Bildkompression mit endlichen Automaten

A. Schrader
Universität-Gesamthochschule Siegen
Fachbereich 12 - Elektrotechnik und Informatik
Programmiersprachen
57068 Siegen

Zusammenfassung

Das 'multimediale Zeitalter' hat zu einer ständig wachsenden Integration von Bilddaten bei der Übertragung und Speicherung von Informationen geführt. Die Begrenzungen der Kapazitäten in Übertragungskanälen und Speichermedien erforderte dabei schon früh die Entwicklung geeigneter Kompressionsalgorithmen.

In den klassischen Verfahren der ersten Generation werden Kenntnisse oder Vermutungen über die statistische Verteilung der Daten zur Konstruktion eines redundanzreduzierenden Codes verwendet. Der zweidimensionale Charakter der Bilddaten bleibt dabei im wesentlichen unberücksichtigt.

In diesem Artikel wird die Codierung von endlichen diskreten Bilddaten mit Hilfe von endlichen Automaten beschrieben. Der Bildinhalt wird dabei als endliche Sprache interpretiert, die sich aufgrund von zweidimensionalen rekursiven Selbstähnlichkeiten ergibt und ein deterministischer endlicher Automat als Akzeptor dieser Sprache konstruiert. Dabei werden einige heuristische Ansätze zur effizienten Speicherung der Automatentabelle mit Methoden der ersten Generation vorgestellt und die Ergebnisse dieses Ansatzes anhand von Bildbeispielen demonstriert.

1 Einführung

Die technische Behandlung von Bildern in Rechnersystemen erfordert eine Beschreibung des Bildinhaltes durch eine geeignete digitale Repräsentation. Dies kann zum Beispiel die serielle Angabe der diskreten Intensitäten der Helligkeitswerte aller Pixel sein (Bitmap); eine andere Möglichkeit besteht in der Angabe der grafischen Primitive, aus denen das Bild zusammengesetzt ist (bspw. in Grafikprogrammen). Für die unterschiedlichen Aufgaben der digitalen Bildverarbeitung gibt es jeweilige Qualitätskriterien, die eine bestimmte Codierung gegenüber anderen Alternativen vorteilhaft erscheinen lassen. So ist z.B. die Quadtree-Codierung von quadratischen Bildern eine geeignete Darstellung für die Anwendung grundlegender Operationen der *Bildverarbeitung*, wie z.B. das Skalieren. Für die *Bilderkennung* ist die Beschreibung der Bildstruktur mit Hilfe von grammatikalischen Methoden wichtig.

Hier wollen wir die Möglichkeiten der Beschreibung eines Bildes mit Methoden der formalen Sprachen für die *Datenkompression* nutzen. Ausgangspunkt unserer Überlegungen ist dabei eine zweidimensional in einem kartesischen Raster mit äquidistanten Rasterpunkten angeordnete Menge P von $n \times n$ Bildpunkten (Pixel), denen als Intensitätswert ein Element aus der Menge $\{0,1\}$ zugeordnet wird (Binärbild, oder Bitmap mit 2 Graustufen):

$$P : [0 \ldots n-1] \times [0 \ldots n-1] \rightarrow \{0,1\}.$$

Die semantische Interpretation ist durch eine bijektive Abbildung

$$Sem : \{0,1\} \rightarrow \{schwarz, weiss\}$$

gegeben. Dabei ist es zunächst zweitrangig, ob die Bitmap durch Digitalisierung und Quantisierung des Ausgangsignals eines bildgebenden Sensors oder in einem Grafikprogramm entstanden ist. Die Behandlung von Bildern mit mehreren Graustufen oder von Farbbildern läßt sich durch eine entsprechende Zerlegung in mehrere Binärbilder (bspw. durch Extraktion von Bitebenen) erreichen.

Für die Übertragung oder Speicherung von digitalen Bildern müssen diese als Nachricht interpretiert und durch ein geeignetes Verfahren codiert werden. Als kanonische Form bei der Codierung von Bitmaps bietet sich die serielle Angabe der Intensitätswerte durch Codes einheitlicher Länge an. Für Binärbilder reicht die Verwendung von einem Bit pro Pixel aus. Die zweidimensionale Anordnung der Pixel kann

durch ein zeilenweises Abtasten in eine serielle Folge umgewandelt werden. Dieses Verfahren findet in vielen Grafikformaten Anwendung, wobei allerdings weitere Parameter (wie z.B. Kantenlänge, Anzahl der Graustufen, usw.) mit abgelegt werden. Die eindeutige Decodierbarkeit ergibt sich durch die Eindeutigkeit von Position und Codelänge jedes einzelnen Parameters im Datenstrom.

Hauptnachteil dieser einfachen Codierung ist ihr hoher Speicherplatzbedarf. Bei heute üblichen True-Color-Bildern der Auflösung 1024×1024 Pixel ergibt sich auf diese Weise eine Datenmenge von 3 MByte. Speicherplatz und Datenübertragungskapazität sind knappe Ressourcen und so sind seit mehreren Jahrzehnten Verfahren entwickelt worden, die eine Reduzierung der Datenrate erreichen. Dabei unterscheidet man grundsätzlich zwischen verlustlosen und verlustbehafteten Techniken.

Wichtige Anwendungsgebiete der *verlustbehafteten* Verfahren sind bspw. die Codierung von Bildsequenzen bei der Bildtelefon-Übertragung oder auch die Erzeugung progressiver Bildqualitätsstufen bei Zugriffen auf Bilddatenbanken. Hier finden Verfahren der sogenannten zweiten Generation Anwendung, die neben den statistischen Eigenschaften der Quelle auch Aspekte des menschlichen visuellen Wahrnehmungsprozesses berücksichtigen. So wird bspw. im JPEG-Verfahren [10] die Tatsache ausgenutzt, daß das menschliche Auge unempfindlicher auf Störungen in den hochfrequenten Bildteilen reagiert als auf Störungen in den niederfrequenten Bereichen. In jüngster Zeit haben zwei verwandte Verfahren die Aufmerksamkeit erregt, die die rekursive Selbstähnlichkeit von bildlichen Darstellungen natürlicher (und damit oft fraktaler) Objekte mit den Möglichkeiten der zweiten Generation verbinden und zur Datenkompression ausnutzen: Die IFS-Codierung von Barnsley [7] und die WFA-Methode von Culik [12]. Wenn die verlustbehafteten Verfahren auch hohe Kompressionsraten erreichen, so geht dies doch oft mit einer starken Verfälschung des eigentlichen Bildinhaltes einher (hier sei nur der Kacheleffekt bei der JPEG-Codierung genannt [13]).

Manche Anwendungen fordern von den Kompressionsalgorithmen die absolute Erhaltung der Bildqualität und zwar nicht nur der subjektiv empfundenen (die auch bei den obengenannten Verfahren bei mittleren Kompressionsfaktoren noch in vertretbaren Größenordnungen liegt), sondern gerade bei Anwendungen mit dem Ziel der automatischen Bildauswertung auch die Erhaltung der objektiven Bildqualität (z.B. die Speicherung und Übertragung von Röntgenbildern in medizinischen Bildverarbeitungs- und Archivierungsanlagen. Hier kann ein einzelner verfälschter Bildpunkt u.U. die automatische oder manuelle Diagnose verändern.) In solchen Anwendungen gilt es also, *verlustlos* zu komprimieren.

