Verlustfreie Kompression in rekonfigurierbarer Hardware

DIPLOMARBEIT

zur Erlangung des akademischen Grades
Diplominformatiker

eingereicht an der
Mathematisch-Naturwissenschaftlichen Fakultät II
Institut für Informatik
Humboldt-Universität zu Berlin

von

Herr Maximilian Buder

geboren am 08.01.1981 in Berlin

Betreuer:

1. Prof. Dr. Beate Meffert
2. Dr. rer. nat. David Krutz
3. Dr. Ing. Frank Winkler

eingereicht am: 23. Oktober 2007

At times when high input data rates cannot be handled due to insufficient capacities of communication links or receivers, data reduction might be a way to ease the constraints on the transmission and the processing units. For satellite and airborne imagery, lossless image compression at realtime is desired. The work done in this thesis aims to evaluate lossless image compression techniques with respect to achievable compression rates and a possible implementation in reconfigurable hardware. A realtime context adaptive hardware implementation is presented, that was written in vhdl. The design was verified on a standard fpga to operate at one-pixel per clock and a clock rate up to 125 Mhz. The chosen implementation is very resource efficient and achieves good compression rates.

Keywords:
lossless compression, fpga, image processing, remote sensing

Immer dann, wenn die Kapazitäten von Kommunikationswegen oder Empfängern durch zu hohe Datenraten voll ausgeschöpft oder nicht ausreichend vorhanden sind, können durch eine Reduktion der Daten die Anforderungen an die Übertragungsstrecke bzw. den Empfänger reduziert werden. Speziell für Bildsensoren aus dem Bereich der Fernerkundung ist eine verlustfreie Verminderung der Datenmenge in Echtzeit wünschenswert. In dieser Arbeit werden für Sensordaten von Matrix- und Zeilenkameras verlustfreie Kompressionsverfahren bezüglich ihrer Kompressionsrate und einer möglichen Umsetzung in rekonfigurierbarer Hardware evaluiert. Eine Hardwareimplementierung für ein echtzeitfähiges kontextadaptives Verfahren wird vorgestellt. Die Sensordaten werden mit einer Eingangsdatenrate von einem Pixel pro Takt verarbeitet. Der Entwurf wurde auf einem FPGA mit einem Takt von 125 Mhz erfolgreich getestet und erzielt gute Kompressionsraten bei sehr geringem Ressourcenverbrauch.

Schlagwörter:
verlustfreie Kompression, FPGA, Bilderverarbeitung, Fernerkundung

Inhaltsverzeichnis

1. Einleitung

1.1. Motivation

Die Benutzung von Kamerasystemen, wie z.B. zur Verkehrsszenenanalyse, stellt große Anforderungen an die Datenübermittlung und die Datenvorverarbeitung. Dabei kann es sein, dass die Übertragungs- und Verarbeitungseinheiten den Anforderungen nicht gewachsen sind. Kann weder die Datenrate der Datenquelle reduziert, noch die Kapazität des Übertragungskanals vergrößert werden, besteht durch Kompression der Daten eine Möglichkeit, das Problem bezüglich der Datenübertragung zu mindern. Für den Fall, dass es nicht möglich ist, die Daten auf der Empfängerseite in Echtzeit zu prozessieren, müssen diese zwischengespeichert werden. Durch die Benutzung von echtzeitfähigen Kompressionsalgorithmen können die Anforderungen an die Zwischenspeicherung reduziert werden. Weiterhin können Reserven erschlossen werden, um so Einsparpotentiale und Erweiterungsmöglichkeiten zu gewinnen. Dabei ist der Aufwand, der betrieben werden muss, um eine immer bessere Kompression zu erreichen, nicht unerheblich und hat Einfluss auf die Verarbeitungsgeschwindigkeit des Gesamtsystems.

1.2. Aufgabenstellung

Im Rahmen dieser Arbeit sollen verlustfreie und echtzeitfähige Kompressionsalgorithmen speziell für Bildsensoren (Matrix- und Zeilensensoren) hinsichtlich ihrer Kompressionsrate und ihrer Umsetzung in konfigurierbarer Hardware untersucht werden. Dabei soll anhand der Kriterien Ressourcenverbrauch, Datendurchsatz und Latenzzeit eine Hardwarerealisierung bewertet werden. Das Kompressionsmodul soll mit Bezug zum VHDL-Betriebssystemkonzept, das von Krutz entworfen wurde [16], entwickelt werden, um einen möglichst hohen Grad an Wiederverwendung zu gewährleisten.

