4 Datenkompression

M. Dürst, M. Mühlhäuser

4.1 Grundlagen

4.1.1 Übersicht

Überträgt man Nachrichten (Daten) über Raum (Netze) und Zeit (Speichermedien), kann Datenkompression die Datenmenge bei Beibehaltung des Informationsgehalts stark reduzieren und Übertragungszeit und Speicherplatz einsparen. Speichermedien und Kommunikationskanäle werden immer preiswerter, aber gleichzeitig nehmen die Datenmengen und Prozessorleistungen zu, so daß Datenkompression weiterhin sinnvoll ist.

Nachrichten stammen aus einer *Quelle*, werden von einem *Sender* komprimiert, über einen *Kanal* geschickt und von einem *Empfänger* dekomprimiert (siehe Bild 1). Nachrichten können Daten beliebiger Art enthalten, z. B. Text, Bilder, Klänge usw. Allgemein spricht man von *Quellcodierung*, da die Eigenschaften der Quelle einen starken Einfluß auf die Kompressionsmethoden und die mögliche Stärke der Kompression haben. Im Gegensatz dazu befaßt sich die in Kapitel B2 behandelte *Kanalcodierung* mit der Erkennung und Vermeidung von Fehlern bei der Übermittlung über einen fehlerhaften Kanal.

Eine Nachricht aus einer diskreten Quelle wird als Folge von *Symbolen* aus einem *Quellalphabet* aufgefaßt. Diese werden durch den Sender in eine Folge von Symbolen aus einem *Kanalalphabet* abgebildet. Das Kanalalphabet ist meist das binäre Alphabet mit

den Symbolen 0 und 1. Der Empfänger kehrt die Abbildung des Senders um und rekonstruiert die ursprüngliche Nachricht.

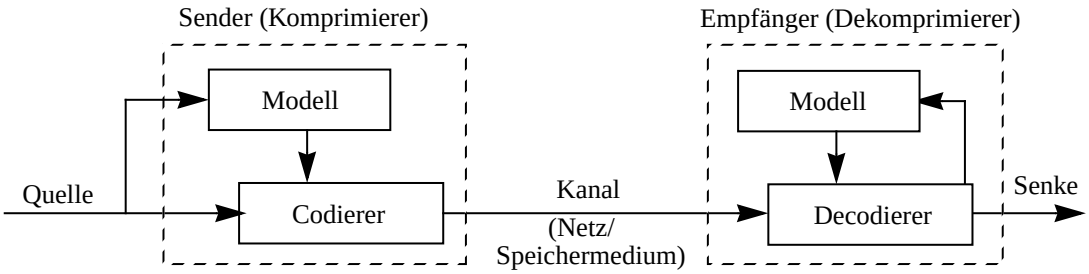

Bild 1 Strukturschema der Datenkompression

Kompression wird erreicht, indem man häufigen Nachrichten kürzere und selteneren Nachrichten längere Sequenzen von Kanalsymbolen zuordnet. Kompression ist damit nur im Durchschnitt über eine große Menge von Nachrichten garantiert; es können nicht alle möglichen Nachrichten komprimiert werden. Um die Stärke der Kompression anzugeben, wird das ursprüngliche Datenvolumen mit dem komprimierten in Beziehung gesetzt, aber es wird kein einheitliches Kompressionsmaß verwendet. Mit den Angaben „Kompression auf 25 %“, „Kompression um 75 %“ und „Kompression um den Faktor 4“ ist das gleiche gemeint. Hier wird als einheitliches Maß der Quotient aus ursprünglichem und komprimiertem Datenvolumen mit dem Namen „Kompressionsgrad“ belegt und verwendet.

Für die Datenkompression wurden vielfach Ad-Hoc-Methoden verwendet. Ein einfaches Beispiel ist die Ersetzung von mehreren Leerzeichen durch ein Tabulatorzeichen. Die Aufteilung von Sender und Empfänger in einen Modell- und einen Codierteil hat sich aber generell durchgesetzt (siehe Bild 1); auch wo diese Aufteilung nicht explizit vorhanden ist, erhöht sie das Verständnis der Zusammenhänge. Die Informationstheorie (siehe Kapitel B2.2) bildet die Grundlage für diese Aufteilung und für die Datenkompression generell. Die Modellierung wird in Abschnitt 4.1.2 besprochen, die Codierung in Abschnitt 4.2.1 und 4.2.2.

Bei vielen Daten aus dem Multimedia-Bereich verwendet man neben *verlustfreier* (*lossless*) Kompression auch *verlustbehaftete* (*lossy*) Kompression. Im ersten Fall werden die Originaldaten exakt rekonstruiert, im zweiten Fall werden gewisse Fehler in Kauf genommen, die sich je nach Kompressionsgrad mehr oder weniger stark auf die Qualität auswirken, die der menschliche Empfänger wahrnimmt. Eine zusätzliche Möglichkeit ist *progressive* Kompression und Übertragung. Dabei ergibt eine Umordnung der Information eine schrittweise Annäherung an die Originaldaten im Verlauf der Decodierung.

Die wichtigsten Kriterien für Kompressionsverfahren sind Kompressionsgrad, Zeitaufwand für die Kompression und für die Dekompression, Aufwand für die Implementierung in Hardware oder Software, sowie aus praktischer Hinsicht die Verbreitung und die Standardisierung eines Verfahrens. Besser komprimierende Verfahren sind vielfach zeitaufwendiger. Es ist zu beachten, daß gewisse Kompressionsalgorithmen und Implementierungen oder Teile davon durch Patente geschützt sind.

4.1.2 Modellierung

Die zentrale Bedeutung der Wahrscheinlichkeiten in der Informationstheorie führt zu einer zumindest konzeptionellen Gliederung des Kompressionsverfahrens in *Modellierung* und *Codierung* (siehe Bild 1). Die Modellierung ist verantwortlich für die Schätzung der Wahrscheinlichkeiten, mit denen die Symbole im Quelltext auftreten. Bei einigen Kompressionsverfahren werden die Quellsymbole vor der eigentlichen Kompression gezählt. Dann tritt die Auftrittshäufigkeit jedes Symbols an die Stelle seiner Auftrittswahrscheinlichkeit. Die Codierung benutzt die Auftrittswahrscheinlichkeiten, um die Quellsymbole in Kanalsymbole umzusetzen. Sie wird auch Entropiecodierung genannt.

Es gibt zwar Modelle, die für alle Arten von Daten gute Kompressionsergebnisse liefern (siehe Abschnitt 4.2.3), aber eine wirklich effiziente Modellierung hängt stark von der Art der Daten ab. Gewisse Modelle erfassen die Symbolwahrscheinlichkeiten in Texten sehr gut, andere sind für Bilder, Graphiken, Video, Animationen, Klänge oder Daten anderer Art konstruiert. Neue Modelle werden immer wieder vorgeschlagen. Dieser Abschnitt beschränkt sich deshalb auf einige allgemeine Anmerkungen zur Modellierung.

Modellgenauigkeit. Es ist nur selten möglich, daß das Modell die tatsächlichen Wahrscheinlichkeiten exakt schätzt. Abweichungen führen zu schlechterer Kompression. Abweichungen in der Nähe der Gleichverteilung ergeben kleinere, Abweichungen bei sehr ungleichmäßigen Verteilungen größere Verluste. Abweichungen in Richtung Gleichverteilung sind weniger kritisch als Abweichungen in die andere Richtung.

Mit *Modell n. Ordnung* bezeichnet man ein Modell, das *n* Vorgängersymbole zur Schätzung der (bedingten) Wahrscheinlichkeiten verwendet. Mit zunehmender Ordnung kann die Entropie eines genauen Modells nur sinken oder gleichbleiben; eine genaue Schätzung wird aber wegen der kleineren Datenbasis immer schwieriger. Die Entropie eines Modells ist dabei gleich der Entropie einer Quelle, die genau dem Modell entspricht.

Adaptivität. Modelle können sich mehr oder weniger und auf verschiedene Weise an Nachrichten anpassen. Dafür werden die Begriffe *statisc*h (nicht anpassend) und *adaptiv* (anpassend) verwendet, wobei sich diese Unterscheidung auf verschiedene Aspekte beziehen kann. Die wichtigsten sind:

- Bildung von Einheiten: Einheitliches Modell für alle Nachrichten (basierend auf Beispieldokumenten), ein Modell pro Nachricht, Unterteilung der Nachricht in Blöcke (z.B. Rechtecke oder Quadrate in Bildern).
- Bildung von Klassen: Ganze Nachrichten oder Nachrichtenteile werden vor der Kompression klassifiziert. Die Klasse wird übermittelt; pro Klasse wird ein statisches Modell verwendet, das sowohl dem Sender wie dem Empfänger bekannt ist.
- Vorausberechnung: Als Vorverarbeitung wird eine Statistik der Nachricht erstellt, und das resultierende Modell oder der Code vor der eigentlichen Nachricht übermittelt.
- Fortlaufende Anpassung: Ausgehend von einem Initialmodell (z.B. mit Gleichverteilung) wird das Modell auf Grund der Eingangssymbole verfeinert. Wichtig ist, daß das Modell nur auf Eingangssymbolen basiert, die schon übertragen wurden oder die vom Empfänger hergeleitet werden können. Andernfalls ist eine Decodierung unmöglich.
- Zerfall: Nimmt man an, daß die Wahrscheinlichkeiten nicht stationär sind, kann man vor kurzem gesehene Symbole für die Modellbildung stärker gewichten und den Einfluß früher aufgetretener Symbole zerfallen lassen.