In diesem Artikel werden nun die Möglichkeiten untersucht, rekursive Selbstähnlichkeiten der zweidimensionalen Daten zur *verlustlosen* Kompression auszunutzen. Dabei wird von der kanonischen Form der Bildcodierung abgewichen und die Bildinformation in eine endliche Menge von Wörtern über einem endlichen Alphabet abgebildet, wobei die notwendigen Bedingungen für eine eindeutige Decodierbarkeit berücksichtigt werden. Aufgrund der Endlichkeit der Menge der Pixel im Bild P kann durch eine solche Codierung auch nur eine endliche Sprache entstehen. Endliche Sprachen gehören zu der regulären Typ-3-Sprachklasse der Chomsky-Hierarchie, für deren Beschreibung sowohl generative Verfahren (reguläre Grammatiken) als auch analysierende Verfahren (endliche Automaten) existieren. Somit läßt sich zu jedem Bild leicht ein endlicher Automat konstruieren, der genau die sich aus der Codierung ergebende endliche Sprache akzeptiert. Datenkompression kann dann erreicht werden, wenn die zur Abspeicherung dieses (möglichst minimal gewählten) Automaten notwendige Datenmenge kleiner ist, als die zur kanonischen Darstellung der Bitmap.

Der Artikel ist folgendermaßen gegliedert: Kapitel 2 beschreibt den klassischen Ansatz der Verfahren der ersten Generation, die mit informationstheoretischen Methoden optimale Codes erzeugen und zeigt die Schwierigkeiten dieses Ansatzes bei Unkenntnis des stochastischen Bilderzeugungsprozesses. Kapitel 3 erläutert den Ansatz, das Bild mit Hilfe der Quadtree-Zerlegung als endliche Sprache zu betrachten und demonstriert die Verwendung eines endlichen Automaten als zugehörigen Akzeptor dieser Sprache. Das Kapitel 4 beschreibt einige heuristische Ansätze zur effizienten Codierung der Automatentabelle. Die Ergebnisse dieser Ansätze werden für die in Kapitel 5 vorgestellten Beispielbilder mit herkömmlichen Verfahren und theoretischen Zahlen verglichen.

2 Der klassische Ansatz - Optimalcodierung der Shannon'schen Quelle mit Präfixcodierern (Huffman)

Die klassische *Informationstheorie* nach Shannon, die streng genommen eigentlich eher eine *Kommunikationstheorie* ist, stellt ein Maß für die Informati-

on zur Verfügung, die zwischen zwei Systemen übermittelt wird. Der Sender wird dabei als diskreter stochastischer Zufallsprozeß (Nachrichtenquelle) interpretiert, der über einen endlichen Vorrat an Nachrichten verfügt. Die Nachrichen selber setzen sich aus einer Folge von Zeichen a_i aus einem endlichen Alphabet $A = \{a_1, a_2, \ldots, a_n\}$ zusammen. Diese Zeichen sind die Elementarereignisse des Zufallsprozesses und treten mit bestimmten Wahrscheinlichkeiten $p(a_i)$ auf. Dabei gilt die Vollständigkeit: $\sum_{i=1}^{n} p(a_i) = 1$. Die Information wird in diesem Modell gleichgesetzt mit der Vorhersagbarkeit eines Ereignisses. Aus einer Reihe von Forderungen an die Eigenschaften einer solchen Informationsfunktion I ergibt sich eine logarithmische Funktion [2]: $I_i = -\operatorname{ld}\ p(a_i)$.

Für die Basis 2 im Logarithmus (ld = logarithmus dualis) wird die Maßeinheit der Information [1 Bit] genannt. Wir können somit den mittleren Informationsgehalt einer Quelle durch Berücksichtigung der Auftrittswahrscheinlichkeiten der einzelnen Zeichen bestimmen:

$$H = -\sum_{i=1}^{n} p(a_i) \cdot \operatorname{ld}\ p(a_i)\ \text{[Bit]}.$$

Aufgrund der formalen Ähnlichkeit zu der entsprechenden Formel in der stochastischen Thermodynamik wird diese Größe auch *Entropie* genannt.

Die obengenannten Überlegungen gelten dabei zunächst für Quellen, die kein Gedächtnis besitzen, d.h. bei denen $p(a_i)$ für ein Zeichen unabhängig von allen zuvor ausgesendeten Zeichen a_j ist.

So gesehen besteht ein endliches, binäres Bild aus einer Nachricht mit einer endlichen Anzahl von Zeichen, die von einer stochastischen Nachrichtenquelle (dem Bilderzeugungsprozeß) produziert wurden, wobei die Art und Anzahl der Zeichen weiter unten diskutiert wird.

2.1 Quellencodierung

Zur Speicherung des Bildes in einem digitalen Rechnersystem wird eine (binäre) Codierung der Nachricht notwendig. Die injektive Abbildung der (Quellen-)Codierung

$$C : A \to \{0,1\}^* \backslash \{\epsilon\}$$

ordnet jedem Zeichen des Alphabets einen eindeutigen Code zu. Wir gehen dabei im folgenden stets von einem störungsfreien, idealisierten Übertragungskanal (oder Speichermedium) aus und können daher Aspekte der Kanalcodierung unberücksichtigt lassen. Da wir bei der Speicherung oder Übertragung einer Zeichenfolge auch eine Codefolge erzeugen, muß deren eindeutige Decodierbarkeit garantiert sein. Dies läßt sich prinzipiell auf drei verschiedene Arten erreichen:

- Alle Codes besitzen die gleiche Codelänge.
- Man verwendet spezielle Trennzeichen zwischen den Codewörtern.
- Man verwendet einen Präfixcode, bei dem garantiert ist, daß kein Codewort der Anfang (Präfix) eines anderen Codewortes ist.

Da gezeigt werden kann, daß jeder eindeutig decodierbare Code durch einen Präfixcode mit gleicher Codewortlänge ersetzt werden kann ([9], S. 35), kann man sich auf die Behandlung ebensolcher beschränken, wenn die Codelänge einziges Kriterium ist.

Für die (Bilddaten-)Kompression besteht die Aufgabe nun darin, einen Code zu finden, der eine Codierung der (Bild-)Daten mit möglichst geringer Länge ermöglicht. Es kann gezeigt werden, daß die mittlere Codelänge

$$n(C) = \sum_{i=1}^{n} p(a_i) \cdot |C(a_i)|$$

die Entropie H der Quelle nicht unterschreiten kann. Für die Effizienz eines Codes

$$\text{Eff}(C) = \frac{H}{n(C)}$$

gilt daher stets ([9], S. 131):

$$\text{Eff}(C) \leq 1.$$

Ziel der Datenkompression ist folglich die Minimierung der Redundanz

$$R = n(C) - H.$$

Die Redundanz ist damit der objektiv unwichtige Teil der Codierung, d.h. man kann einen anderen Code wählen, dessen mittlere Codewortlänge kürzer ist, der somit also eine geringere Redundanz besitzt und trotzdem die gleiche Information übermittelt. Die *objektive* Redundanz ist dabei nicht mit der *subjektiven* Irrelevanz einer Information zu verwechseln ([11], S. 83f.).