Die Abteilung Optische Informationssysteme am Deutschen Zentrum für Luft- und Raumfahrt e.V. (DLR) [3] ist an internationalen Projekten beteiligt, für die sie optoelektronische Komponenten entwickelt, die für die Fernerkundung sowohl aus der Luft als auch aus dem Weltraum eingesetzt werden. Bevor jedoch die neuen optischen Sensormodule für Fernerkundungssatelliten zur Auslieferung kommen, werden diese vom DLR verifiziert. Zu der Testumgebung (engl.: electronic ground support equipment (EGSE)) gehören sowohl Hardware- als auch Softwarekomponenten, mit denen der Austausch der Nutz- und Steuerdaten zwischen dem Testobjekt und dem Messplatz realisiert wird. Dabei modelliert die EGSE alle Funktionen des Satelliten die nötig sind, um das neue Modul im Weltall zu betreiben. Das heißt, dass die EGSE den Sensordatenfluss in Echtzeit empfängt und zur Auswertung auf einem Speicherarray ablegt. Bevor die Sensordaten gespeichert werden, müssen sie wegen der beschränkten Schreibgeschwindigkeit des Festplattenarrays in Echtzeit verlustfrei komprimiert werden. In diesem Zusammenhang ist Echtzeit definiert als ein Pixel pro Takt. Typischerweise sind heutige Fernerkundungssatelliten mit Zeilenkameras mit bis zu 24.000 Pixeln ausgestattet, die je nach Flughöhe des Satelliten mit circa 10 kHz ausgelesen werden. Bei Auflösungen von 16 bit je Bildpunkt kommen damit pro Zeile Datenraten von bis zu 480 MByte/s zustande.

1.3. Multifunctional Camera

Eine Eigenentwicklung des DLR ist die Multifunctional Camera (MFC). Die MFC ist ein Kamerasystem mit modularem Aufbau. In festen Stufen kann die optische Fertigkeit der Gesamtkamera durch den Einbau bzw. den Ausbau optoelektronischer Module - sogenannter Fokalebenenmodule - verändert werden. Die digitalen Daten der Module werden über eine USB-2.0-Schnittstelle auf einen handelsüblichen Rechner (z.B. Laptop, PC104-Stack oder 19-Zoll-Industrierechner) übertragen und dort für jedes Fokalebenenmodul separat auf einer Festplatte gespeichert. Eine Einschränkung stellt die Übertragung der Sensordaten über den USB-2.0-Kanal dar, der im Augenblick eine maximale Datenrate von effektiv 25 Mbyte/s zulässt. Eine Kompressionsstufe vor der USB-2.0-Schnittstelle könnte die Leistungsfähigkeit der MFC steigern. In Bezug auf das Kompressionsmodul ist zu beachten, dass kein externer Speicher

zur Speicherung von Zwischenergebnissen zur Verfügung steht. Die Rohdaten der MFC-Module entstammen einer Zeilen-CCD mit wahlweise 14000 oder 8000 Pixelbreite. Die Auslesegeschwindigkeit ist von mehreren Faktoren wie zum Beispiel der Flughöhe und der Fluggeschwindigkeit abhängig.

2. Informationstheorie

Die Informationstheorie bildet das mathematische Fundament für verlustbehaftete und verlustfreie Datenkompression. In diesem Abschnitt werden Begriffe und Theoreme aus dem Gebiet der Informationstheorie eingeführt, die eine Analyse der Kompressionsverfahren ermöglichen.

Notation

L_u	=	Länge (Anzahl) der unkomprimierten Daten in bits
L_c	=	Länge (Anzahl) der komprimierten Daten in bits
$\mathcal{A}$	=	Alphabet mit N unterschiedlichen Symbolen
s_i	=	i-tes Element eines Alphabets
c_i	=	Codewort, das dem Symbol s_i zugeordnet ist
$\mathcal{C}$	=	Menge der Codewörter (Codebuch)
$l(c_i)$	=	Länge eines Codeworts c_i in bits
N	=	Anzahl der unterschiedlichen Symbole eines Alphabets
$x[n]$	=	n-te Realisierung einer Zufallsgröße X
Pixel	=	Bildpunkt, Bildzelle oder Bildelement

2.1. Entropie

Sei X eine Zufallsveränderliche und dadurch beschrieben, dass sie unter Zufallsbedingungen Elementarereignissen E Werte x aus einem reellen Bereich zuordnet.

$$X \;:\; E \longrightarrow X(E) \in \mathbb{R} \tag{2.1}$$

Die Menge E wird als Ereignisraum oder Alphabet $\mathcal{A}$ bezeichnet und umfasst die Menge aller möglichen Ereignisse bzw. aller unterschiedlichen Symbole mit

$$\mathcal{A} = \{s_i\} \text{ und } i \in \mathbb{N}.$$

Ist der Wertebereich von X endlich oder abzählbar unendlich, spricht man von einer diskreten Zufallsgröße. Ein wert- und zeitdiskretes Zufallssignal $X[n]$ kann als eine Folge von Realisierungen $x[n]$ einer diskreten Zufallsgröße zu den Abtastpunkten $n = (0, 1, \ldots)$ interpretiert werden.