Verfahren mit mehr statischen Aspekten sind generell bezüglich Speicherplatz und Ausführungszeit effizienter, während die adaptiven Verfahren besser komprimieren.

Umformungen der Quelle. Um den Codierteil einfacher (Programmgröße), effizienter (Ausführungszeit) und effektiver (Kompression) zu gestalten, wird in vielen Fällen im Modell eine Umformung der Quelle vorgenommen. Beispiele sind:

- Erweiterung: Eine Folge von Symbolen wird zu einem Symbol zusammengefaßt. Eine implizite Erweiterung ergibt sich, wenn Bits zu Bytes zusammengefaßt werden.
- Differenzbildung: Die Differenz zwischen einer Prognose für den nächsten Wert und dem tatsächlichen Wert bildet das Eingangssymbol des Codierers. Dadurch wird die Wahrscheinlichkeitsverteilung ungleichmäßiger (im Vergleich zu einem Modell 0. Ordnung) oder statischer (im Vergleich zu einem Modell höherer Ordnung ohne Differenzbildung). Die Differenzbildung wird insbesondere bei Daten mit kontinuierlicher Natur (Bilder, Klänge) verwendet (siehe auch Abschnitt 4.3).
- Lauflängencodierung (run length coding): Aufeinanderfolgende gleiche Symbole werden zu einem *Lauf* (*run*) zusammengefaßt. Symbole und ihre Lauflängen werden abwechselnd codiert.
- Strukturierung: Es werden mehrere Codierer hintereinander und/oder parallel verwendet (Beispiel: Separate Entropiecodierung für Symbole und Lauflängen).

Asymmetrie. Insbesondere bei Modellen, die laufend angepaßt werden, ist die Berechnung des Modells im Sender meist wesentlich aufwendiger als die Rekonstruktion des Modells im Empfänger. Kann die Kompression schon vor der Versendung geschehen oder wird die komprimierte Nachricht an viele Empfänger verteilt, so kann der erhöhte Kompressionsaufwand in Kauf genommen werden. Man spricht in diesem Fall von einem asymmetrischen Kompressionsverfahren.

4.2 Verlustfreie Kompression

4.2.1 Huffman-Codierung

Die Aufgabe der Codierung ist klar definiert: Gegeben ein Modell mit Wahrscheinlichkeiten für die Quellsymbole, codiere so nahe wie möglich am theoretischen Grenzwert, d.h. so wenig wie möglich über der Entropie des Modells. Dies geschieht mit der Huffman-Codierung und der arithmetischen Codierung. Ein Vergleich dieser beiden Methoden befindet sich am Ende von Abschnitt 4.2.2. Für gegebene Wahrscheinlichkeiten erstellt der Huffman-Algorithmus [Huffman 52] eine Codetabelle, die jedem Quellsymbol eine Bitfolge zuordnet. Die Huffman-Codierung setzt also jedes Quellsymbol in eine *ganze* Anzahl von Bits um. Der Algorithmus ist optimal in dem Sinne, daß mit keiner anderen Zuordnung von Bitfolgen eine bessere Kompression erreicht werden könnte. Die Menge der verwendeten Bitfolgen ist *kommafrei* (*Präfixcode*), d.h. das Ende einer Bitfolge, und damit der Anfang der nächsten, kann fehlerfrei erkannt werden.

Die Codetabelle kann als binärer Baum dargestellt werden, bei dem die beiden von einem Knoten ausgehenden Äste jeweils mit den Bitwerten „0“ und „1“ und die Blätter des Baumes mit den übertragenen Symbolen markiert sind. Eine solche Darstellung ist insbesondere für den Empfänger geeignet. Ausgehend von der Wurzel des Baumes wird mit jedem empfangenen Bit die Position im Baum geändert. Beim Erreichen eines Blattes wird das decodierte Symbol ausgeschrieben, und es wird an die Wurzel des Baumes zurückgesprungen.

Der Huffman-Algorithmus baut den Codebaum rekursiv auf, indem er jeweils die zwei Symbole mit den kleinsten Wahrscheinlichkeiten zu einem Teilbaum zusammenfaßt. Dieser Teilbaum geht als ein Symbol mit der Summe der Wahrscheinlichkeiten der zusam-

mengefaßten Symbole in den weiteren Verlauf des Algorithmus ein. Den beiden Ästen des Baumes werden die Bitwerte „0“ und „1“ zugewiesen. Dieser Prozeß wird solange fortgesetzt, bis alle Symbole zu einem Baum zusammengefaßt sind. Meist wird die Huffman-Codierung mit einem statischen Modell verwendet, d. h., es wird eine einzige Codetabelle berechnet. Adaptive Varianten vermeiden die Neuberechnung des ganzen Baumes nach jeder Modelländerung, indem sie nur gewisse Knoten austauschen.

Beispiel (1)

Gegeben sei eine Quelle mit den fünf Symbolen *a*, *b*, *c*, *d* und *e* mit den Wahrscheinlichkeiten 0.16, 0.22, 0.14, 0.12 und 0.36. Der Ablauf des Huffman-Algorithmus ist in Bild 2 links dargestellt. Die Zusammenfassung der zwei Symbole mit kleinster Wahrscheinlichkeit ist durch Pfeile mit den zugeordneten Kanalsymbolen angezeigt. Daraus ergibt sich der Codebaum in Bild 2 rechts.

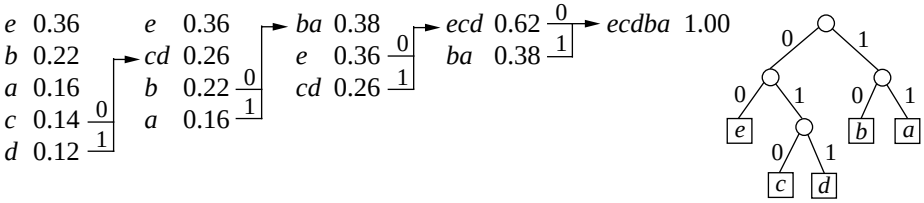

Bild 2 Beispiel des Huffman-Algorithmus: Symbolzusammenfassung und Codebaum

Bild 3 zeigt die Berechnung des Informationsgehaltes *I* der einzelnen Symbole, der Entropie *H* der Quelle und der durchschnittlichen Codewortlänge *L*. Vergleicht man die theoretisch optimale Codierung (gemäß Quellcodierungstheorem die Entropie von 2.198 Bits/Symbol) mit der tatsächlich erreichten Datenrate (2.26 Bits/Symbol), ergibt sich eine Redundanz von 2.7%. Nimmt man trivialerweise 3 Bits/Symbol für die Quelle an, so resultiert eine Kompression um 24.7%.

s	*p*	*I*	*p I*	Code	Bit/Symbol	*p* Bit/Symbol
a	0.16	2.644	0.423	**11**	2	0.32
b	0.22	2.184	0.481	**10**	2	0.44
c	0.14	2.837	0.397	**010**	3	0.42
d	0.12	3.059	0.367	**011**	3	0.36
e	0.36	1.474	0.531	**00**	2	0.72
Summe	1.00	*H* =	2.198			*L* = 2.26

Bild 3 Berechnung von Entropie *H* und Kompression *L* für Beispiel (1).
s = Zeichen, *p* = seine Wahrscheinlichkeit, *I* = sein Informationsgehalt.
I und *H* sind in Kapitel B2, Randnummer (2) und (3) definiert.

4.2.2 Arithmetische Codierung

Die arithmetische Codierung [Rissanen 79] geht wie die Huffman-Codierung von Wahrscheinlichkeiten aus, die von einem Modell zur Verfügung gestellt werden. Dabei wird im

Gegensatz zu anderen Verfahren jedes Quellsymbol in eine *gebrochene* Anzahl Bits umgesetzt. Abgesehen von Hardwarebeschränkungen für die Rechengenauigkeit ist damit eine absolut redundanzfreie Kompression möglich.

Das Prinzip der arithmetischen Codierung ist die Darstellung von Nachrichten als Intervalle in den rationalen Zahlen. Ausgehend vom Intervall [0 … 1) am Anfang der Übermittlung wird das Quellintervall mit jedem neuen Quellsymbol verkleinert. Es wird entsprechend den vom Modell gelieferten Wahrscheinlichkeiten proportional aufgeteilt, und der dem nächsten Symbol entsprechende Abschnitt wird ausgewählt. Die Quellintervalle werden auf binäre Kanalintervalle abgebildet. Sobald ein Quellintervall ganz in der einen oder anderen Hälfte des Kanalintervalls enthalten ist, kann das Kanalintervall halbiert und ein Kanalsymbol ausgeschrieben werden.