2.2 Optimale Präfixcodes

Wenn nun für die verlustlose Kompression die untere Grenze einer Codierung durch die Entropie der Quelle gegeben ist, so ist damit noch nichts

darüber ausgesagt, wie ein entsprechender optimaler Code zu konstruieren ist. Es wurden allerdings schon sehr früh drei Algorithmen entwickelt, die relativ nahe an der optimalen Codierung liegen (Shannon-, Shannon/Fano- und Huffman-Codierung, [11], S. 112f). Während die beiden erstgenannten gute Laufzeiten aufweisen, kann der Huffman-Algorithmus für viele Wahrscheinlichkeitsverteilungen (WV) bessere Werte liefern. Der Huffman-Algorithmus bildet einen binären Codierungsbaum, indem rekursiv die beiden Zeichen mit geringster Wahrscheinlichkeit zusammengefaßt werden [1]. Auf diese Weise erzeugt der Huffman-Algorithmus einen optimalen Präfixcode, d.h. es gibt keinen anderen Präfixcode mit geringerer Codelänge für die gegebene Quelle, wenn jedem Quellenzeichen genau ein Code zugeordnet wird. In ([11], S. 115–118) wird anschaulich diskutiert, bei welchen Quellen sich die Optimalität gegenüber der einfacheren Shannon- oder Shannon/Fano-Codierung bemerkbar macht. Für eine Quelle mit nur einem Zeichen ist keine Codierung sinnvoll, bei $|A| = 2$ kann jedem Zeichen 0 oder 1 zugeordnet werden, wobei sich immer eine mittlere Codelänge von 1 ergibt. Für $|A| = 3$ und $|A| = 4$ liefern die Algorithmen gleiche Codelängen und erst für Nachrichtenquellen mit mehr als 4 Zeichen zeigt sich die Überlegenheit des Huffman-Codes, wenn auch nur für bestimmte WV. Der Aufbau des Huffman-Codierungsbaumes benötigt dabei für eine Quelle mit $|A| = n$ Zeichen eine Laufzeit aus der Klasse $O(n \cdot \mathrm{ld}\ n)$.

Die Optimalität eines Präfix-Codes darf nicht darüber hinwegtäuschen, daß die Entropie der Quelle nur selten erreicht wird. Der Grund hierfür liegt in der Tatsache, daß jedem einzelnen Zeichen ein binäres Codewort zugeordnet wird. Es lassen sich somit keine rationalen Anteile, sondern nur ganzzahlige Bit-Werte auf die Zeichen verteilen. Dieser Nachteil kann auf zwei verschiedene Weisen aufgehoben werden:

1. Durch die Verwendung eines Verfahrens, das alternativ zu Präfix-Codes auch nichtganzzahlige Bit-Werte zuordnen kann. Hier ist die arithmetische Codierung zu nennen, die ein Zahlenintervall der reellen Achse in Abhängigkeit der WV der Quelle in immer genauere Teile zerlegt und die Trenngrenze als Code verwendet. Nachteil dieses Verfahrens ist der erhöhte Implementierungsaufwand. Testergebnisse haben gezeigt, daß die arithmetische Codierung in den meisten Fällen die Huffman-Codierung um weniger als 10% übertrifft ([6], S. 538).

2. Durch die Zusammenfassung von jeweils k Zeichen der Quelle zu n^k Pseudozeichen (Blöcken). Die Blockcodierung nähert sich für $k \to \infty$ der Entropie der Quelle immer näher an.

Der erste Ansatz wird in diesem Artikel nicht weiter verfolgt und kann durch den 10%-Effekt implizit berücksichtigt werden. Der zweite Ansatz trifft in der praktischen Realisierung auf zwei kritische Probleme:

- Für die Decodierung der Nachricht muß der Codebaum im Decodierer bekannt sein. Will man nicht immer die gleiche WV verwenden (was wenig sinnvoll wäre), ist man gezwungen, den Codebaum zusätzlich zur eigentlichen Information mit zu übertragen. Dadurch wird sowohl bei kurzen Nachrichten als auch bei langer Blockbildung (wegen des exponentiellen Wachstums des Codebaums) die Kompressionsrate gesenkt. Einen Kompromiß kann man mit den in diesem Artikel ebenfalls unberücksichtigten adaptiven Verfahren erreichen, bei denen der Codebaum im Laufe der Codierung sowohl im Codierer als auch im Decodierer ständig dem aktuellen Verlauf der Nachrichtenstatistik angepaßt wird.

- Die Blöcke können nicht beliebig lang werden, da sie stets kleiner als die endliche Nachricht (das Bild) sein müssen. Außerdem ist die WV der Quelle i.a. gar nicht bekannt und kann nur durch die relative Häufigkeit der Zeichen bestimmt werden. Je länger die Blockbildung ist, umso ungenauer wird dadurch aber auch die Abschätzung der Wahrscheinlichkeiten. Im Extremfall besteht die gesamte Nachricht aus einem einzigen Block, dessen rel. Häufigkeit = 1 ist, womit eine sinnvolle Codierung nicht mehr möglich ist.

2.3 Der stochastische Bilderzeugungsprozeß

Kehren wir zurück zu den Binärbildern, so stellt sich die Frage, wieviele Elemente enthält das Alphabet A der Nachrichtenquelle und aus wie vielen Zeichen setzt sich das Bild zusammen? Betrachten wir zunächst zwei Extreme:

1. $A = \{0, 1\}$. Dabei bedeutet das Zeichen $a_i = 0$, daß der Bildpunkt schwarz ist und das Zeichen $a_i = 1$, daß der Bildpunkt weiß ist. Für jeden einzelnen Bildpunkt wird folglich ein Zeichen erzeugt und das Bild enthält somit n^2 einzelne Zeichen, die alle codiert werden müssen. Der optimale Huffman-Code ergibt sich dabei direkt aus der Tatsache, daß es nur zwei unterschiedliche Zeichen gibt und die kleinste Informationseinheit das Bit ist: $C(0) = 0, C(1) = 1$. D.h. jedes Pixel wird mit genau einem Bit codiert und wir erhalten eine Codelänge von n^2 Bit. Diese kanonische Form

wird auch als *Bitmap* bezeichnet und gilt im folgenden als Bezugsgröße für die Bestimmung des erreichten Kompressionsfaktors. Dabei ist die gewählte Codierung unabhängig von der gegebenen Wahrscheinlichkeitsverteilung optimal (siehe obige Ausführungen zur Huffman-Codierung).

2. $A = \{0, 1, \ldots, 2^{n^2} - 1\}$. D.h. wir fassen das Bild als ein einziges Zeichen auf, indem wir die Intensitätswerte der Pixel als Ziffer einer Dualzahl mit n^2 Stellen interpretieren. Dadurch gibt es insgesamt 2^{n^2} verschiedene Bilder (und somit Zahlen) deren Zahlenwert wir als Nachricht interpretieren. Setzen wir voraus, daß alle möglichen Bilder gleichwahrscheinlich vorkommen und eine andere Aussage läßt sich bei der Codierung eines einzigen Bildes über den Bilderzeugungsprozeß schließlich nicht machen, so ergibt sich für die Entropie der Quelle:

$$H = -\sum_{i=1}^{2^{n^2}} p(a_i) \cdot \mathrm{ld}\ p(a_i)\ [\mathrm{Bit}].$$

Mit $p(a_i) = \frac{1}{2^{n^2}}$ ergibt sich damit $H = -\ \mathrm{ld}\ \frac{1}{2^{n^2}}[\mathrm{Bit}] = n^2[\mathrm{Bit}]$. D.h. auch in diesem Fall kann die Datenmenge nicht unter die Bitmap-Größe gesenkt werden. Man kann dieses Vorgehen auch als die Blockbildung mit $k = n^2$ interpretieren.