Jedes Symbol s_i besitzt eine Auftretenswahrscheinlichkeit p_i, die direkt im Zusammenhang mit dem Begriff des Informationsgehaltes (self-information) eines Symbols steht. Der 1948 von Claude E. Shannon in [33] geprägte Begriff des Informationsgehaltes I eines Zeichens s_i wird mit

$$I(s_i) \quad = \quad -log(p_i) \tag{2.2}$$

berechnet. Die Gleichung (2.2) sagt aus, dass Information als Beseitigung von Unsicherheit betrachtet werden muss. Ist zum Beispiel die Auftretenswahrscheinlichkeit eines Symbols s_i gleich eins ($p_i = 1$), so ist der Grad an gewonnener Information gleich null. Sollte $p_i = 0$ und der rechte Term der Gleichung 2.2 nicht definiert sein, wird per Definition $I(s_i) = 0$ gesetzt. Ist die Basis des Logarithmus aus Gleichung (2.2) zwei, dann ist die Einheit für den Informationsgehalt das Bit.

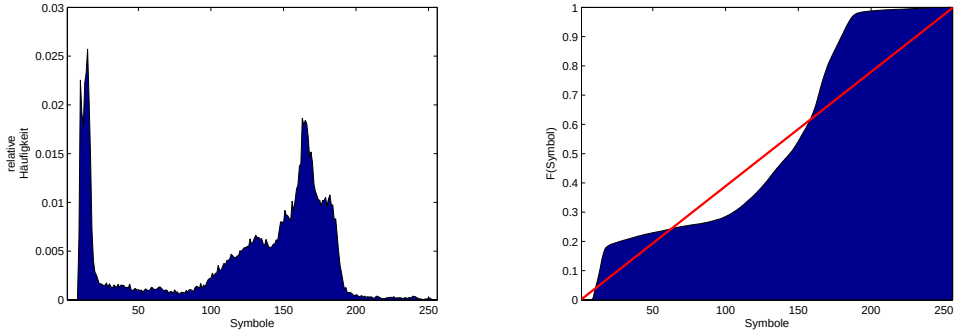
Für die Berechnung des mittleren Informationsgehaltes H_i eines Zeichens muss der Term aus der Gleichung (2.2) mit der Auftretenswahrscheinlichkeit des Zeichens gewichtet werden. Dabei entsteht oftmals das Problem, dass die Auftretenswahrscheinlichkeiten nicht direkt vorliegen. Stattdessen werden die relativen Häufigkeiten einer endlichen Realisierung $j = \{1, 2, \ldots, N\}$

$$p_i = \frac{Anzahl(s_i)}{\sum_j s_j}$$

als Näherung verwendet, die zum Beispiel aus einem normierten Histogramm ablesbar sind (Abbildung 2.1(a)).

$$H_i(s_i) \quad = \quad -p_i \cdot log(p_i) \tag{2.3}$$

Als Entropie H einer Signalquelle wird die Aufsummierung des gewichteten Informationsgehaltes aller Elemente eines Alphabets $\mathcal{A}$ bezeichnet. Die Entropie gibt somit den mittleren Informationsgehalt einer Signalquelle bzw. eines Al-



(a) Wahrscheinlichkeitsdichtefunktion für das Grauwertbild A.5(b)

(b) Verteilungsfunktion für das Grauwertbild A.5(b)

Abbildung 2.1.: statistische Untersuchungen zum Kameramannbild A.5(b)

phabets an. N ist die Anzahl unterschiedlicher Symbole aus dem Alphabet $\mathcal{A}$.

$$H = \sum_{i=0}^{N-1} H_i \tag{2.4}$$

$$= -\sum_{i=0}^{N-1} p_i \cdot log(p_i) \tag{2.5}$$

Der Wertebereich der Entropie H ist durch die Anzahl N verschiedener Symbole aus $\mathcal{A}$ bestimmt.

$$0 \leqslant H \leqslant log(N) \tag{2.6}$$

Der Wert der Entropie ist am höchsten, wenn alle Symbole des Alphabets gleichverteilt sind, d.h. $p_i = \frac{1}{N}$. Betrachtet man die Verteilungsfunktion für die Symbole eines Alphabets, dann muß ein geradliniger und diagonaler Verlauf zu erkennen sein (Abbildung 2.1(b)).

$$H_{max} = \sum_{i=0}^{N-1} \frac{1}{N} \cdot log(N) \tag{2.7}$$

$$= log(N) \tag{2.8}$$

Der maximale Informationsgehalt H_{max} wird auch Entscheidungsgehalt genannt und signalisiert, dass ein Symbol maximal viel Information enthält [19].

