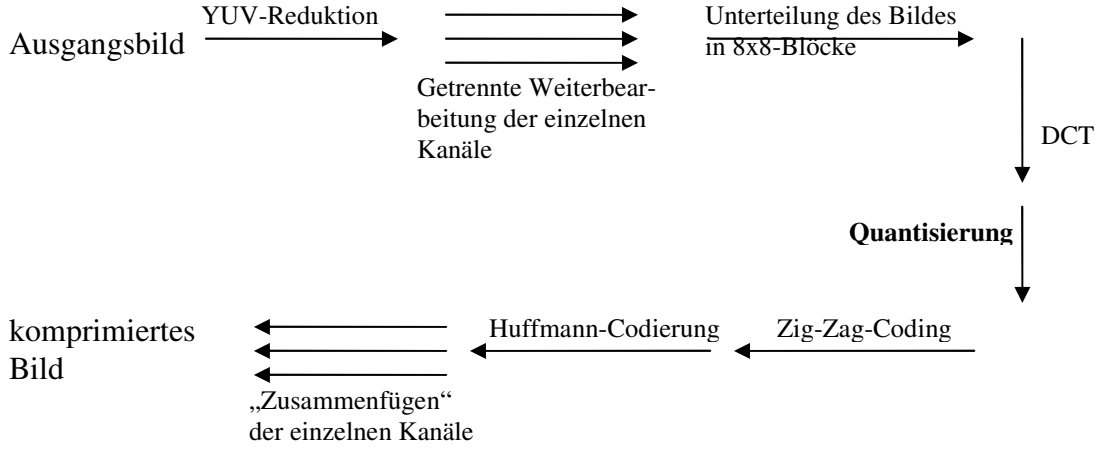
3. Anwendungsspezifische, verlustbehaftete Verfahren

Die anwendungsspezifischen, verlustbehafteten Verfahren wurden speziell für bestimmte Datentypen (z.B. Video, Audio, Bilder) entwickelt und garantieren durch die verlustbehaftete Kompression auf jeden Fall eine Reduktion der Datenmenge, wobei allerdings die Ausgangsdatei nicht mehr exakt wiederhergestellt werden kann. Die anwendungsspezifischen, verlustbehafteten Verfahren entfernen aus der Ausgangsdatei Informationen, deren Verlust dem Menschen kaum „ins Auge fällt“. Es werden bei den meisten dieser Verfahren also konsequent die Schwächen der menschlichen

Wahrnehmung ausgenutzt. Im Folgenden soll kurz auf die Verfahren zur Datenkompression bei Bild-, Audio- und Videodaten eingegangen werden.

3.1 JPEG-Verfahren

Das JPEG-Verfahren (Joint Photographic Expert Group) wurde Anfang der 90er von einer Expertengruppe entwickelt um ein leistungsfähiges Kompressionsverfahren für Bilder/Fotos zu entwickeln. JPEG stellt bis heute das am weitesten verbreitete Bildformat dar und ist im Internet allgegenwärtig.
Das JPEG-Verfahren ist eine Hintereinanderschaltung mehrerer Kompressions- und Transformationsverfahren. Die folgende Skizze zeigt das grobe Ablaufschema.



Farbbilder werden normalerweise mit Hilfe von drei Farbkanälen gespeichert (Rot, Grün, Blau), die als RGB-Format bezeichnet werden. Die optische Wahrnehmung des Menschen nimmt geringe Helligkeitsunterschiede sehr viel stärker war als minimale Farbunterschiede. Diese Gegebenheit führte zur „Entwicklung“ eines neuen Farbraums, dessen drei Kanäle als Y, U und V bezeichnet werden. Y stellt dabei den Helligkeitswert dar, der sich durch folgende empirische Formel aus dem RGB-Farbraum ableitet: $Y = 0{,}299\ R + 0{,}587\ G + 0{,}114\ B$
Die beiden anderen Kanäle enthalten die Farbinformationen als Differenz der Kanäle R und B von der Helligkeit. Der Grün-Faktor stellt in der oben berechneten Helligkeit ja bereits fast 60% dar. Die beiden anderen Teile werden folgendermaßen berechnet: $U = R - Y$

$V = B - Y$

Diese Umwandlung reduziert die anfallenden Daten jedoch nicht, sondern transformiert die Daten nur in eine bessere Form. Wie schon erwähnt reagiert das Auge we-

sentlich sensitiver auf Helligkeitsänderungen als auf Farbänderungen. Durch die „Entkopplung“ der Helligkeitsinformationen aus den drei Farbkanälen RGB in einen separaten Helligkeitskanal können die Farbkanäle wesentlich stärker (verlustbehaftet) komprimiert werden ohne dass die Helligkeit des Bildes beeinflusst wird (Die Kompression erfolgt durch eine Reduktion der Auflösung der beiden Farbkanäle.).