Während also im 2. Fall aufgrund der Tatsache, daß die Stichprobe nur ein einziges Zeichen enthält, mit Hilfe der relativen Häufigkeiten keine Rückschlüsse auf die Wahrscheinlichkeitsverteilung der Quelle gezogen werden können, hat im 1. Fall die Abschätzung der WV keinen Einfluß auf die erzeugte Codelänge. Das Optimum liegt nun irgendwo zwischen diesen Extremen und hängt wesentlich von der Größe des erzeugten Codebaums ab.

Tabelle 1 zeigt die Entropie der Quelle bei Blockbildung für die Beispielbilder. Die Zeichen $a_1, \ldots, a_n$ der Quelle werden zu neuen Pseudozeichen (z.B. $a'_1 = a_1a_1, a'_2 = a_1a_2, a'_3 = a_2a_1, a'_4 = a_2a_2$ für eine Blocklänge von $k = 2$ und $n = 2$ Zeichen). Für diese Menge $A' = \{a'_1, \ldots, a'_m\}$ berechnet sich die Block-Entropie durch

$$H' = -\sum_{i=1}^{m} p(a'_i) \cdot \mathrm{ld}\ p(a'_i)\ Bit.$$

Dabei ist zu beachten, daß es aufgrund der Pseudokonstruktion dazu kommen kann, daß $m \neq n^k$, da manche Kombinationen in der Bildmenge eventuell überhaupt nicht vorkommen, d.h. $p(a'_i) = 0$ ist. Diese Zeichen haben daher keinen Code und folglich gilt die sonst gültige Ungleichung $H' \leq k \cdot H$ hier nicht. H' berechnet außerdem die Entropie bezogen auf die Pseudozeichen. Da diese k Zeichen enthalten, muß die Pseudoentropie durch die Blocklänge dividiert werden, um zur eigentlichen Entropie bzgl. der Zeichen zu kommen: $H = \frac{H'}{k}$.

Die letzte Spalte in Tabelle 1 zeigt die auf diese Weise korrigierte Blockentropie $H = \frac{H'}{k}$. Man erkennt deutlich die fallenden Entropiewerte bei steigender Blocklänge. Dabei muß aber beachtet werden, daß zusätzlich noch der Codierungsbaum mit abgelegt werden muß. Bei $k = 8$ beginnt sich der oben erwähnte exponentielle Größeneffekt des Codierungsbaumes bemerkbar zu machen, denn er beinhaltet hier schon n^k Codes und für unsere Anwendung mit $n = 2$ bedeutet dies, 256 Codewörter als Huffman-Codebaum mit übertragen zu müssen. Bei $k = 16$ macht die Übertragung mit den schon 65.536 Codewörtern keinen Sinn mehr, auch wenn man nur die tatsächlich vorkommenden Codewörter berücksichtigen würde (dies wären bei 512×512 Pixeln Bildgröße maximal $\frac{262.144}{16} = 16.384$ verschiedene Codes). Wir wollen daher in Kap. 4 die Block-Entropiewerte für $k = 8$ mit den Ergebnissen des unten vorgestellten Verfahrens vergleichen.

3 Interpretation des Bildes als endliche Sprache

Alternativ zur Interpretation des Bildes als serielle Folge von Intensitätswerten (u.U. mit Blockbildung) wollen wir nun die Beschreibung des Bildes P mit Hilfe einer endlichen Sprache L über einem festgelegten Alphabet Σ vornehmen:

$$L \subset \Sigma^*, L \text{ endlich}.$$

Endliche Sprachen lassen sich durch reguläre Ausdrücke beschreiben und sind somit Teilklasse der Chomsky-Typ3-Sprachklasse, deren Beschreibung auch durch die Definition eines entsprechenden endlichen Automaten (FA) erfolgen kann.

Zur Erzeugung einer solchen Bildsprache L muß nun eine bijektive Abbildung ϕ:

$$\phi : [0 \ldots n-1] \times [0 \ldots n-1] \rightarrow L$$

gefunden werden, die jedem Pixel des Bildes ein eindeutiges Codewort $\omega \in L$ zuordnet. Zur Vereinfachung des Zerlegungsprozesses wollen wir im folgenden von der Quadtree-Zerlegung ausgehen, die mit Hilfe des Alphabets $\Sigma = \{0, 1, 2, 3\}$ für jedes Pixel ein Codewort $\omega \in \Sigma^k$ der Länge $|\omega| = k$ erzeugt. Der Parameter k ergibt sich bei dieser Methode direkt aus

Bild	Block-größe	Anzahl Zeichen	Wahrscheinlich-keitsverteilung		Entropie H [Bit]
			p(0)	p(1)	
Cranach	1	2	0.18	0.82	0.67
	2	4	—	—	0.51
	4	16	—	—	0.34
	8	256	—	—	0.38
Duden	1	2	0.22	0.78	0.75
	2	4	—	—	0.61
	4	16	—	—	0.42
	8	256	—	—	0.46
Röntgen	1	2	0.97	0.03	0.19
	2	4	—	—	0.15
	4	16	—	—	0.10
	8	256	—	—	0.11
Schlüssel	1	2	0.09	0.91	0.44
	2	4	—	—	0.30
	4	16	—	—	0.18
	8	256	—	—	0.16

Tabelle 1: Entropie der Beispielbilder

der Größe des (quadratischen) Bildes: $|\Sigma|^k = n^2$ (Bei $n = 512 \Rightarrow k = 9$).

Diese Zuordnung wird erreicht, indem das Bild rekursiv in vier gleichgroße Subquadranten mit einer Zuordnung nach Abb. 1 zerlegt wird:

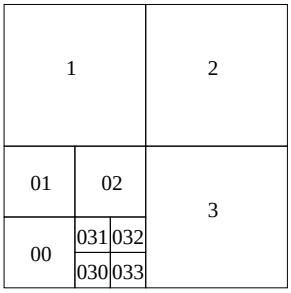

Abbildung 1: Einige Quadtree-Zuordnungen bis $k = 3$

Die Codeworte entstehen durch Konkatenation dieser Präfixe. Für $n = 512$ besitzt ein Teilquadrat bei $k = 9$ die Größe eines Pixels und der Zerlegungsprozeß endet bei dieser Auflösung.

Die Sprache L_P enthält nun alle Codewörter $\omega \in L$, deren Pixel in der Bitmap den Wert 1 besitzen:

$$L_P := \{\omega \in L | P(\phi^{-1}(\omega)) = 1\}$$

Es kann gezeigt werden, daß die Wahl der Werte mit $P(\phi^{-1}(\omega)) = 0$ einen identischen Kompressionsfaktor ergibt [14]. Ergebnis dieses Codierungsprozesses ist somit eine endliche Menge $L_P \subseteq \Sigma^k$ mit $|L_P| \leq n^2$. Diese Menge charakterisiert in eindeutiger Weise das Bild P.

3.1 Codierung mit endlichen Automaten

Endliche Sprachen lassen sich, als Teilklasse der regulären Sprachen, mit Hilfe von endlichen Automaten beschreiben. Wir wollen im folgenden nur deterministische und vollständige endliche Automaten $A = (Q, \Sigma, \delta, q_0, F)$ verwenden, mit Q: endliche Menge von Zuständen, Σ: Eingabealphabet, $\delta : Q \times \Sigma \rightarrow Q$: (totale) Übergangsfunktion, $q_0 \in Q$: ausgezeichneter Startzustand und $F \subseteq Q$: Menge der akzeptierenden Endzustände.