Eine Signalquelle mit dem Alphabet $\mathcal{A}_{bin} = \{0, 1\}$ wird als binäre Quelle bezeichnet. Die Abbildung 2.2 zeigt das Verhältnis von Auftretenswahrscheinlichkeit und Entropie einer binären Quelle mit $p(1) = 1 - p(0)$. Die Entropie erreicht ihr Maximum bei $p = 0.5$ und ihr Minimum bei $p(0) = 1$ bzw. $p(1) = 1$. Dies deckt sich mit der Aussage, dass die Entropie der Quelle den Wert der maximalen Entropie H_{max} annimmt, wenn die Symbole gleichverteilt sind.

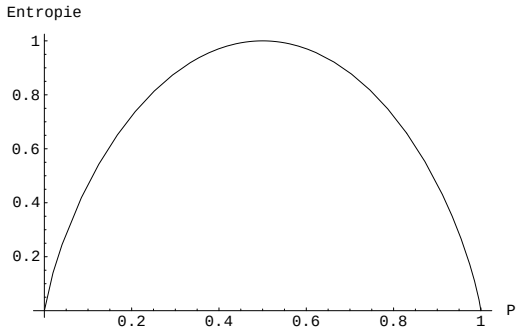


Abbildung 2.2.: Entropie eines binären Signals in Abhängigkeit der Symbolwahrscheinlichkeit p

Von Symbolredundanz R_s spricht man, wenn die tatsächlich auftretenden Symbolhäufigkeiten nicht gleichverteilt sind, d.h. die Menge an Information je Zeichen nicht optimal ist.

$$R_s = H_{max} - H_{src} \tag{2.9}$$

Seine praktische Bedeutung erhält Gleichung (2.9) dadurch, dass für $R_s \neq 0$ gute Voraussetzungen zur verlustfreien Kompression einer Symbolfolge gegeben sind. Angenommen N_D ist die eingesetzte Anzahl an Bits zur Speicherung von N_S Symbolen, dann kann der durchschnittliche Aufwand zur Speicherung eines Symbols S_{cod} mit fester Bitlänge als Quotient aus

$$S_{cod} = \frac{N_D}{N_S} \geqslant \lceil log_2(N) \rceil \tag{2.10}$$

bestimmt werden. Die Differenz aus dem Aufwand der Codierung S_{cod} und der Entropie der Quelle H_{src} wird oft auch als Codierungsredundanz bezeichnet.

$$R_{cod} = S_{cod} - H_{src} \tag{2.11}$$

Sie kann als Mehraufwand interpretiert werden, der für die Codierung mit Datenwörtern (fester und variabler Länge) betrieben werden muss. Üblich ist auch die normierte Version der Codierungsredundanz, die als sog. relative Codierungsredundanz bezeichnet wird und eine einfachere Vergleichsmöglichkeit unterschiedlicher Codierungsverfahren gestattet.

$$R_{rel.cod} = \frac{(S_{cod} - H_{src})}{H_{src}} \tag{2.12}$$

Als eine bisherige Voraussetzung zur Berechnung der Redundanz galt die Annahme, dass die Symbole unabhängig voneinander sind. Dies ist in der Praxis eher selten der Fall, da die Symbole häufig zu Symbolen in der unmittelbaren Nachbarschaft korreliert sind. Effektive Kompressionsalgorithmen müssen diesen Zusammenhang erkennen. Seien X, Y zwei diskrete Zufallsgrößen, dann wird die bedingte Entropie H_{cond} in der Form

$$H_{cond}(X|Y) = -\sum_{(X,Y)\in X\times Y} p(x|y) \cdot p(y) \cdot log\left(p(x|y)\right) \tag{2.13}$$

notiert. Unter Intersymbolredundanz wird die Differenz

$$R_{inter} = H_{src} - H_{cond} \tag{2.14}$$

aus der Entropie der Signalquelle H_{src} und der bedingten Entropie H_{cond} mit

$$H_{cond} \leqslant H_{src}$$

verstanden. Durch die Hinzunahme der bedingten Entropie H_{cond} ist lediglich eine bessere Abschätzung der Quellenentropie H_{src} möglich.