Bei der Codierung wird für jedes neue Quellsymbol das neue, reduzierte Intervall berechnet und, falls möglich, ein Kanalsymbol ausgegeben. Der Empfänger berechnet ebenfalls Quellintervalle und Kanalintervalle und kann ein Quellsymbol ausgeben, sobald das Kanalintervall ganz in einem Quellintervall enthalten ist.

Bild 4 zeigt ein Beispiel dazu, mit den Symbolen und Wahrscheinlichkeiten des Beispiels (1). Auf Grund der Quellsymbole „b“ und „d“ kann die Bitkette 01010 ausgegeben werden. Schon nach dem dritten Bit weiß der Empfänger, daß die Nachricht mit „b“ beginnt, und kann „b“ ausgeben. Nach fünf Bits kann die Nachricht nur noch mit „bd“, „bce“ oder „bea“ beginnen.

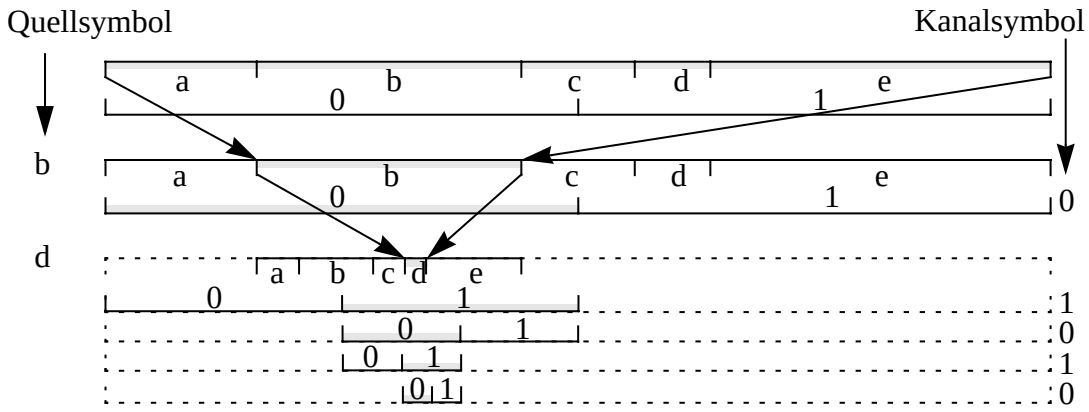

Bild 4 Arithmetische Codierung: Intervallschachtelung

Implementierung. Bei der Implementierung der arithmetischen Codierung sind die folgenden praktischen Probleme zu beachten [Witten 87]:

- Numerik: Der Algorithmus verwendet Multiplikationen und Divisionen. Es muß garantiert sein, daß Codierer und Decodierer exakt gleich rechnen. Die Auflösung der Intervalldarstellung wird durch Skalierung des Intervalls so groß wie möglich gehalten. Bei 32-Bit-Arithmetik stehen für die Wahrscheinlichkeiten des Modells bis zu 15 Bit zur Verfügung. 16-Bit-Arithmetik ist für die Codierung von Text ungenügend.
- Verzögerungen: Wenn das Quellintervall die Mitte des Kanalintervalls umspannt oder die Kompression sehr stark ist, kann lange kein Bit ausgegeben werden. Solche Verzögerungen können stören (z.B. in Echtzeitumgebungen). Das Modell wird leicht verzerrt, was einen gewissen Effizienzverlust ergibt.

- Abschluß der Nachricht: Das Quellalphabet wird um ein zusätzliches Symbol *Nachrichtenende* erweitert, das eine sehr kleine Wahrscheinlichkeit besitzt. Damit dieses letzte Symbol beim Empfänger ankommt, genügt es, mit zwei Bits das Kanalintervall so zu verkleinern, daß es ganz in das Quellintervall fällt.

Der Q-Coder. Der Q-Coder [IBM 88] ist ein stark vereinfachter arithmetischer Codierer Ein vorgeschaltetes Modell bildet Quellsymbole auf möglichst unabhängige und asymmetrisch verteilte Binärsymbole ab. Der Q-Coder schätzt die Wahrscheinlichkeiten nullter Ordnung dieser Binärsymbole mit einem kurzen Zerfall, diskretisiert so, daß mit möglichst wenigen Stufen möglichst kleine Kompressionsverluste entstehen. Der Q-Coder wird in den Normen ISO/IEC 11544 und ISO/IEC 12042 verwendet (siehe Kapitel F2).

Vergleich mit Huffman-Codierung. Die Huffman-Codierung war über Jahrzehnte die gebräuchlichste Methode und ist deshalb in vielen Verfahren und Standards anzutreffen, vielfach zusammen mit vorgeschalteten Umformungen. Sie wird erst in jüngster Zeit langsam von der arithmetischen Codierung abgelöst. Die arithmetische Codierung komprimiert effizienter, insbesondere bei sehr ungleichen Wahrscheinlichkeitsverteilungen. Die Huffman-Codierung ist in der statischen Variante schneller, nicht aber bei einem adaptiven Modell. Ein statischer Huffman-Codierer und -Decodierer läßt sich sehr effizient in Spezialhardware implementieren; die arithmetische Codierung setzt für die Kompression von 8-Bit-Bytes 32-Bit-Arithmetik voraus und ist für 16-Bit-Maschinen ungeeignet.

4.2.3 Lempel-Ziv-Codierungen

In den siebziger Jahren untersuchten Ziv und Lempel verschiedene Möglichkeiten, die Kompressibilität *einzelner* Nachrichten zu definieren. In ihren Arbeiten führten sie verschiedene Algorithmen ein, von denen zwei mit Varianten in den meisten heutigen Kompressions- und Archivierungswerkzeugen verwendet werden. Sie werden entsprechend dem Publikationsjahr meist mit LZ77 [Ziv 77] und LZ78 [Ziv 78] bezeichnet.

Beide Algorithmen komprimieren keine Einzelzeichen, sondern merken sich wiederkehrende Zeichengruppen im Text, fassen sie als neue „Superzeichen“ auf und codieren sie wie Einzelzeichen.

Das implizite Modell beruht auf der Annahme, daß schon gesehene Symbolfolgen häufig wieder auftauchen, und kann viele speziellere Modelle gut annähern. Die Lempel-Ziv-Codierungen werden deshalb als „Allesfresser“ in Kompressions- und Archivierungswerkzeugen eingesetzt; ihre „Verdauung“ (d. h. Kompressionsgrad) ist aber selten ganz so gut wie explizit modellbasierte Kompression mit einem geeigneten Modell. Ziv und Lempel konnten zwar zeigen, daß ihre Algorithmen in einem weiten Sinn optimal codieren, dazu muß aber die Nachricht lang und der Arbeitsspeicher groß sein.

Die Lempel-Ziv-Kompressionsmethoden sind insbesondere für Textkompression geeignet. Texte, von gewöhnlicher Prosa bis zu Programmtexten, enthalten wiederholt vorkommende Zeichenfolgen und Wörter. Diese Zeichenfolgen sind für verschiedene Texte unterschiedlich, so daß sich eine globale Tabelle nicht lohnt. Auch bestehen bei Texten keine inneren Zusammenhänge zwischen den Codewerten der einzelnen Symbole (Buchstaben) und ihren Wahrscheinlichkeiten, welche mit spezieller Modellierung ausgenützt werden könnten.

Um sequentielle Abhängigkeiten zu berücksichtigen, werden Quellsymbolfolgen variabler Länge auf einzelne Kanalsymbole umcodiert. Dabei werden die am häufigsten vorkommenden Symbolfolgen, unabhängig von ihrer Länge, ausgewählt; ihre relativen Häufigkeiten liegen daher näher bei der Gleichverteilung als diejenigen der Quellsymbole,

und es wird meist eine einfache Codierung mit einer konstanten Anzahl Bits pro Kanalsymbol gewählt.

LZ77: Beliebige Teilfolgen

LZ77 [Ziv 77] beruht darauf, daß beliebige schon gesehene Teilfolgen zur Übertragung weiterer Symbole benutzt werden. Die längstmögliche Teilfolge wird gesucht, und ihre Anfangsposition und Länge werden übertragen. Varianten unterscheiden sich darin, wie noch nicht vorgekommene Symbole übertragen werden. Aus praktischen Gründen wird die Suche auf einen Puffer mit dem zuletzt gesehenen Teil der Nachricht beschränkt. Dadurch ist der Algorithmus automatisch adaptiv; er paßt sich langfristig den sich ändernden Wahrscheinlichkeitsverteilungen in der Nachricht an. Das Hauptproblem der Implementierung besteht darin, die längste Folge im gespeicherten Nachrichtenteil schnell zu finden. Dazu können als Suchbäume *Tries* oder ähnliche Datenstrukturen verwendet werden. Eine Variante von LZ77 ist als *LZSS* (Lempel-Ziv-Storer-Szymansky) bekannt. ISO/IEC 15200 (siehe Kapitel F2) und Programme wie *zip* und *gzip* verwenden LZ77.