Die Überführungsfunktion ist eine totale Abbildung, d.h. $\forall q \in Q, a \in \Sigma \;\; \exists q' \in Q : (q, a, q') \in \delta$. Sie wird rekursiv erweitert auf die Mehrschrittableitung $\delta^* : Q \times \Sigma^* \rightarrow Q$, wobei $\delta^*(q, \epsilon) := q$ und $\delta^*(q, a\omega) := \delta^*(\delta(q, a), \omega)$, mit $q \in Q, \omega \in \Sigma^*, a \in \Sigma$. Sie beschreibt den Wechsel der Zustände beim Lesen der Eingabe $\omega \in \Sigma^*$. Die von A akzeptierte (Bild-)Sprache ist dann:

$$L(A) := \{\omega \in \Sigma^* | \delta^*(q_0, \omega) \in F\}.$$

Für eine gegebene endliche (Bild-)Sprache $L_P \subseteq \Sigma^*$ läßt sich ein DFA mit einem schlingenfreien Übergangsgraph und einem akzeptierenden (F) und einem nichtakzeptierenden ($\overline{F}$) Endzustand leicht konstruieren, für den gilt: $L(A) = L_P$. Das Bild P läßt

sich aus diesem Automaten rekonstruieren, indem der Automat für alle $\omega \in \Sigma^k \cap L_P$, die einen schwarzen Bildpunkt repräsentieren in den Zustand F und für alle anderen $\omega \in L \setminus L_P$ in den Zustand $\overline{F}$ übergeht. Wir haben auf diese Weise eine Transformation der Bildinformation von der Intensitätsdarstellung in die Automaten-Darstellung gewonnen.

Ein DFA kann durch die Zusammenfassung von äquivalenten Zuständen reduziert werden. Zwei Zustände q_1, q_2 sind äquivalent ($q_1 \equiv q_2$), wenn $\forall \omega \in \Sigma^*$ gilt: $\delta^*(q_1,\omega) \in F \iff \delta^*(q_2,\omega) \in F$. Ein Automat ist dann reduziert, wenn $\forall q_1, q_2 \in Q : q_1 \equiv q_2 \iff q_1 = q_2$ und alle Zustände vom Startzustand q_0 aus erreichbar sind: $\forall q \in Q\ \exists \omega \in \Sigma^*$ mit $\delta^*(q_0,\omega) = q$.

Betrachtet man die Teile des Bildes, die durch einen Zustand repräsentiert werden, so läßt sich die Äquivalenz zweier Zustände als die Ähnlichkeit zweier Bildteile deuten. Jeder Zustand steht im unreduzierten Automaten für einen bestimmten Subquadranten des Bildes. Der akzeptierende Endzustand F steht für ein schwarzes Quadrat (beliebiger Größe, allerdings mit einer Kantenlänge, die einer 2-er-Potenz entspricht.) und der nichtakzeptierende Endzustand $\overline{F}$ steht für ein weißes Quadrat. Interpretiert man den Übergangsgraph als Baum der Ordnung $|\Sigma|$, so sind jeweils die Zustände äquivalent, deren Teilbäume äquivalent sind. Dies wird in Abbildung 2 verdeutlicht. Das Bild P wird durch die Sprache L_P repräsentiert, für die der entsprechende DFA skizziert ist (dabei sind die Übergänge nach $\overline{F}$ zur besseren Lesbarkeit weggelassen.) Man erkennt die Äquivalenz der Zustände q_4 und q_5, die ein identisches Übergangsverhalten in die nachfolgenden Endzustände besitzen. Sie repräsentieren die beiden großen schwarzen Quadrate im rechten oberen Teilquadranten des Bildes und sind somit äquivalent zum Endzustand F. Nach dem ersten Reduktionsschritt erkennt man die Äquivalenz der Zustände $q_2^{'}$ und $q_3^{'}$, die ein identisches Übergangsverhalten in die nachfolgenden Endzustände besitzen. Beide akzeptieren die Worte, die durch den regulären Ausdruck (1|3)(0|1|2|3)* gebildet werden und können somit zu einem neuen Zustand $q_3^{''}$ zusammengefaßt werden. Die rechte Seite von Abbildung 2 zeigt den auf diese Weise reduzierten Automaten, der die gleiche Sprache L_P akzeptiert, wenn vorausgesetzt wird, daß $L_P \subset \Sigma^k$, d.h. die Wortlänge auf $|\omega| \leq k$ beschränkt ist. Während der Zustand $q_3^{'}$ die kleine Diagonale im linken unteren Teilquadranten des Bildes repräsentierte, wurde mit $q_2^{'}$ die große Diagonale im rechten, oberen Teilquadranten dargestellt. Die Ähnlichkeit dieser beiden Bildteile, die sich nur durch einen Skalierungsfaktor voneinander unterscheiden, wird für die Bildung des Zustands $q_3^{''}$ ausgenutzt, der das Prinzip 'Diagonale' repräsentiert. Die Art der Übergänge, die in diesen neuen Zustand $q_3^{''}$ hineinführen, bestimmen durch ihre Zeichenfolge ('02' oder '2') und die Länge dieser Folge die Position und die Größe des Musters. Auf diese Weise wirken sich Ähnlichkeiten im Bild günstig auf die Anzahl der Zustände des Automaten aus. Im folgenden wollen wir stets davon ausgehen, daß ein vollständig reduzierter Automat erzeugt wurde.

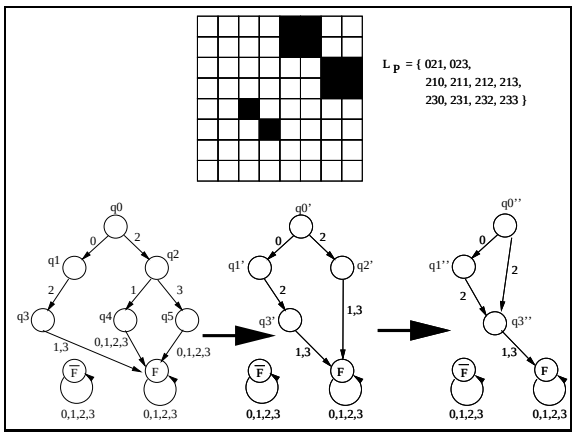

Abbildung 2: Beispielbild mit bis auf Skalierung identischen Bildteilen.

4 Informationsgehalt von Zustandsübergangstabellen

Nachdem im ersten Schritt des Codierungsmechanismus ein DFA mit einer minimalen Zahl von Zuständen erzeugt wurde, stellt sich die Frage der effizienten Codierung des Automaten. Den größten informationstragenden Teil des Automaten stellt die Überführungsrelation dar. Die linke Spalte von Tabelle 2 gibt einen Einblick in die Größenordnungen der Automaten, die aus den Beispielbildern (siehe Kap. 5) erstellt wurden.

Da wir für die Sprache ein vierwertiges Alphabet Σ ($|\Sigma| = 4$) verwendet haben, gibt es für jeden Automatenzustand vier Übergänge zu Folgezuständen. Die Übergänge des akzeptierenden (F) und nichtakzeptierenden ($\overline{F}$) Endzustands müssen nicht explizit codiert werden und so handelt es sich um eine Tabelle mit $(|Q| - 2) \cdot |\Sigma|$ Einträgen.