2.1.1. Entropie einer geometrischen Verteilung

Häufig werden Modelle einer Signalquelle benutzt, die auf einer geometrischen Verteilung F_g beruhen. Deren Wahrscheinlichkeit

$$P_g(X = i) = p_i \tag{2.15}$$

$$= \rho^i(1-\rho), \quad i \in \mathbb{N}; \quad \rho \in (0,1) \tag{2.16}$$

ist von einem Parameter ρ abhängig. Für geometrisch verteilte Zufallsgrößen X kann ebenfalls die Entropie in Abhängigkeit zum Parameter ρ und der Erwartungswert von X berechnet werden.

$$H_g(p_i) = -\sum p_i \cdot log_2\,(p_i) \tag{2.17}$$

$$= -log_2(1-\rho) - \frac{\rho}{1-\rho} \cdot log_2(\rho) \tag{2.18}$$

Abbildung 2.3 zeigt den Zusammenhang zwischen dem Parameter ρ und der Entropie H_g. Da die meisten Entropiecodierer einzelne Symbole codieren, kann die Codierungsentropie nicht geringer als 1 bit pro Symbol sein |[1]. Daraus ergibt sich eine untere Schranke für den Parameter ρ der geometrischen Verteilung von $\rho \geqslant 0.227092$ (bzw. $\rho \geqslant \frac{1}{4}$). Je näher der Parameter ρ dem Wert eins kommt, umso mehr steigt der Informationsgehalt.

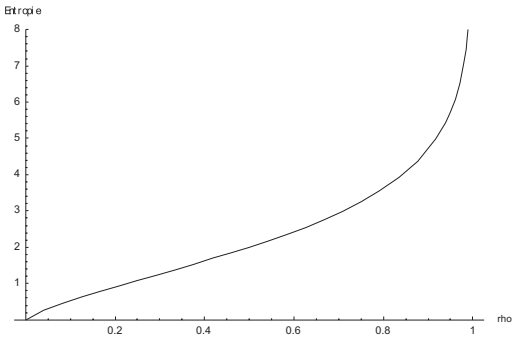


Abbildung 2.3.: Entropie einer geometrischen Verteilung in Abhängigkeit zu einem gewählten ρ

Der Erwartungswert einer Zufallsgröße mit einer geometrischen Verteilung errechnet sich aus

$$E[X] = \frac{\rho}{1-\rho}. \tag{2.19}$$

Maximum-Likelihood-Approximation

Der Parameter ρ einer geometrischen Verteilung ist meistens nicht bekannt und kann anhand einer endlichen Stichprobe geschätzt werden. In einem ersten Durchgang werden die einzelnen N Vorkommnisse der Symbole in einer

[1] siehe Abschnitt 3.1.2

Summe A aufsummiert. Am Ende des ersten Durchlaufs kann die geometrische Verteilung mit Parameter ρ durch die Abschätzung $\widehat{\rho} = \frac{A}{N}$ approximiert und in einem zweiten Gang für die eigentliche Aufgabe genutzt werden. Diese Vorgehensweise ist jedoch zu zeitintensiv und für große Alphabete zu kostenintensiv, da für jedes Symbol ein Zähler (die Zählerbreite ist abhängig von der Größe der Stichprobe) bereitgehalten werden muss. In einem sequentiellen Verfahren wird versucht, die Eingangssymbole nur einmal zu lesen und den Parameter ρ anhand einer kleinen Zahl N_v von Vorgängern zu ermitteln. Eine einfache Methode zur sequentiellen Berechnung von ρ stellt die Maximum-Likelihood-Schätzung dar. Für eine sog. Likelihood-Function $L[\rho]$ wird hierbei das Maximum ermittelt. Zur Vereinfachung wird die logarithmische Form der Likelihood-Funktion für die geometrische Verteilung als Ausgangspunkt verwendet.

$$L[\rho] = \sum_{i=0}^{N_v-1} ln(p_i) \tag{2.20}$$

$$= ln(1-\rho) \cdot N_v + ln(\rho) \cdot \sum_{i=0}^{N_v} s_i \tag{2.21}$$

Die erste Ableitung von $L[\rho]$ wird null gesetzt und nach ρ aufgelöst. Unter der Bedingung $dL[\rho]/d\rho = 0$ ergibt sich:

$$\widehat{\rho} = \frac{\sum_{i=0}^{N_v-1} s_i}{\sum_{i=0}^{N_v-1} s_i + N_v} \tag{2.22}$$

wobei $\widehat{\rho}$ der Wert ist, bei dem L extremal ist. Unter Berücksichtigung der zweiten Ableitung von L folgt, dass $\widehat{p}$ L maximiert. Durch eine einfache Mittelwertbildung

$$\widehat{\mu} = \frac{\sum_{i=0}^{N_v-1} s_i}{N_v} \tag{2.23}$$

weniger unabhängiger Symbole, die in der Abarbeitungssequenz vorweg aufgenommen worden sind, kann der Parameter ρ einer geometrischen Verteilung durch die Approximation $\widehat{\rho}$ mit

$$\widehat{\rho} = \frac{\widehat{\mu}}{1+\widehat{\mu}} \tag{2.24}$$

auf einfache Art angenähert werden.

3. Datenkompression

Unter Datenkompression versteht man im Allgemeinen eine Reduzierung der Datenmenge. Sie ist in vielen Fällen wünschenswert, da sie Ressourcen und damit Kosten sparen kann. Codierung, Dekorrelation und Datenreduktion sind die drei Bereiche, in die man die Methoden zur Verringerung der Datenmenge unterteilen kann.