LZ78: Teilfolgen aus Wörterbuch

LZ78 [Ziv 78] teilt die Nachricht in Teilfolgen (Wörter) auf, die in einem Wörterbuch abgelegt werden. Es wird jeweils das längste passende Wort im Wörterbuch gesucht und seine Nummer übertragen. Mit jedem übertragenen Wort wird auch das Wörterbuch erweitert. Das neue Wort besteht aus dem übertragenen Wort, verlängert um das nächste Quellsymbol. Das nächste Quellsymbol kann separat übertragen werden, oder es kann das erste Symbol des nächsten übertragenen Wortes extrahiert werden. Im diesem Fall muß das Wörterbuch mit den einzelnen Symbolen als Anfangsworten initialisiert werden. Diese Variante ist unter der Bezeichnung *LZW* (Lempel-Ziv-Welch [Welch 84]) bekannt. ISO/IEC 11558 (siehe Kapitel F2) und *UNIX compress* verwenden LZ78.

Aus praktischen Gründen ist die Größe des Wörterbuches beschränkt. Meist wird bei Erreichen dieser Größe die Aufnahme neuer Wörter einfach gestoppt. Zerfall läßt sich nicht so einfach implementieren wie bei LZ77, dafür ist der Aufbau des Suchbaums für das längste Wort einfacher als bei LZ77.

Als Beispiel diene die Kompression von „abaabacdabaacdabaacd“ mit LZW (Bild 5). 20 Eingabesymbole zu je 8 Bit werden mit 12 Ausgabesymbolen zu je 9 Bit codiert, das ergibt eine Kompression um 32.5%.

Ausgabe	Eintrag	Ausgabe	Eintrag	Ausgabe	Eintrag
a (97)	-	a (97)	aba (259)	ac (260)	abaa (263)
b (98)	ab (256)	c (99)	ac (260)	da (262)	acd (264)
a (97)	ba (257)	d (100)	cd (261)	ba (257)	dab (265)
ab (256)	aa (258)	aba (259)	da (262)	acd (264)	baa (266)

Bild 5 Kompression von „abaabacdabaacdabaacd“ mit LZW. Die drei Spaltenpaare (Ausgabe, Eintrag) sind untereinander zu denken. Die Zahlen in Klammern bezeichnen die Positionen der Einzelsymbole und Symbolfolgen im Wörterbuch.

Bild 5 zeigt das in einem Schritt jeweils übertragene Wort (Ausgabe) und das neu im Wörterbuch eingetragene Wort (Eintrag), das sich aus der Ausgabe des vorhergehenden Schrittes und dem ersten Symbol des gegenwärtigen Schrittes zusammensetzt. Der Decodierer

initialisiert sein Wörterbuch ebenfalls mit den Einzelsymbolen (in den Positionen 0 bis 255) und baut das gleiche Wörterbuch auf wie der Codierer. Wenn die Wortanzahl eine neue Zweierpotenz erreicht, wird die Anzahl der Bits pro Kanalsymbol erhöht.

4.2.4 Burrows-Wheeler-Transformation

In der letzten Zeit gewinnt ein neues Verfahren langsam an Bedeutung: Eine Kombination der Burrows-Wheeler-Transformation mit anschließender Kompression [Burrows 94]. Die Burrows-Wheeler-Transformation benutzt eine komplette Sortierung aller Rotationen einer Nachricht oder eines Blocks einer Nachricht, um die Nachricht umzuordnen. Die Vorgängersymbole von nach der Sortierung benachbarten Rotationen zeigen eine hohe Regelmäßigkeit und können deshalb sehr stark komprimiert werden. Auf den ersten Blick erstaunlich, aber für die Benutzung in einem Kompressionsverfahren unabdingbar ist, daß die Burrows-Wheeler-Transformation umgekehrt werden kann. Klarer als bei anderen Verfahren steigt der Kompressionsgrad auch bei großen Blocklängen bei Vergrößerung der Blocklänge immer noch an. Dafür steigt aber auch der nötige Speicherplatz linear mit der Blocklänge.

4.3 Verlustbehaftete Kompression

Mit dem Aufkommen von Multimedia wurden verlustbehaftete Verfahren relevant: verfälschte Rekonstruktion wird hingenommen, solange der Benutzer dies bei der Präsentation nicht oder nur als zulässige Qualitätsreduktion wahrnimmt. Dafür steigt der Kompressionsfaktor von durchschnittlich 2 bis 3 auf teilweise über 100 (für Video).

Die wichtigsten verlustfreien Kompressionsverfahren beziehen sich auf drei Medientypen: Audio (eindimensionale Tonschwingung), Festbilder (zweidimensional als horizontale mal vertikale Auflösung) und Video (zum Festbild kommt die Dimension Zeit hinzu). Die gängigen Kompressionsverfahren für diese Medientypen unterscheiden sich teilweise erheblich. Ursache dafür sind die unterschiedlichen Dimensionalitäten, ausgenutzte Spezifika der menschlichen Wahrnehmung und die historische Entwicklung.

Audio ist weniger speicherintensiv, verlangt aber hohe Klangtreue (das Ohr unterscheidet u. a. in einem Klanggemisch verschiedene Quellen, während das Auge z. B. über unterbrochene Linien hinweg zu „integrieren“ gewohnt ist). Daher wurden lange Zeit nur Verfahren betrachtet, die mit den ursprünglichen Signalen (Schallwellen) arbeiten und diese möglichst originalgetreu wiederherstellen. Als dann an Verfahren herangegangen wurde, die den Höreindruck, aber nicht notwendigerweise die Signalform, möglichst gut erhalten wollten, konzentrierte man sich auf Sprachübertragung. Hier kommt Wissen über die menschliche Sprache zum Einsatz. Erst in jüngster Zeit wird versucht, Audio in seine Frequenzen zu zerlegen, um im Frequenzbereich höhere Kompressionsgrade zu erzielen.

Dieser dritte Weg ist seit etlichen Jahren für Festbilder der am häufigsten benutzte. (Abschnitt 4.3.1 geht darauf ein, wie Bilder als Schwingungen in 2D interpretiert werden können.) Speziell die diskrete Kosinustransformation DCT und neuerdings Wavelets kommen zum Einsatz. Fraktale Kompression nimmt hier eine Sonderstellung ein, andere Verfahren gelten derzeit nicht als konkurrenzfähig.

Die Interpretation von Video als dreidimensionale Schwingung wird in der neuesten Forschung für Frequenztransformation ausgenutzt, Standards sind dafür noch nicht in Sicht. Üblich ist die Interpretation als Folge von Bildern. Daher kommen Verfahren zur Festbildkompression zum Einsatz, und zwar fast ausschließlich DCT-basiert, ergänzt um Bewe-

gungsschätzung, d.h. die Vorhersage von Folgebildern aus dem aktuellen Bild über Bewegungsvektoren. Bilder werden als Matrix von kleinen quadratischen Kacheln verstanden, und die Vektoren geben an, wie Kacheln vom Referenz- zum Vorhersagebild „wandern“ (z.B. bei einem Kamera-Zoom von der Mitte zur Bildperipherie).

Bild 6 faßt diesen Überblick zusammen und ordnet wichtige im folgenden behandelte Standards ein.

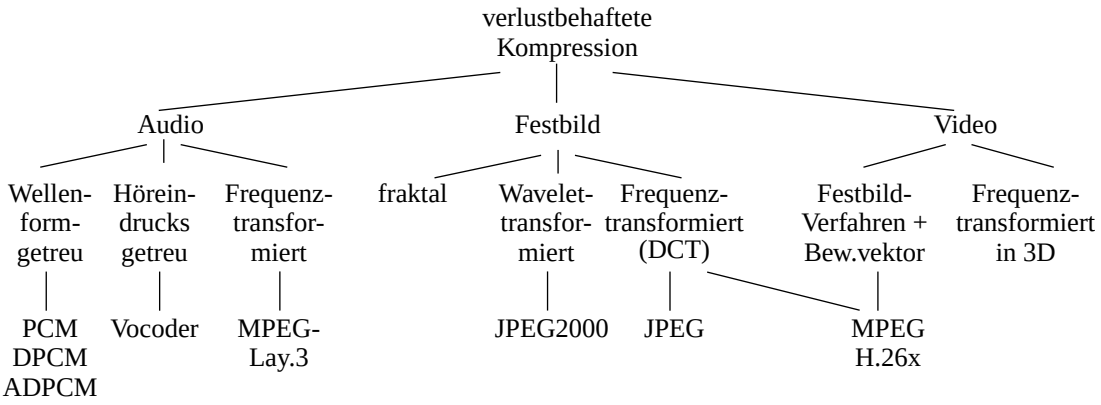

Bild 6 Wichtige verlustbehaftete Kompressionsverfahren und -Standards

4.3.1 Zusammenhang mit verlustfreier Kompression

Ausgangspunkt sind folgende Eigenschaften vieler Quellen, die auch für verlustfreie Verfahren ausgenutzt werden:

1 ungleiche Zeichenverteilung;

2 Zeichen(ketten-)vervielfachung;

3 häufiges Auftreten einer gegenüber den kombinatorischen Möglichkeiten kleinen Anzahl von Zeichenketten;

4 datentypspezifische funktionale Zusammenhänge, die zur Vorhersage erwarteter Codewörter genutzt werden.