Bild	Anzahl Zustände $\|Q\|$	Information I_A [Bit]	c.f.
Cranach	4.113	128.520	0.49
Duden	5.108	165.268	0.63
Röntgen	1.656	41.437	0.16
Schlüssel	1.796	47.497	0.18

Tabelle 2: Automatengröße der Beispielbilder

4.1 Automatentabelle als Shannonsche Nachrichtenquelle

Die naheliegendste Möglichkeit der effizienten Codierung der Tabelle besteht in der Interpretation der Tabelle als Nachricht einer stochastischen Quelle mit den Tabelleneinträgen als Zeichen. Das Alphabet der Quelle enthält dabei genau die möglichen $|A| = |Q|$ Folgezustände. Für eine Optimal-Codierung dieser Quelle können wir die Häufigkeiten $h(a_i)$ der einzelnen Alphabetzeichen der Tabelle als Abschätzung für die WV der Quelle verwenden und mit ihrer Hilfe die Entropie der Tabelle bestimmen:

$$H_A = \sum_{i=1}^{|Q|} \frac{h(a_i)}{|\Sigma| \cdot (|Q|-2)} \cdot \mathrm{ld} \left(\frac{h(a_i)}{|\Sigma| \cdot (|Q|-2)} \right) \text{ [Bit].}$$

Der Informationsgehalt der Tabelle ergibt sich somit durch

$$I_A = |\Sigma| \cdot (|Q|-2) \cdot H_A.$$

Die rechten Spalten von Tabelle 2 zeigen die auf diese Weise gebildeten Werte und den Kompressionsfaktor (c.f.), der sich durch Division mit der Bitmap-Größe ergibt. Im Vergleich mit Tabelle 1 zeigt sich, daß die Werte größer sind als die Entropie-Werte der Bitmap bei Blockbildung mit $k = 8$ (der Vergleich ist deshalb möglich, weil die Entropiewerte in Tabelle 1 den Kompressionsfaktoren entsprechen). Im folgenden soll nun gezeigt werden, wie mit Hilfe von einigen heuristischen Ansätzen die offensichtlich in der Tabelle enthaltene Redundanz reduziert werden kann.

4.2 Heuristische Optimierungs-Ansätze

Im folgenden werden zwei heuristische Ansätze vorgestellt, mit denen eine kompakte Speicherung der Automatentabelle erreicht werden kann. Die Ergebnisse werden an den Beispielbildern quantitativ abgeschätzt und mit bestehenden Kompressionsverfahren verglichen.

Wir können die Zustände des Automaten in Äquivalenzklassen nach folgender Relation aufteilen:

$$[Q]_i := \{q \in Q \backslash \{F, \overline{F}\} | \exists \omega \in \Sigma^* : \delta^*(q_0, \omega) = q \wedge |\omega| = i$$

$$\wedge |\omega| \geq |\omega'| \forall \omega' \in \Sigma^* : \delta^*(q_0, \omega') = q\}$$

D.h. wir bilden Schichten im Übergangsgraphen und alle Zustände mit gleicher maximaler Pfadlänge vom Startzustand gehören zur selben Schicht. Für den linken Automaten in Abb. 2 gilt dann z.B. : $[Q]_0 = \{q_0\}$, $[Q]_1 = \{q_1, q_2\}$, $[Q]_2 = \{q_3, q_4, q_5\}$ und $[Q]_3 = \{F, \overline{F}\}$.

Offensichtlich können die Übergänge von Zuständen in $[Q]_i$ nur auf Zustände aus den Schichten $[Q]_j$ mit $j > i$ oder auf $\{F, \overline{F}\}$ zeigen. D.h. die Anzahl der möglichen Folgezustände ist nicht für alle Tabelleneinträge gleich groß. Diese Tatsache wird zu einer effizienten Codierung ausgenutzt. Dazu zeigt Tabelle 3 für die Beispielbilder die Anzahl der Zustände der einzelnen Schichten $[Q]_i$ mit $i = 1 \ldots 9$. Der 'Max'-Wert ergibt sich aus einer Abschätzung in [14] für die maximale Anzahl von Zuständen im Automaten im worst-case-Fall:

$$|Q|_{max,k} = \sum_{j=0}^{k-l-1} |\Sigma|^j + \sum_{j=0}^{|\Sigma|^l} \binom{|\Sigma|^l}{j},$$

wobei l die größte ganzzahlige Lösung der folgenden Ungleichung darstellt:

$$|\Sigma|^{k-l} \geq 2^{|\Sigma|^l} - 2^{|\Sigma|^{l-1}}$$

(Im Fall k=9 also l=1).

Die jeweils zweite Zeile 'Σ_{Follow}' bildet die Summe aller für einen Zustand $q \in [Q]_i$ in Frage kommenden Zustände $q' \in \{[Q]_{i+1} \ldots [Q]_k\}$.

Für ein *Verfahren A* bietet sich die Codierung der Folgezustände in kanonischer Weise mit jeweils $\lceil \log_2 \Sigma_{Follow} \rceil$ Bit an. So lassen sich z.B. die Zustände der Menge $[Q]_8 \cup [Q]_9$ mit jeweils 4 Bit codieren. Die Trennindizes der einzelnen Schichten müssen bei einer solchen Vorgehensweise als Zusatzinformation berücksichtigt werden. So ergibt sich ein zusätzlicher Aufwand von k Grenzwerten mit Werten $\leq |Q|_{max,9} = 21.861$, also $k \cdot \lceil \log_2 21.861 \rceil$ Bit $= k \cdot 15$ Bit. Tabelle 4 zeigt die mit diesem Vorgehen erzielten Werte.

Die Ergebnisse erreichen zwar nur knapp die Entropiewerte der Huffman-Codierung laut Tabelle 2, aber dabei muß man berücksichtigen, daß hier keine zusätzliche Information abgelegt werden muß (Im worst-case-Fall wird schließlich die Übertragung einer Huffman-Tabelle für 21.861 Codezeichen notwendig).

i	0	1	2	3	4	5	6	7	8	9
Max	1 (q_0)	4	16	64	256	1.024	4.096	16.384	14	2 ($F, \overline{F}$)
Cranach	1	4	15	56	208	642	1.795	1.376	14	2
Σ_{Follow}	4.113	4.112	4.108	4.093	4.037	3.829	3.187	1.392	16	2
$\lceil \log_2 \Sigma_{Follow} \rceil$	13	13	13	12	12	12	12	11	4	1
Duden	1	4	16	64	240	894	2.442	1.431	14	2
Σ_{Follow}	5.108	5.107	5.103	5.087	5.023	4.783	3.889	1.447	16	2
$\lceil \log_2 \Sigma_{Follow} \rceil$	13	13	13	13	13	13	12	11	4	1
Röntgen	1	4	15	41	95	215	497	772	14	2
Σ_{Follow}	1.656	1.655	1.651	1.636	1.595	1.500	1.285	788	16	2
$\lceil \log_2 \Sigma_{Follow} \rceil$	11	11	11	11	11	11	11	10	4	1
Schlüssel	1	3	8	20	54	163	531	1.000	14	2
Σ_{Follow}	1.796	1.795	1.792	1.784	1.764	1.710	1.547	1.016	16	2
$\lceil \log_2 \Sigma_{Follow} \rceil$	11	11	11	11	11	11	11	10	4	1

Tabelle 3: Zugehörigkeit der Zustände zu den Schichten $[Q]_i$

Bild	Codieraufwand $[Bit]$	c.f.
Cranach	145.520	0.56
Duden	190.212	0.73
Röntgen	48.612	0.19
Schlüssel	48.252	0.18

Tabelle 4: Ergebnisse nach Verfahren A

Das *Verfahren B* erreicht wesentlich bessere Ergebnisse und beruht auf den folgenden Beobachtungen:

- Der akzeptierende und der nichtakzeptierende Endzustand treten häufig als Folgezustand auf.
- In den ersten Schichten des Automaten ist dieser im allgemeinen relativ vollständig, d.h. Zustände konnten nicht zusammengefaßt werden und so bilden sich in der Tabelle lange Folgen von Folgezuständen, deren Numerierung sich aufgrund der Bildungsreihenfolge im Algorithmus nur durch eine Eins voneinander unterscheidet.