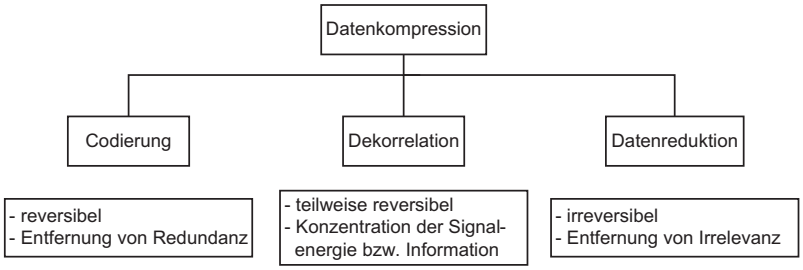


Abbildung 3.1.: Schematische Gliederung der Methoden zur Datenkompression (Quelle: [40])

Die Aufgabe von Codierern besteht darin, die Codierungsredundanz durch Beobachtung der Symbolverteilungen zu minimieren. Durch Umcodierung wird versucht, die mittlere Bitrate an an den Wert der Entropie H_{src} der Signalquelle anzunähern. Symbolen mit hoher Auftretenswahrscheinlichkeit wird eine entsprechend hohe Bedeutung beigemessen. Zum Beispiel weisen Codes mit variabler Länge (engl.: variable length code (vlc)) häufig vorkommenden Symbolen kurze Codewörter zu. Umgekehrt werden bei seltenem Vorkommen lange Codes vergeben. Intersymbolredundanz lässt sich damit jedoch nicht beseitigen. In vielen Fällen wird ein Modell der Daten aufgestellt, das zu jedem Zeitpunkt der Codierung Vorhersagen zu den Auftretenswahrscheinlichkeiten des aktuellen Symbols macht. Dabei geht das Modell auf die Korrelation aufeinander folgender Symbole ein. Übertragen wird lediglich eine Beschreibung des Modells und Information darüber, wie stark das Eingangssignal vom Modell abweicht. Eine ebenso populäre Herangehensweise sind Transformationen des

gesamten Eingangssignals in eine günstigere Darstellung. Der Schritt der Dekorrelierung findet in Phasen der Präcodierung statt. Dabei können mehrere solcher Präcodierungsstufen parallel, seriell oder umschaltbar angeordnet sein. Fein abgestimmte Modelle im Zusammenspiel mit optimalen Codierern erlauben am Ende ideale Kompressionsraten. Jedoch steigt die Komplexität der Verfahren überproportional zu der hinzugewonnenen Kompression an, sodass der Datendurchsatz erheblich verschlechtert wird. Methoden, bei denen Information durch Datenreduktion verloren geht, sollen in dieser Arbeit nicht betrachtet werden. Algorithmen zur Verminderung der Datenmenge bergen auch Nachteile. So bewirkt die Entfernung bzw. Minimierung von Redundanz auch eine gesteigerte Anfälligkeit der Daten gegenüber fehleranfälligen Übertragungswegen. Weiterhin besteht die Gefahr, dass anstelle einer Reduzierung eine Datenexpansion durchgeführt wird.

3.1. Codierung

Abbildung (2.1(b)) aus dem vorherigen Kapitel zeigt das kumulierte Histogramm für das Kameramannbild A.5(b) aus der Grauwert Bildersammlung Waterloo GraySet1 [15]. Es ist zu erkennen, dass die Entropie der Quelle H_{src} nicht der Maximalentropie H_{max} entspricht, da die Symbole in der Signalquelle nicht gleichverteilt vorkommen. Die Codierung der Symbole mit festen 8-bit-Codewörten pro Symbol bzw. 8 bit pro Pixel (bpp) ist für dieses Bild nicht optimal gewählt. Es sollte eine Möglichkeit geben, eine günstigere Darstellung zu finden, die Rücksicht auf die Symbolverteilung nimmt. Bevor diese günstigeren Codierungsverfahren vorgestellt werden, ist zu klären, was ein optimaler Code ist. Mit Hilfe der Ungleichungen von Kraft-McMillan Satz 3.1 konnte Shannon in seiner Arbeit [33] grundlegende Aussagen zur Qualität eindeutig decodierbarer Codes treffen.