Auf diesen Eigenschaften beruhen die in Abschnitt 4.1.2 genannten „Umformungen der Quelle“: Erweiterung (3), Differenzbildung (4) und Lauflängencodierung (2), sowie die Entropiecodierung allgemein (1) und der Einsatz von Wörterbüchern (4). Multimediadaten aus der realen Welt (Fotos, Audio- und Videoaufzeichnungen) weisen Eigenschaft (4) deutlicher oder zumindest in anderer Weise auf als computergenerierte Graphiken, Töne und Videos („die Natur macht keine Sprünge”). Bei der Ausnutzung der vier Eigenschaften können folgende Maßnahmen für verlustbehaftete Kompression ergriffen werden:

a Weglassen kleiner Vorhersagefehler (z.B. ein geringfügig falsch vorhergesagter Wert wird bei Differenzbildung nicht korrigiert);

b Ignorieren seltener oder wenig relevanter Quellsymbole (z.B. Ausreißer in Lauflängen);

c Zusammenfassen ähnlicher (dabei seltener oder wenig wichtiger) Quellsymbole zum selben Code (Dekompression als repräsentatives Symbol).

Quantisierung. Die Zusammenfassung nach (c) erfolgt bei numerischen Quellsymbolen meist durch Quantisierung: jeder Code repräsentiert dabei ein Intervall von Quellcode-

wörtern. Entweder wird die Intervallnummer codiert oder die Zusammenfassung erfolgt durch Darstellung mit geringer numerischer Genauigkeit – im einfachsten Fall durch Weglassen niederwertiger Bits, häufiger durch Integer-Division ohne Rest. Bei Multimediadaten kann der Kompressionsgrad durch ungleichförmige Quantisierung erhöht werden: (bezüglich der Wahrnehmung) weniger wichtige oder schlechter unterscheidbare Symbole werden grob quantisiert (in größeren Intervallen zusammengefaßt). Multimediadaten aus der (analogen) realen Welt werden bei der Analog-Digital-Wandlung auf Grund der beschränkten numerischen Präzision immer auch quantisiert. *Vektorquantisierung* codiert eine Menge von Quellcodewörtern als Index in ein Codebuch, die Decodierung rekonstruiert alle Quellcodewörter derselben Menge als ein und dasselbe Codewort. Das rechenaufwendige Verfahren minimiert bei der Codebuch-Konstruktion (ggf. anhand einer Trainings-Sequenz) den mittleren Fehler der rekonstruierten Codewörter.

Transformationscodierung. Bei der heute noch weitgehend signalorientierten Medienrepräsentation werden die Daten häufig zunächst in eine Darstellung transformiert, bei der (für die menschliche Wahrnehmung) bedeutende und weniger bedeutende Signalanteile getrennt sind. Ein wichtiges Beispiel hierfür ist die gegenüber Helligkeit geringere Farbempfindlichkeit des Auges: transformiert man Bilder von dem für Computer üblichen Farbraum RGB (hier wird jeder Bildpunkt durch seine Rot-, Grün- und Blau-Anteile dargestellt) nach YUV (hier gibt Y die Helligkeit des Bildpunktes an, U und V codieren die Farbe), so kann die Farbinformation rund 4 bis 8 mal seltener abgetastet oder mit weniger Bits codiert werden (*subsampling*). Die Bildqualität bleibt dabei subjektiv unverändert.

Viele gängige Kompressionsverfahren nutzen die unterschiedliche Empfindlichkeit menschlicher Wahrnehmung für verschiedene Frequenzen aus. Faßt man z.B. Bilder als zweidimensionale Schwingungen auf, dann sind niedere Frequenzen (Verläufe) wichtiger als hohe (Kanten). Die Signale werden abschnittsweise transformiert, wozu sich insbesondere die Fourier- und die Kosinustransformation eignen, genauer gesagt deren digitale Versionen (DFT und DCT): dabei werden die zu transformierenden Abschnitte durch je n Amplitudenwerte repräsentiert (bei Bildern meist Planquadrate mit $n = m \cdot m$ Bildpunkten). Diese können mit der schnellen Fouriertransformation (FFT) in der Zeit $O(n \log n)$ transformiert werden. Die diskrete Kosinustransformation ist am weitesten verbreitet, da sie Randeffekte vermeidet und nur Kosinus-, keine Sinusanteile enthält. Die zu transformierenden Signale sind technisch bedingt immer bandbegrenzt. Daher kann bei geeigneter Wahl der Frequenzen die Reihenentwicklung nach n Schwingungen abgebrochen werden. Aus n Abtastwerten werden also n Koeffizienten der Transformierten, die Transformation ist bis auf numerische Ungenauigkeiten verlustfrei. Wie die Farbraumanpassung ist die Frequenztransformation keine Kompression, sondern ein Vorverarbeitungsschritt für verlustbehaftete Schritte, vor allem starke Quantisierung.

4.3.2 Audiokompression

Pulscodemodulation. Das Frequenzgemisch eines Audiosignals wird z.B. beim Telefon auf 3.4 kHz, bei Audio-CDs auf 22 kHz begrenzt. Das Abtasttheorem von Nyquist besagt, daß Signale mit Grenzfrequenz f_G zur fehlerfreien Rekonstruktion mit einer Abtastfrequenz $f_A > 2f_G$ abgetastet werden müssen (Telefon: 8 kHz, Audio-CD: 44.1 kHz). Logarithmische Quantisierung nutzt die Tatsache aus, daß Quantisierungsfehler bei leisen Tönen (Abtastwert nahe Null) deutlicher zu hören sind als bei lauten: um den Nullpunkt sind die Quantisierungsintervalle klein, nach oben und unten verdoppelt sich ihre Länge mit jedem Intervall. Die drei Schritte Abtastung, Quantisierung und Quellcodierung (meist einfach die Darstellung der Intervallnummer als Integer) werden als *Pulscodemodulation* (PCM) zusammengefaßt. Beim ISDN-Telefon (8 kHz, 8 Bit d.h. 256 Quantisie-

rungsintervalle, also 64 KBit/s) und bei Audio-CDs (44.1 kHz, 16 Bit, Stereo, also über 1.4 MBit/s) wird PCM ohne weitere Kompression verwendet. Dagegen wird bei differentieller PCM (DPCM) nur die Veränderung eines Abtastwertes A_i (der Intervallnummer) gegenüber A_{i-1} angegeben. Auf Grund der Eigenschaft „die Natur macht keine Sprünge“ sind weniger Bits für die Darstellung der Differenz notwendig. Bild 7 veranschaulicht PCM und DPCM.

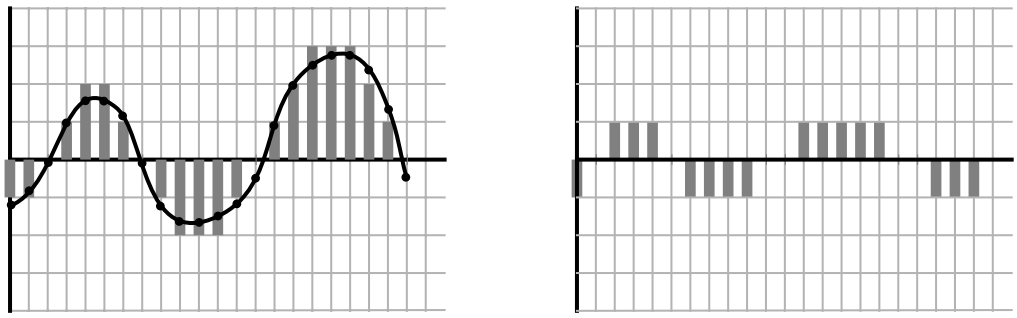

Bild 7 PCM-Codierung (links): Abtastung (schwarze Punkte), Quantisierung (graue Balken, horizontale Striche entsprechen der Intervallmitte); entsprechende DPCM-Codierung (rechts).