In den folgenden Schritten werden die einzelnen Kanäle getrennt voneinander bearbeitet. Im nächsten Schritt wird das Ausgangsbild in 8x8-große Blöcke (Abb.1) zerteilt. Die 64 Farbwerte dieser Blöcke werden nun mittels einer diskreten Cosinustransformation (DCT) untersucht und transformiert. Die transformierten Werte (Abb. 2) besitzen im Gegensatz zu den Ausgangswerten eine günstigere Beschaffenheit für die anschließenden Schritte. Im nächsten Schritt, der Quantisierung, werden die Werte des 8x8-großen Blocks durch die Werte einer ebenfalls 8x8-großen Quantisierungstabelle dividiert und die entstehenden Werte auf Ganzzahlen gerundet. Durch diese Prozedur werden die kleinen durch die DCT entstehenden Werte auf Null gerundet. Je kleiner die Werte der Quantisierungstabelle sind, desto besser die spätere Bildqualität (Übergabe der Qualitätseinstellungen des Benutzers an den Encoder). Es entsteht der in Abb. 3 dargestellte 8x8-große Block, der fast nur aus Nullen besteht. Der anschließende Schritt des Zig-Zag-Codings ist eine spezielle Form der RLE (verlustfrei!!). Hierbei werden die Werte nicht zeilenweise eingelesen, sondern wie der Name besagt in Zick-Zack-Form um die Anzahl sich wiederholender Nullen ohne Unterbrechung zu erhalten. Zur weiteren Verkleinerung der Datenmenge wird noch das Huffmann-Verfahren angewendet.

139	144	149	153	155	155	155	155
144	151	153	156	159	156	156	155
150	155	160	163	158	156	156	156
159	161	162	160	160	159	159	159
159	160	161	161	160	155	155	155
161	161	161	161	160	157	157	157
162	162	161	163	162	157	157	157
162	162	161	161	163	158	158	158

Abb.1: Farbwerte der Ausgangsdatei

235.6	-1.0	-12.1	-5.2	2.1	-1.7	-2.7	1.3
-22.6	-17.5	-6.2	-3.2	-2.9	-0.1	0.4	-1.2
-10.9	-9.3	-1.6	1.5	0.2	-0.9	-0.6	-0.1
-7.1	-1.9	0.2	1.5	0.9	-0.1	0.0	0.3
-0.6	-0.8	1.5	1.6	-0.1	-0.7	0.6	1.3
1.8	-0.2	1.6	-0.3	-0.8	1.5	1.0	-1.0
-1.3	-0.4	-0.3	-1.5	-0.5	1.7	1.1	-0.8
-2.6	1.6	-3.8	-1.8	1.9	1.2	-0.6	-0.4

Abb.2: Werte nach der DCT

240	0	-10	0	0	0	0	0
-24	-12	0	0	0	0	0	0
-14	-13	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

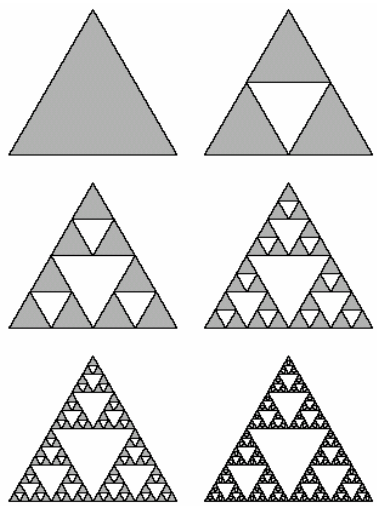
Abb.3: Werte nach der Quantisierung der DCT-Werte

Die Vorteile des JPEG-Verfahrens sind die effektive Kompression bei guter Bildqualität und die optionale Progressivität (das Bild wird nicht zeilenweise, sondern mit immer mehr Details aufgebaut), die zur Verbreitung im Internet sehr beigetragen hat. Dennoch hat das JPEG-Verfahren einige Mängel.