Satz 3.1

Wenn $\mathcal{C}$ eine Menge an eineindeutig decodierbaren Codes $c_i \in \mathcal{C}$ ist und l_i die Codewortlänge von c_i, dann gilt:

$$\sum_{i=1}^{N} 2^{-l_i} \leqslant 1. \tag{3.1}$$

Mehrere Aspekte lassen sich aus Satz 3.1 ableiten. Zum einen kann immer ein eineindeutig decodierbarer Code gefunden werden, der eine Darstellung mit weniger als $H_{src} + 1$ bit, jedoch mehr als H_{src} bit pro Abtastwert ermöglicht.

$$H_{src} \leqslant \bar{l} \leqslant H_{src} + 1 \tag{3.2}$$

wobei

$$\bar{l} = \sum_{i \in N} p_i \cdot l_i \tag{3.3}$$

die mittlere Codelänge einer Signalquelle und l_i die Länge von c_i ist. Die Tatsache, das H_{src} der kleinsten möglichen Anzahl Codebits zur Darstellung der

Symbole entspricht, macht den Begriff der Entropie so bedeutend. Für einen ausführlichen Beweis von Satz 3.2 sei auf Hoggar [7] verwiesen.

Zum anderen besteht ein weiterer Aspekt von Satz 3.1 darin, dass bei Gleichheit von 3.1 gezeigt wird, dass alle möglichen Codewörter c_i tatsächlich auch verwendet werden. Die Kraft-McMillan Ungleichung 3.1 ist somit auch ein Maß für die Effizienz von $\mathcal{C}$.

Satz 3.2

In Umkehrung zu Satz 3.1 kann bewiesen werden, dass wenn $l_1, \ldots, l_N$ gegeben sind und die Ungleichung

$$\sum_{i=1}^{N} 2^{-l_i} \leqslant 1 \tag{3.4}$$

erfüllt ist, immer ein präfixfreier Code mit den Längen $l_1, \ldots, l_N$ gefunden werden kann.

Satz 3.2 erlaubt es, sich auf präfixfreie Codes zu beschränken, ohne andere eineindeutig decodierbare Codes zu vernachlässigen, für die kürzere $\bar{l}$ existieren.

3.1.1. Präfixfreie Codes

Ein häufig genanntes Beispiel zur Einführung ist der Morsecode. Mit Rücksicht auf die Auftretenswahrscheinlichkeiten der Buchstaben eines Alphabets werden den Zeichen verschieden lange Folgen von Strichen und Punkten zugewiesen. Zusätzlich zu diesen beiden Codezeichen existiert noch ein gesondertes Pausezeichen, das als drittes Codezeichen die eigentlichen Codewörter bei der Decodierung separiert. Dieses Sonderzeichen ist unerlässlich, denn nur dieses ermöglicht eine fehlerfreie Decodierung, da kurze Codewörter wiederum teil längerer sein können. Codes, die diese Eigenschaft nicht besitzen, werden präfixfreie Codes bzw. Präfixcodes genannt. Der Huffman-Code ist solch ein Präfixcode und eine Verbesserung gegenüber dem Fano-Code von 1949.

3.1.2. Huffman-Code

Der Huffman-Code findet sich in vielen älteren Kompressionsverfahren wie zum Beispiel dem bekannten JPEG-Verfahren der Joint Photographic Experts Group [11] oder MNP5 |[1] wieder bzw. kann in moderneren Verfahren wie JPEG 2000 [10] durch einen Schalter aktiviert werden. Aufgrund der umfangreichen Literatur die über den Huffman-Code existiert |[2], wird auf eine genauere Betrachtung verzichtet. Lediglich die Besonderheiten sollen hervorgehoben werden. In der Anwendung ist der Huffman-Code ein sehr schnelles Codierungsverfahren. Durch eine einfache Anfrage an eine Ersetzungstabelle können die Symbole der Signalquelle durch ihre entsprechenden Codewörter zur Laufzeit substituiert werden. Die Huffman-Ersetzungstabellen sind ohne großen Aufwand in Hardware realisierbar. Unter zu Hilfenahme von sogenannten Look-up-tables (LUTs) werden sehr hohe Taktraten erzielt. Anzumerken ist, dass die Codewörter in jedem Fall vorweg berechnet worden sind. Wie bereits erwähnt, ist es selten, dass die Auftretenswahrscheinlichkeiten der Symbole bekannt sind. Das heißt, dass sie auf irgendeine Weise approximiert werden müssen. Anhand von Trainingssequenzen (Testexemplaren) werden die Verteilungen der Symbole geschätzt. Für die Qualität der Huffman-Codes bedeuten variierende Wahrscheinlichkeiten und eine feste Ersetzungstabelle bzw. ein Satz von Ersetzungstabellen beinahe zwangsläufig suboptimale Kompressionsergebnisse.