Tabelle 5 zeigt einige ausgewählte Daten der Automatentabellen für die Beispielbilder. Der akzeptierende Endzustand F und der nichtakzeptierende Endzustand $\overline{F}$ machen zusammen mindestens 25% der Tabelleneinträge aus. Etwa in derselben Größenordnung liegt auch die Anzahl der Zustände, die nach Eliminierung aller anderen Tabellenelemente eine monoton steigende Folge von Zahlen bilden, die jeweils eine Differenz von '1' zueinander aufweisen. Eine Verbesserung der Ergebnisse kann also erreicht werden, wenn man einen zweistufigen Codierungsmechanismus verwendet, bei dem zuerst entschieden wird, zu welcher der vier in der Tabelle aufgeführten Kategorien der jeweilige Eintrag gehört und dann im zweiten Schritt eine geeignete Codierung für die 'Rest'-Elemente findet.

Tabelle 6 zeigt in Spalte 1 die Ergebnisse für Schritt 1 bei Verwendung einer festen Codelänge von 2 Bit pro Tabelleneintrag und folgender Zuordnung:

$'00' \rightarrow F,\ '01' \rightarrow \overline{F},\ '10' \rightarrow$ Diff '1', $'11' \rightarrow$ 'Rest'.

Spalte 2 zeigt die Werte bei Verwendung eines Präfixcodes mit Huffman-Zuordnung entsprechend der Häufigkeit der einzelnen Kategorien.

Die Rekonstruktion der Automaten-Tabelle ist somit für die Werte der Kategorien F, $\overline{F}$ und Diff '1' eindeutig möglich, wenn der Startwert der monoton steigenden Folge mit abgelegt wird. Für die Werte der 'Rest'-Menge können wir eine kanonische Codierung vornehmen, die mit $\lceil \log_2(gr\ddot{o}\ss terWert - kleinsterWert) \rceil$ Bit pro Eintrag auskommt. Tabelle 7 zeigt diese Werte für die Beispielbilder und die Summe der Werte aus Schritt 1 und 2 mit den erreichten Kompressionsfaktoren.

Im Vergleich mit den Ergebnissen der klassischen Huffman-Codierung mit Blocklänge k=8 zeigt sich die Überlegenheit des Verfahrens im Bild *Schlüssel*. Bei genauer Betrachtung wird deutlich, daß in diesem Bild der linke obere Quadrant vollständig weiß ist und somit mit einem einzigen Zustand im Automaten repräsentiert werden kann. Bei den anderen Bildern zeigen sich geringfügig schlechtere Werte. Dabei ist aber wiederum zu beachten, daß bei der Huffman-Codierung zusätzlich der Codebaum zu berücksichti-

Bild	Größe der Automatentafel $(\|Q\|-2)\cdot\|\Sigma\|$	Anzahl			
		F	$\overline{F}$	Diff '1'	Rest
Cranach	16.444	1.418	3.473	5.202	6.351
Duden	20.424	1.259	4.404	5.267	9.494
Röntgen	6.616	2.066	440	2.334	1.776
Schlüssel	7.176	1.490	778	2.825	2.083

Tabelle 5: Häufigkeiten der Kategorien in den Automatentabellen

Bild	feste Codelänge				Aufwand	Präfixcode				Aufwand
	F	$\overline{F}$	Diff '1'	Rest	[Bit]	F	$\overline{F}$	Diff '1'	Rest	[Bit]
Cranach	00	01	10	11	32.888	111	110	10	0	31.428
Duden	00	01	10	11	40.848	111	110	10	0	37.017
Röntgen	00	01	10	11	13.232	10	110	0	111	13.114
Schlüssel	00	01	10	11	14.352	110	111	0	10	13.795

Tabelle 6: Schritt 1 von Verfahren B

Bild	Min	Max	$\lceil\log_2(Max-Min)\rceil$	Schritt 2 [Bit]	Σ Verfahren B [Bit]	c.f.
Cranach	437	4.110	12	76.212	107.640	0.41
Duden	1.226	5.105	12	113.928	150.945	0.58
Röntgen	159	1.653	11	19.536	32.650	0.12
Schlüssel	109	1.793	11	22.913	36.708	0.14

Tabelle 7: Schritt 2 und Gesamtergebnis von Verfahren B

gen ist, während beim hier vorgestellten Verfahren lediglich Angaben über Anzahl der Zustände, Wert von $|\Sigma|$ usw. abgelegt werden müssen.

Um den Vergleich mit bestehenden Kompressionsverfahren zu verdeutlichen, zeigt Tabelle 8 abschliessend die Werte für die Programme:

- COMPRESS (Version 8.2) Adaptive Lempel-Ziv-Codierung [5].
- PACK Huffman-Codierung.
- COMPACT Adaptive Huffman-Codierung.

Alle Tools wurden unter DEC ULTRIX 4.4 verwendet. Die Bitmap wurde dabei als 32 KByte-Datei gelesen, in der jeweils 8 Pixel zu einem Byte zusammengefaßt wurden. Das Programm COMPRESS verwendet einen Tabellen-basierten Ansatz und wurde deshalb mit in den Vergleich integriert, um neuere Verfahren für sequentielle Daten zu berücksichtigen, die höhere Kompressionsfaktoren erreichen können, als die traditionellen Huffman-Codierer.

Auch hier wird deutlich, daß sich die verlustlose Kompression mit Automaten durchaus mit herkömmlichen Methoden vergleichen läßt. Bei den Bildern *Cranach*, *Röntgen* und *Schlüssel* werden ähnliche Werte erreicht bzw. übertroffen. Nur im Bild *Duden* wird ein wesentlich schlechteres Ergebnis erzielt.

5 Verwendetes Bildmaterial

Die vorgestellten Ergebnisse wurden aus den folgenden vier Bildbeispielen berechnet, die so ausgewählt wurden, daß eine möglichst große Bandbreite von Nutzanwendungen abgedeckt wird. Alle Abbildungen sind mit Hilfe eines Scanners von natürlichen Vorgaben abgetastet und anschließend binarisiert und auf eine einheitliche Größe von 512 mal 512 Pixel gebracht worden.