So entstehen durch die Verarbeitung der 8x8-großen Blöcke ohne Abgleich mit den Nachbarblöcken an den Blockgrenzen z.T. harte Übergange. Die Abschätzung der Größe des JPEG-Bildes ist aufgrund der abstrakten Parameter (es kann nur eine abstrakte Qualitätseinstellung an den Encoder übergeben werden) nicht möglich.

3.2 Fraktale Bildkompression

Die Grundidee der fraktalen Bildkompression ist, dass in einem Bild oft mehrere sehr ähnliche Objekte auftreten bzw. das ganze Bilder aus nur einem „Grundbaustein“ bestehen. Diese Grundbausteine könnten durch die Verwendung von „programmierten Linsen“ vergrößert, gedreht, gespiegelt und verschoben werden. Durch diese mit wenigen Daten (Linsenparameter) durchführbaren Aktionen könnte ein Bild sehr effektiv gespeichert werden. An dem folgenden Beispiel (Sierpinski-Dreieck) soll die Effektivität des Verfahrens demonstriert werden. Das rechte untere Dreieck kann durch die Verkleinerung und der Verschiebung der oberen Dreiecke erzeugt werden. Es ist verständlich, dass die Speicherung des linken oberen Dreiecks und ein paar „Linsenparametern“ sehr Platz sparend ist.



Dieses Verfahren bietet zwar gigantische Kompressionsraten ist aber aufgrund der enormen Rechenleistung zur Auffindung von Mustern nicht für den Heimgebrauch geeignet. Obwohl der Verbreitung im Internet nichts im Wege stehen würde, da dieses Verfahren sowohl optionale Progressivität als auch im Vergleich zur Kompression eine blitzschnelle Dekompression besitzt.

3.3 MP3 (MPEG-Audio-Layer 3)

Das wohl bekannteste Audiokompressionsverfahren wurde Anfang der 90er entwickelt und erreichte durch die massenhafte Verbreitung von MP3-Files in den Internettauschbörsen Ende der 90er einen hohen Bekanntheitsgrad. Das Verfahren beruht auf der Ausnutzung der Schwächen des menschlichen Ohres, die von der Psychoakustik untersucht werden. Im Folgenden soll kurz auf die Methoden zur Ausnutzung der Schwächen, die als Maskierung bezeichnet wird, eingegangen werden. Eine wichtige Maskierung bei der MP3-Kompression stellt die simultane oder zeitgleiche

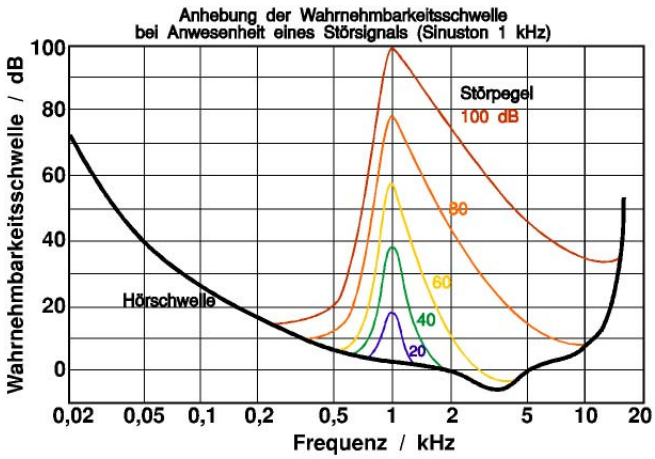


Abb.4: Hörschwelle des Menschen; www.wikipedia.de

Maskierung dar. Aus Abb.4 ist ersichtlich, dass ein Störsignal die Hörschwelle der „benachbarten“ Frequenzen deutlich beeinflusst. Durch diese Tatsache ist es dem Encoder möglich diejenigen Fre-quenzen, deren Schalldruck sich unterhalb der Hörschelle befindet, zu entfernen. Dadurch kann die eingehende Informationsmenge beträchtlich reduziert werden. Eine weitere Maskierungsart ergibt sich aus der physiologischen Begebenheit, dass ein lautes Geräusch die Wahrnehmung anderer Töne für eine gewisse Zeit