Von adaptiver DPCM (ADPCM) gibt es verschiedene Varianten: im einfachsten Fall wird die Differenz $A_{i+1} - A_i$ als $A_i - A_{i-1}$ vorhergesagt; der Code (Fehlerkorrektur) für A_{i+1} verbessert A_{i+1} und adaptiert die Vorhersage für A_{i+2}. Es werden noch weniger Bits für die Fehlerkorrektur benötigt. Die im CD-ROM/XA- und CD-I-Format standardisierten ADPCM-Varianten verwenden 4 Bit zur Codierung von $A_i - A_{i-1}$: ein Vorzeichenbit und eine 3-Bit-Kombination, die erstens auf einen Tabelleneintrag mit der eigentlichen Vorhersage verweist und zweitens die für den nächsten Wert zu verwendende Tabelle bestimmt; so können in steilen und flachen Abschnitten des Signals verschiedene Tabellen verwendet werden. Statt expliziter Fehlerkorrektur geht das vorhergesagte A_i in den nächsten Wert ein, also $A_{i+1} - A_i$. Ein nachgeschalteter Filter glättet die meist alternierenden Fehler. Gegenüber der Audio-CD wird so ohne hörbaren Verlust die Speicherkapazität von 74 Minuten auf fast 5 Stunden vervierfacht.

Delta-Modulation verwendet nur ein einziges Bit, das die Richtung des Signals codiert, der Betrag der vorhergesagten Differenz bleibt konstant. Gerade Signalabschnitte werden durch Null-Eins-Folgen modelliert, steile Flanken können durch Null- oder Einsfolgen, gegebenenfalls erst verzögert, „aufgeholt“ werden. Das Verfahren bringt dennoch sehr gute Ergebnisse, wenn wie oben Filter eingesetzt werden und „Oversampling“: jedes Abtastintervall wird im Rahmen der Berechnung unterteilt, ein Bit codiert nur die Signaländerung in einem Teilintervall. Bei *CVSD* (*continuous variable slope delta modulation*) verdoppeln aufeinanderfolgende gleichwertige Bits die vorhergesagte Signaldifferenz, ungleichwertige wechseln die Richtung und halbieren die Vorhersage. Es kommt im Bluetooth-Standard zum Einsatz.

Sprachübertragung. Codierer, die auf Sprachübertragung optimiert sind (z. B. für GSM-Mobiltelefone) werden als *Vocoder* (*voice coder*) bezeichnet. Sie arbeiten häufig mit einer Kurz- und Langzeitvorhersage. Kurzzeit bezieht sich auf die Länge des in einem Schritt verarbeiteten Signalabschnitts (um die Verzögerung zu begrenzen, verwendet die Mobiltelefonie Abschnitte von rund 20 ms, also z. B. 160 Amplitudenwerte bei 8 kHz Abtastrate). Da sich die Lautformung im menschliche Sprechapparat nur alle 10 – 100 ms

ändert, kann ein solcher Abschnitt als *eine* „Einstellung“ des Sprachtrakts interpretiert werden, mathematisch modelliert mit z. B. 13 Parametern (Luftstromintensität, Grundfrequenz sowie Parametern der Sprachtrakt-Geometrie, die die Lautformung und mathematisch gesehen die Impulsanregung, engl. *excitation*, bestimmt). Weit verbreitet ist die Abbildung dieser Parameter auf Codebucheinträge (*code excited linear prediction*, *CELP*), ggf. werden mittels Vektorquantisierung optimale Codebücher erzeugt. Die Langzeitvorhersage benutzt Wissen über die menschliche Sprache und betrifft größere Signalabschnitte. Weitergehende Verfahren konzentrieren sich auf die energiereichsten Frequenzen (z. B. fünf) pro Abschnitt (sog. *Formanten*). Allgemein werden die Verfahren laufend verbessert, Telefonqualität (64 KBit/s in PCM) wird heute bei GSM-Mobiltelefonen mit 13.2 KBit/s annähernd erreicht neuerdings auch mit der halben Rate. Mit deutlichen Qualitätseinbußen („Roboterstimme“) sind heute schon Datenraten unter 4 KBit/s realisierbar. Ein weiterer Fortschritt ist durch Spracherkennung und -synthese denkbar, wenn Sprechercharakteristika und Satzmelodie (Prosodie) geeignet modelliert und mitübertragen werden und die Erkennungsrate (in Echtzeit) gegenüber heutigen Verfahren noch gesteigert werden kann.

Frequenztransformation. Die für Videos gedachten MPEG-Standards enthalten Substandards zur Videokompression, eingeteilt in verschiedene *Schichten* (*layers*), die im wesentlichen den Aufwand zur Kompression und Dekompression und damit die erreichbare Klangqualität widerspiegeln. MPEG-Audiocodierer sind nicht auf Sprache optimiert und basieren auf der schnellen Fouriertransformation. Abschnitte mit 32 Abtastwerten werden dabei in die Koeffizienten von 32 Frequenzbändern transformiert. Die anschließende Quantisierung orientiert sich an einem *psychoakustischen Modell*, das die durchschnittliche Empfindlichkeit des Menschen für verschiedene Frequenzanteile und die sog. *Maskierung* berücksichtigt. Die Empfindlichkeit ist bei 4 kHz am höchsten, andere Frequenzen werden also nur wahrgenommen, wenn sie energiereich genug sind; als Referenz dient die empirisch ermittelte „Fletcher-Munson-Kurve“. Maskierung bedeutet, daß energiereiche Frequenzanteile die Empfindlichkeit für benachbarte Frequenzen verringern. MPEG Layer 3, auch als MP3 bekannt, unterstützt mehrere Bitraten zwischen 32 und 320 KBit/s und erreicht fast CD-Qualität (rund 1.4 MBit/s in PCM) bei 192 KBit/s. MP3 setzt sich als Internet-Musikformat und zunehmend in der Unterhaltungsindustrie durch; über Tauschbörsen auf Servern im Internet werden unter Umgehung der Verwertungsrechte Inhalte von Audio-CDs in diesem Format angeboten.

4.3.3 Festbildkompression

Festbildkompression im Ortsbereich kann z. B. Wissen über die Geometrie der Abtastwerte ausnutzen. So kann ein $N{\cdot}N$-Block der Helligkeitswerte von Bildpunkten zu deren Mittelwert und Standardabweichung plus einer $N{\cdot}N$-Ein-Bit-Matrix komprimiert werden. Die Ein-Bit-Matrix gibt an, ob der Wert im Urbild größer oder kleiner als der Mittelwert ist; bei der Decodierung wird dementsprechend der um die Standardabweichung vergrößerte oder verkleinerte Mittelwert verwendet. Am weitaus meisten verbreitet ist aber heute die Kompression im Frequenzbereich mit DCT. Fast alle Standards für Multimedia-Kompression *kombinieren* mehrere Kompressionsverfahren; insbesondere schließt sich an eine Phase der verlustbehafteten Kompression eine Phase der „unschädlichen“ verlustfreien Kompression an. DCT-basierte Verfahren seien daher am Beispiel des JPEG-Standards erläutert.

JPEG. *JPEG* ist benannt nach der gemeinsamen Expertengruppe von ISO und ITU, die die Standardisierung vorantrieb: joint photographic (pictures encoding) experts group.

Die Standardversion besteht aus folgenden sechs wesentlichen Schritten pro Bildebene. Dabei wird angenommen, daß jeder Bildpunkt aus einem Wort der Länge p besteht.

1 Clip/Pad: Die Bildgrenzen werden an Vielfache von 8 in beiden Dimensionen angepaßt.

2 Shift: jedes Wort (Bildpunkt) wird aus dem Intervall $[0, 2^p - 1]$ nach $[-2^{p-1}, 2^{p-1} - 1]$ verschoben, also z.B. bei 8-Bit-Bytes durch Subtraktion von 128 als vorzeichenbehaftet interpretiert. Das Gesamtbild wird dadurch ein annähernd symmetrisches „Oszillogramm".

3 DCT: Die 64 Bildpunkte $f(k, l)$ mit $0 \le k, l < 8$ jedes 8·8-Blockes werden DCT-transformiert in die 64 Koeffizienten $F(u,v)$ mit $0 \le u, v < 8$. Die 64 Bildpunkte werden dabeiquasi als Elemente einer zweidimensionalen Kombination von Kosinus-Schwingungen aufgefaßt und die Koeffizienten der 64 „untersten" Schwingungen nach folgender Formel ermittelt (für $u = 0$ bzw. $v = 0$ sind $C(u)$ bzw. $C(v) = 1/\sqrt{2}$, sonst =1):

$$F(u,v) = \frac{2}{N} C(u) C(v) \left[\sum_{k=0}^{N-1} \sum_{l=0}^{N-1} f(k,l) \cos\frac{u\pi(2k+1)}{2N} \cos\frac{v\pi(2l+1)}{2N} \right] \text{ (mit } N = 8)$$

Es entsteht also eine 8·8-Matrix reellwertiger Koeffizienten.

4 Quantisierung: Nachdem die Schritte 1 und 2 keine Kompression brachten und Schritt 3 im Gegenteil das Datenvolumen *vervierfacht* hat (durch den Übergang von Integer auf Real, falls Real-Werte den vierfachen Speicherbedarf haben), wird nun jeder Koeffizient durch einen spezifischen Wert dividiert (zu finden in einer 8·8-*Quantisierungsmatrix*), und nur das ganzzahlige Divisionsergebnis ohne Rest wird aufbewahrt. Viele Koeffizienten werden dabei zu 0 oder 1. Dieser Schritt macht das Gros der Kompression und des Qualitätsverlustes aus; es ist durch die Wahl der Quantisierungsmatrix steuerbar.