- *Cranach* ist eine Karikatur auf das Papsttum von Lukas Cranach dem Älteren um 1545. Es handelt sich dabei um eine Nachzeichnung nach einem Holzschnitt [4].
- *Duden* ist ein Ausschnitt aus dem etymologischen Wörterbuch der deutschen Sprache [8] und steht als Beispiel für ein Faksimile.
- *Röntgen* ist aus einem gescannten Röntgenbild des Autors entstanden und steht für medizinische Bilddatenverarbeitung.
- *Schlüssel* ist der gescannte Zimmerschlüssel des Autors, wie er als Vorlage in einer Bilderkennung zu finden wäre.

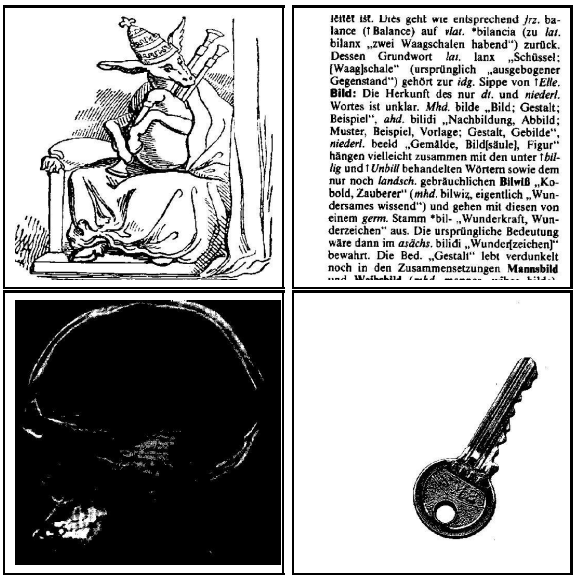

Abbildung 3: Beispielbilder *Cranach, Duden, Röntgen, Schlüssel*

6 Zusammenfassung und Ausblick

Wir haben in diesem Artikel die Codierung von digitalen Bildsignalen mit Hilfe von endlichen deterministischen Automaten beschrieben. Dazu werden den relevanten Pixeln der Bildstruktur Wörter einer endlichen Sprache zugeordnet, welche dann durch einen DFA repräsentiert wird. Es wurden weiterhin zwei heuristische Ansätze zu einer effektiven Codierung der Automatenübergangstabelle des reduzierten Automaten angegeben. Die erreichten Kompressionsfaktoren wurden mit den Ergebnissen klassischer Entropiecodierungsverfahren verglichen, die aufgrund von Kenntnissen über den stochastischen Bilderzeugungsprozeß einen 'optimalen' Präfixcode erzeugen. Dabei wurde der Nachteil dieser Verfahren deutlich, die aufgrund der Nichtstationarität des zweidimensionalen Bildsignals nur suboptimale Lösungen erzeugen können. Auch die Verwendung eines adaptiven Huffman-Verfahren (COMPACT, siehe Tabelle 8)

Bild	COMPRESS [Bit]	c.f.	PACK [Bit]	c.f.	COMPACT [Bit]	c.f.
Cranach	107.960	0.41	102.944	0.39	102.672	0.39
Duden	125.768	0.48	123,264	0.47	123.192	0.47
Röntgen	31.048	0.12	49.712	0.19	48.848	0.19
Schlüssel	36.616	0.14	57.856	0.22	56.944	0.22

Tabelle 8: Vergleich mit bestehenden Kompressionsverfahren.

konnte diesen Nachteil nur bedingt korrigieren. Der hier verwendete Ansatz findet sich in ähnlicher Form auch in der WFA-Methode von Culik [12], wobei der Hauptunterschied in der Intention und in der Interpretation der Automatenzustände besteht. Bei Culik steht jeder einzelne Zustand für ein vollständiges Bild und das Originalbild wird durch eine gewichtete Linearkombination aus diesen Zustandsbildern erzeugt. Das Verfahren ist auf die verlustbehaftete Codierung von Grauwertbildern ausgerichtet und verwendet dazu Abstandskriterien, die an die subjektiven Vergleichskriterien des Menschen angepaßt sind.

Der hier vorgestellte Ansatz dient der Ausnutzung zweidimensionaler Redundanz zur verlustlosen Kompression. Hier entspricht jeder Automatenzustand einem bestimmten Teilmuster des Bildes. Dabei entfällt eine explizite Bestimmung der Faktoren der Linearkombination und das Originalbild ergibt sich durch Vereinigung der Teilmuster.

Die Quadtree-Zerlegung ist der erste kanonische Ansatz zur Erzeugung der endlichen Bildsprache. Hier lassen sich allerdings noch eine Reihe bildabhängiger Alternativen denken. Z.B. die Variation von $|\Sigma|$ auf jeder Schicht $[Q]_i$ oder eine feinere und abweichende Struktur bei gleichbleibendem $|\Sigma|$. Weiterführende Arbeiten könnten die Zerlegung von transformationscodierten Bilddaten untersuchen, wobei durch die Dekorrelation der Daten eine Steigerung des Kompressionsfaktors zu erwarten ist. Eine weitere Problematik ergibt sich bei der Analyse der Mächtigkeiten alternativer Automatentypen und der Erweiterung der Bilddaten von der zweiwertigen Bitmap zu Grauwert-, Farb- und Bewegtbildern.

Literatur

[1] D.A. Huffman, *A method for the construction of minimum redundancy codes*, Proc. IRE 40 (1952), 10, pp. 1098-1101.

[2] Claude E. Shannon, Warren Weaver, *Mathematische Grundlagen der Informationstheorie*, Oldenbourg Verlag, München, 1976.

[3] J. Ziv, A. Lempel, *A Universal Algorithm for Sequential Data Compression*, IEEE Transactions on Information Theory, Vol. IT-23, No. 3, pp. 337-343, May 1977.

[4] Georg Reinhardt [Hrsg.], *A. Paul Weber, Das graphische Werk: Handzeichnungen und Lithographien 1930 - 1978*, Schirmer-Mosel, München, 1980.

[5] Terry A. Welch, *A Technique for High-Performance Data Compression*, IEEE Computer, Vol. 15, No. 6, pp. 8-19, June 1984.

[6] I.H. Witten, M.N. Radford, J.G. Cleary, *Arithmetic Coding for Data Compression*, Communications of the ACM, Vol. 30, No. 6, pp. 520-540, 1987.

[7] Michael F. Barnsley, Alan D. Sloan, *A Better Way to Compress Images*, Byte, No. 1, pp. 215-223, 1988.

[8] G. Drosdowski, *Duden Etymologie: Herkunftswörterbuch der deutschen Sprache*, Dudenverlag, Mannheim, Wien, Zürich, 1989.

[9] W. Heise, P. Quattrocchi, *Informations- und Codierungstheorie*, Springer Verlag, Berlin, Heidelberg, 1989.

[10] Gregory K. Wallace, *The JPEG Still Picture Compression Standard*, Communication of the ACM, Vol. 34, No. 4, pp. 40-44, 1991.

[11] H. Völz, *Grundlagen der Information*, Akademie Verlag, Berlin, 1991.

[12] Karel Culik II, Jarkko Kari, *Image-data compression using edge-optimizing algorithm for WFA inference*, Information Processing & Management, Vol. 30, No. 6, pp. 829-834, 1994.

[13] Frank Katritzke, *Über die redundanzvermindernde Codierung digitaler Bilddaten*, Diplomarbeit, Universität-Gesamthochschule Siegen, 1994.

[14] Michael Zettler, *Rekursive, hierarchische Zerlegungskodierung digitaler Bilddaten mit endlichen Automaten*, Diplomarbeit, Universität-Gesamthochschule Siegen, 1994.