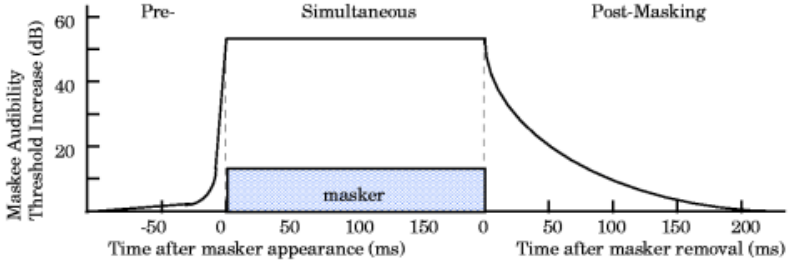


Abb.5: Phänomen der zeitlichen Maskierung

vor und nach seinem Auftreten verhindert (Abb.5). Diese Maskierungsart wird als zeitliche Maskierung bezeichnet. Sie ermöglicht die Entfernung „unhörbarer Zeitbereiche“ aus dem Eingangsmaterial. Die letzte und ebenfalls sehr effektive Maskierungsart wird als Stereo-Maskierung bezeichnet und berücksichtigt die Schwäche des menschlichen Gehörs bei der Ortung von Geräuschen. So können zum Beispiel tiefe Töne nicht geortet werden (Subwoofereffekt) und müssen folglich nicht in Stereo gespeichert werden. Des Weiteren unterscheiden sich die beiden Kanäle (links und rechts) in den meisten Fällen nur minimal, so dass eine Reduktion der Stereoinformation auf einen Kanal und die Speicherung eines zusätzlichen Seitensignals, das die Unterschiede zwischen den beiden Ursprungskanälen speichert, eine enorme Platzersparnis bietet (Joint-Stereo-Coding).

Das MP3-Kompressionsverfahren bietet eine Reihe von Vorteilen, die z.T. auch zu seiner rasanten Verbreitung führten. So kann man Audio-Daten um den Faktor 10 komprimieren und kaum Unterschiede zu CD-Qualität feststellen. Zudem ist es im Vergleich zum JPEG-Verfahren möglich eine konstante Kompressionsrate und somit die Endgröße genau festzulegen. Auch die Erweiterung der MP3-Kompression mit der Verwendung variabler Bitraten trug zur Steigerung der Sound-Qualität der MP3-

Files bei, da für komplexere Musikbereiche mehr Speicher zur Verfügung gestellt wurde, der in einfacheren Bereichen „eingespart“ werden konnte.

3.4 Videokompression

Bei der Kompression von Videodaten sind riesige Datenmengen zu bewältigen. Um eine wirklich effektive Reduktion der Datenmenge zu erreichen sind verlustbehaftete Verfahren eigentlich unumgänglich. Zur Speicherung von Videodaten existieren mehrere Ansätze. Eine Möglichkeit wäre die Speicherung von komprimierten Einzelbildern, was allerdings immer noch zu riesigen Datenmengen führt. Eine Methode, die gute Kompressionsraten ermöglicht ist, teilt die Einzelbilder (Frames) in sog. Keyframes, die vollständig gespeichert werden, und sog. Deltaframes, die nur den Unterschied zum nächsten Bild speichern (vgl. Differential-Codierung). Dieses Verfahren kann sogar durch die Speicherung einer zusätzlichen Vektorinformation über die Bewegung des Bildes noch verbessert werden. Durch das sog. Motion Compensation können Kameraschwenks kompensiert werden um die Datenmenge der Deltaframes noch einmal zu verringern. Dieses Verfahren ist ein Grundstein des MPEG-Formats (Motion Pictures Expert Group).
Die aktuellste Entwicklung der Videokompression ist der sog. DivX-Codec (MPEG-4), bei dem es sich um einen ursprünglich gehackten Video-Codec von Microsoft handelt, der im Laufe der Zeit allerdings völlig überarbeitet wurde. Bei diesem Verfahren werden die Bildinformationen- und Unterschiede nicht „pixelweise“, sondern „objektweise“ gespeichert. Der Encoder erkennt aufgrund von Farb- und Helligkeitswechseln Objektkanten und unterteilt das Bild in mehrere Objekte. Diese können dann aufgrund ihrer Eigenschaften mit unterschiedlicher Genauigkeit codiert werden. Ein Paradebeispiel liefert eine Nachrichtensendung. Dabei wird das Bild grob in drei Objekte unterteilt: Sprecher, Hintergrund und Tisch. Eventuell würde der Sprecher noch einmal in Brustkorb und Kopf unterteilt, da sich die meiste Bewegung in der Mimik des Sprechers abspielt. Tisch und Hintergrund können als praktisch als statische Objekte sehr Platz sparend gespeichert werden. Aber auch die Bewegungen einzelner Objekte können durch die Verwendung von Vektoren äußerst effizient komprimiert werden.

4. Quellen

http://www.wikipedia.de
Projektarbeit „Datenkompression“ von Martin Fiedler