5 Linearisierung: Die quantisierten Koeffizienten werden im Zickzack (0,0)-(0,1)-(1,0)-(2,0)-(1,1)-(0,2)-… linearisiert. Dabei wird der (0,0)-Wert („DC-Wert") differentiell gegenüber demjenigen des voraufgehenden 8·8-Blockes codiert.

6 Entropiecodierung: Dieser mehrstufige *verlustfreie* Kompressionsschritt besteht aus: Lauflängencodierung für Nullketten (Folgen des Wertes 0) und VLI-Codierung aller Nicht-Null-Koeffizienten außer dem ersten. Bei der VLI-Codierung (VLI = *variable length integers*) werden häufige Werte durch kurze Bitfolgen codiert. Das Codewort-Ende ist beim Decodieren nicht erkennbar, die Länge muß deshalb separat übertragen werden. Es wird angenommen, daß nach jedem VLI-codierten Wert eine Nullkette folgt, gegebenenfalls der Länge Null. Auf die VLI-Codierung folgt noch eine Huffman-Codierung der Paare (Nullketten-Länge, Länge des folgenden VLI-Wertes).

Der Decodierer führt die inversen Schritte durch. Er berechnet $f(k, l)$ aus $F(u,v)$ nach der inversen DCT gemäß der Formel

$$f(k,l) = \frac{2}{N} \left[\sum_{u=0}^{N-1} \sum_{v=0}^{N-1} C(u) C(v) F(u,v) \cos\frac{u\pi(2k+1)}{2N} \cos\frac{v\pi(2l+1)}{2N} \right] \text{ (mit } N = 8)$$

JPEG spezifiziert eine Vielzahl von Varianten, z.B. um bei der Decodierung frühzeitig einen groben Bildeindruck zu erhalten oder um den Kompressionsfaktor dynamisch einem Wunschwert anpassen zu können.

Das von CompuServe standardisierte GIF-Format basiert eigentlich auf dem verlustfreien LZW-Verfahren. Für Festbilder kann es insofern als verlustbehaftet angesehen werden, als

nur 256 Farben zugelassen sind; die theoretisch möglichen 2^{24} Farben eines Bildes müssen entsprechend quantisiert werden (siehe Kapitel E1.8.8).

Wavelet-Transformation. Diese beruht anstelle von Sinusschwingungen auf anderen, nicht periodischen Klassen von Funktionen [Ruskai 92], die nur in einem Intervall um den Ursprung signifikant von Null verschieden sind und durch Parameterwahl in der Frequenz variiert werden können. Bild 8a zeigt einen Vertreter einer solchen Klasse; er kann in diesem Fall durch drei Amplitudenwerte charakterisiert werden. Die Transformation selbst erfolgt durch sog. QMF-Systeme (*quadratic mirror filter*) aus Hoch- und Tiefpässen mit speziellen Eigenschaften, die durch geeignete Wahl des genannten Parameters bestimmt werden. Ein Bild B wird zeilenweise mit einem Tiefpaß gefiltert; zur Darstellung des Resultats B_T reicht die halbe Pixelmenge aus, da sich Pixelwerte nun langsamer ändern; andererseits wird auf B ein Hochpaß angewendet, es werden also nur die hochfrequenten Bildanteile B_H betrachtet; die Anzahl der pro Zeile zu ermittelnden Koeffizienten ist wiederum halb so groß wie die Anzahl der Pixel einer Zeile in B, so daß die Datenmenge von B_T plus B_H der von B entspricht. Aus B_T werden durch vertikalen Tief- bzw. Hochpaß die Bilder B_{TT} und B_{TH} erzeugt, die Anzahl der Zeilen halbiert sich jetzt. Analog werden B_H, B_{HT} und B_{HH} erzeugt, also insgesamt vier Bilder mit je einem Viertel Größe (verlustfreie Transformation). B_{TT} repräsentiert das verkleinerte, geglättete Original, B_{TH} und B_{HT} speichern Details in der Horizontalen bzw. Vertikalen und B_{HH} speichert diagonale Strukturen. Das Verfahren wird rekursiv auf B_{TT} angewendet, Bild 8b veranschaulicht die Ergebnisse des zweiten Durchlaufs. Kompression entsteht wie üblich durch Quantisierung. In die diskrete Wavelet-Transformierte *DWT* eines Bildpunktes gehen (nach Herstellung von Symmetrie) nur wenige Nachbarpunkte ein, sie kann daher sukzessive auf eine ganze Bildzeile angewendet werden statt wie die DCT kachelweise auf z.B. 8·8 Punkte. Die bei DCT-basierten Verfahren störenden Klötzchen-Effekte und Moirés bei starker Kompression entfallen dadurch.

Offensichtlich eignen sich Wavelets sehr gut für sukzessiven Bildaufbau mit zunehmendem Detaillierungsgrad; das Bildformat muß für N-fache Filterung am Rand auf die Größe 2^N mit Null-Pixeln aufgefüllt werden. Durch geeignete Wahl des Wavelet-Typs lassen sich sehr große Kompressionsgrade bei hoher Qualität und mit DCT vergleichbarem Aufwand erzielen. Im MPEG-4-Standard sind Wavelets zur Kompression von Texturen vorgesehen. JPEG2000 stützt sich als Nachfolger von JPEG ganz auf die DWT, unterstützt dennoch optional kachelweise Kompression, außerdem sogenannte ROIs (regions of interest), die schwächer als andere Bildbereiche komprimiert werden.

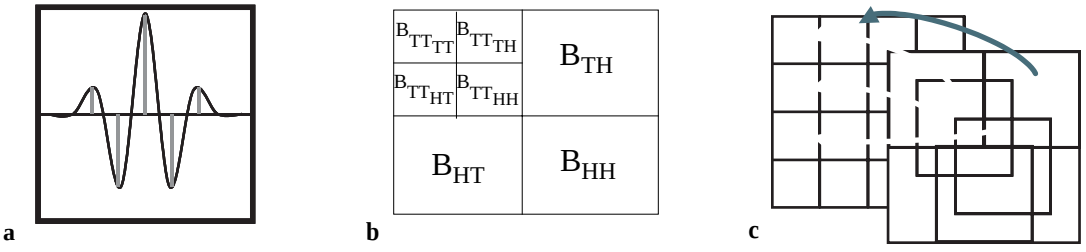

Bild 8 Wavelet-Kompression **a** Wavelet, **b** QMF-System nach dem zweiten Durchlauf und Fraktale Bildkompression **c** Quellblöcke (rechts) und Zielblöcke (links)

Fraktale Verfahren. Sie ermöglichen teilweise spektakuläre Kompressionsfaktoren, erfordern aber bei der Kompression sehr hohen Aufwand [Fisher 95]. Für ihr Verständnis

seien zunächst Schwarzweißbilder betrachtet; deren schwarze Teile können als Punkte in einem zweidimensionalen Raum aufgefaßt werden (bei analogen Bildern ein Ausschnitt aus $\mathbb{R}^2$, bei gerasterten d.h. digitalen Bildern sei er als Ausschnitt aus $\mathbb{Z}^2$ bezeichnet). Affine Abbildungen (wie Drehung, Rotation, Spiegelung) in diesem Raum werden *Kontraktionen* genannt, wenn für zwei beliebige Urbild-Punkte im Raum die Abbilder dieser Punkte geringeren Abstand haben; rechteckige Ursprungsmengen werden dabei in kleinere Rechtecke abgebildet. Der Banachsche Fixpunktsatz besagt in diesem Zusammenhang folgendes: wendet man eine Menge $W = \{w_i\}$ von Kontraktionen auf ein beliebiges Bild B an, erzeugt dabei Bild B', wendet dann W wieder auf B' an – und das unendlich oft, so entsteht am Ende unabhängig vom verwendeten Ausgangsbild B immer dasselbe Bild, der sog. *Attraktor* (auch: das Fraktal). Bei gerasterten Bildern tritt ab einer gewissen Anzahl von Iterationen keine Veränderung mehr ein, der Attraktor wird also nach endlich vielen Schritten erreicht. Der Raum (d.h. die Ursprungsmenge als Ausschnitt aus $\mathbb{Z}^2$, hier: die Menge aller Bilder einer bestimmten Größe) zusammen mit der Menge W wird als iteriertes Funktionensystem IFS bezeichnet. Während in einem IFS jede Kontraktion das gesamte Bild transformiert, hat bei den für fraktale Kompression wichtigen *lokalen IFS* jede Kontraktion nur einen Bild*ausschnitt* als Ursprung – der auf einen kleineren Bildausschnitt abgebildet wird. (Lokale IFS haben meist mehrere Attraktoren, relevant ist hier der „größte“; er hat die Größe des Bildes selbst.)

Die zentrale Frage lautet nun: ist es möglich, bei gegebenem (zu komprimierenden) Bild B ein lokales IFS zu konstruieren, dessen Attraktor eben dieses Bild B ist (oder zumindest ein für die menschliche Wahrnehmung ähnliches)? Dann nämlich brauchten als komprimiertes Bild nur die Bildgeometrie (der „Raum“) und die Daten der Kontraktionen gespeichert zu werden. Der Schlüssel zu der Frage ist der Collage-Satz: möchte man ein lokales IFS wie beschrieben konstruieren, so genügt es, die Menge W der Kontraktionen so zu wählen, daß sie B bei *einmaliger* Anwendung nahezu in sich selbst überführen. Für die Suche nach geeigneten Kontraktionen teilt man in der Praxis B in Quadrate (Zielblöcke) ein und sucht für jeden Zielblock einen (größeren) Quellblock, der ihm bis auf Affinität sehr ähnlich ist, wie in Bild 9c angedeutet. Die Koordinaten von Quell- und Zielblock sowie die Parameter der affinen Abbildung bestimmen die zugehörige Kontraktion. Diese Suche nach Selbstähnlichkeiten in B beginnt man mit möglichst großen Zielblöcken. Jeder Zielblock, für den man nicht genügend ähnliche Quellblöcke findet, wird in kleinere Zielblöcke unterteilt. Je größer die Zielblöcke sind, für die man geeignete Quellblöcke findet, desto kleiner ist die Menge W und desto größer der Kompressionsgrad. (Die Zielblockmenge von W muß ja das ganze Bild abdecken.) Die Dekompression startet mit einem beliebigen (z.B. schwarzen) Bild, besteht aus iterativer Anwendung der Menge W und endet, wenn sich das Bild nicht mehr (wesentlich) ändert. Das für Schwarzweißbilder erklärte Verfahren wird auf Grauwertbilder erweitert, indem die Kontraktionen für jeden Urbildpunkt nicht nur die Zielkoordinaten bestimmen, sondern auch die Änderung von Helligkeit und Kontrast. Farbbilder werden in drei Ebenen unterteilt (z.B. R, G, B), jede Ebene wird als Grauwertbild aufgefaßt.

4.3.4 Videokompression

Videos werden traditionell als schnelle Abfolge von Festbildern modelliert. Die MPEG-Standards (MPEG = *motion picture experts group*) und H.26x-Standards von ISO und

ITU ähneln daher dem beschriebenen JPEG-Standard, ergänzt um Vorhersagen über die Verschiebung von Objekten (näherungsweise: quadratischen Bildausschnitten).

Frame-Typ	Vorhersage für Pixel mit Koordinate k	zu übertragender Bildinhalt	Bewegungs-vektor
I	$I^*(k) = 128$	$I(k)$-128	-
P,B_	$PB^*(k) = IP_-(k + mv_-)$	$PB(k)$-$PB^*(k)$	mv_-
B_+	$B^*(k) = IP_+(k + mv_+)$	$B(k)$-$B^*(k)$	mv_+
B_{-+}	$B^*(k) = 0.5\,[IP_-(k + mv_-) + IP_+(k+mv_+)]$	$B(k)$-$B^*(k)$	mv_-, mv_+

Bild 9 Frame-Typen und deren Codierung bei MPEG
IP: lies „I- oder P-Frame“; PB: lies „P- oder B-Frame“
- bzw. +: lies „bezogen auf vorhergehenden bzw. nachfolgenden IP“
*: kennzeichnet Vorhersagewert; k: Pixelkoordinate der Form (x, y) im Makroblock

Drei *Bildtypen* (*frame types*) werden unterschieden und entsprechend Bild 9 codiert:

- *I-Frames* (*intra-coded*) werden ähnlich zu JPEG ohne Bewegungsvektoren codiert.
- *P-Frames* (*predicted*) werden aus vorangegangenen I- oder P-Frames vorhergesagt. Pro 16·16-Makroblock wird ein Bewegungsvektor ermittelt. Vorhersage für den Inhalt eines Bildpunktes ist derjenige Bildpunkt des vorangegangenen I- oder P-Frames, der laut Bewegungsvektor an diese Stelle „gewandert“ sein müßte. Übertragen wird die stark quantisierte, d.h. für kleine Werte entfallende Differenz zur Vorhersage.

B-Frames (*bidirectional*) werden sowohl vom vorhergegangenen als auch vom nachfolgenden I- oder P-Frame aus vorhergesagt, also über *zwei* Bewegungsvektoren. Da die Abfolge von I-, P- und B-Frames vorher festgelegt wird (z.B. IBBPBBPBBI…) und da Bildschnitte zwischen vorhergesagten und Bezugs-Frames Bewegungsvektoren sinnlos machen, gibt es noch zwei Varianten der B-Frames, bei denen nur rückwärts (also bezüglich des nachfolgenden I- oder P-Frames) oder nur vorwärts (wie bei P-Frames, allerdings meist mit noch gröberer Quantisierung) vorhergesagt wird.

MPEG wurde für eine Datenrate von 1.2 MBit/s optimiert und erreicht dabei ungefähr Videorecorder-Qualität. Das neuere MPEG-2 zielt auf 5 MBit/s und mehr und läßt HDTV-Qualität zu. MPEG-4 wurde 1999 erstmals standardisiert und enthält Verfahren für sehr niedrige Datenraten, ähnlich den ITU-Standards H.261, H.263 und H.264. Die letzten beiden wurden in MPEG-4 integriert. H.261 zielt auf Videophonie und Vielfache des ISDN-B-Kanals mit 64 KBit/s und kennt keine B-Frames. H.263 und H.264 spiegeln jeweils den neuesten Forschungsstand zum Zeitpunkt der Standardisierung wider: sie sind ausgefeilter, erfordern mehr Rechenaufwand, zielen auf noch kleinere Datenraten *und* weit größere bis hin zu HDTV. Da sie auch auf gespeicherte Videos angewendet werden, sind B-Frames sinnvoll und enthalten. Mehr dazu siehe Kapitel E3.

Da unterschiedliche Frame-Typen sowie Art und Dynamik des codierten Inhalts die Kompressionsrate beeinflussen, werden Puffer und Rückkopplungen – z.B. Wechsel der Quantisierungsmatrix – verwendet, um konstante Bitraten zu erzeugen. Darüber hinaus bezeichnet „Skalierung“ die Anpassung einer Videoübertragung an die verfügbare Bandbreite – beim Internet idealerweise sogar während der Übertragung. Dazu wird der Videostrom so komprimiert, daß ein Basis- und gegebenenfalls mehrere die Qualität verbessernde Teilströme entstehen, die dynamisch zu- oder abgeschaltet werden können. Verfah-

ren, die feingranulare Anpassung statt grober Qualitätsstufen erlauben, sind Gegenstand der Forschung.

Allgemeine Literatur

Bell, T. C.; Cleary, J. G.; Witten, I. H.: Text Compression. Englewood Cliffs: Prentice-Hall 1990

Gibson, J. D.; Berger, T.; Lindbergh, D.; Baker, R. L.: Digital compression for multimedia – principles and standards. San Francisco: Morgan Kaufmann 1998

Nelson, M.; Gailly J.-L.: The data compression book. Englewood Cliffs: Prentice Hall 1996

Richardson, I. E.: Video Codec Design, John Wiley & Sons, März 2002

Salomon, D.: Data compression: the complete reference. 2nd ed. Heidelberg: Springer 2001

Sayood, K.: Introduction to data compression. San Francisco: Morgan Kaufmann 2000

Spezielle Literatur

[Burrows 94] Burrows, M.; Wheeler, D. J.: A block-sorting lossless data compression algorithm. Digital Systems Research Center (SRC), Research Report 124

[Fisher 95] Fisher, Y. (Ed.): Fractal image compression – theory and application. New York: Springer 1995

[Huffman 52] Huffman, D. A.: A method for the construction of minimum redundancy codes. Proceedings of the Institute of Radio Engineers 40 (1952) 1098–1101

[IBM 88] IBM Journal of Research and Development 32/6 (1988) 717–795

[JPEG 00] www.jpeg.org/jpeg2000 Web-Seite zu JPEG2000

[Rissanen 79] Rissanen, J. J.; Langdon, G. G.: Arithmetic coding. IBM Journal of Research and Development 23/2 (1997) 149–162

[Ruskai 92] Ruskai, M. B; et al.: Wavelets and their applications. Boston: Jones and Bartlett 1992

[Welch 84] Welch, T. A.: A technique for high-performance data compression. IEEE Computer 17/6 (1984) 8–19

[Witten 87] Witten, I. H.; Neal, R. M.; Cleary, J. G.: Arithmetic coding for data compression. Communications of the ACM 30/6 (1987) 520–540

[Ziv 77] Ziv, J.; Lempel, A.: A universal algorithm for sequential data compression. IEEE Trans. Information Theory 23/3 (1977) 337–343

[Ziv 78] Ziv, J.; Lempel, A.: Compression of individual sequences via variable-rate coding. IEEE Trans. Information Theory 24/5 (1978) 530–536