Angenommen, die Auftretenswahrscheinlichkeiten der Symbole einer Signalquelle sind bekannt und eine Huffman-Ersetzungstabelle kann erzeugt werden, liefert das Huffman-Verfahren nur dann einen optimalen Code variabler Länge, wenn die Wahrscheinlichkeiten sich als

$$p_i = \frac{1}{2^i} \quad , i \in \mathbb{N}$$

darstellen. Die Ursache liegt darin begründet, dass Huffman immer eine ganze Anzahl Bits jedem Symbol zuordnet. So sollte zum Beispiel ein Symbol mit der Wahrscheinlichkeit $p = 0,3$ einen Code mit der Länge $1,7$ bit zugewiesen

[1] Microcom Network Protocol 5

[2] Die Werke von Salomon [30], Strutz [40], Sayood [32] [31] und Witten, Moffat, Bell [44] erläutern neben der Huffman-Codierung weitere gängige Verfahren. Die Arbeiten von Vitter [42] und Knuth [14] behandelt dynamische Varianten der Huffman-Codierung.

bekommen, da der Informationsgehalt

$$-log_2(0,3) \approx 1,7 \text{ bit}$$

beträgt. Das Huffman-Verfahren kann jedoch aufgrund der Ganzzahligkeit nur einen 2-bit-Code (oder 1-bit-Code) für das Symbol vergeben.

Adaptive Huffman-Codes

Mehrere adaptive Varianten wurden vorgeschlagen, die zur Laufzeit die Symbolwahrscheinlichkeiten approximieren. So können die Daten der Signalquelle zweimal verarbeitet werden. In einem ersten Durchgang werden die Vorkommnisse jedes einzelnen Symbols gezählt und in einem zweiten Durchlauf die Symbole mit Hilfe der angepassten Codeersetzungstabelle verarbeitet. Dieses Verfahren wird als semi-adaptiv bezeichnet und ist im Feldeinsatz oftmals zu langsam. Für diesen Zweck wurden voll-adaptive Verfahren entwickelt.

Zu Beginn der Codierung besitzen Codierer und Decodierer einen leeren Huffman-Baum. Die Struktur entspricht einem Binärbaum, an dessen Knoten (welche die empfangenen Symbole repräsentieren) die Auftretenswahrscheinlichkeiten gespeichert sind. Das Codewort wird durch traversieren von der Wurzel des Baumes zu den Symbolknoten erzeugt. Symbole mit hoher Wahrscheinlichkeit befinden sich nahe der Wurzel, während seltene Zeichen an den Blättern des Baumes zu finden sind. Ein großer Nachteil adaptiver Huffman-Codes besteht darin, dass für jedes neu gelesene Eingabewort der gesamte Baum überprüft und eventuell aktualisiert werden muss. Für eine Hardwareimplementierung ist es sehr aufwendig, solch ein dynamisches Verhalten zu imitieren. Die Sortier- und Einfügeoperationen verlangsamen die Gesamtverarbeitungsgeschwindigkeit erheblich|[3]. Für eine genaue Betrachtung verschiedener Varianten dynamischer Huffman-Codes sei auf [30] verwiesen.

[3]siehe Jamro [12]

3.1.3. Golomb-Code

Bereits 1966 wurde von S.W. Golomb [6] ein präfixfreier Code vorgeschlagen, der alle natürlichen Zahlen $n \in \mathbb{N}$ codiert und zur Übermittlung von Laufl\"angen bestimmt war. Zur Strategie des Golomb-Codes gehört die Annahme, dass große Zahlen seltener auftreten als kleine und somit einen längeren Code zugewiesen bekommen. Der Code besteht aus einem Satz von m Codetabellen. Zur Berechnung einer Tabelle werden in einem ersten Schritt zwei Parameterwerte q und r erzeugt.

$$q = \left\lfloor \frac{n}{m} \right\rfloor |^4 \tag{3.5}$$

$$r = n - q \cdot m \tag{3.6}$$

Die Werte q und r können als ganzzahliger Teil und Rest einer Division von $\frac{n}{m}$ betrachtet werden. Der Wert von q wird in unärer $|^5$ Darstellung codiert, während der Rest in $\lceil log_2 m \rceil$-bit-binär-Darstellung verarbeitet wird. Tabelle 3.1 veranschaulicht die Golomb-Codes für $n = \{0, \ldots, 9\}$ und $m = \{1, \ldots, 3\}$.

Zur besseren Lesbarkeit ist zwischen dem ganzzahligen Teil und dem Rest ein Punkt eingefügt worden.

Gallagher und van Voorhis haben in ihrer Arbeit [5] bewiesen, dass der Golomb-Code ein optimaler Code für ein Alphabet mit unendlich vielen Symbolen ist, die einer geometrischen Verteilung

$$F(X = n) = \rho^n (1 - \rho), \quad n \in \mathbb{N} \tag{3.7}$$

[4] Für $m = 0$ ist $q = n$ definiert

[5] Der unäre Code codiert eine Zahl n durch n viele Einsen und eine Null am Ende (oder umgekehrt) und bietet sich als Verfahren für die Golomb-Strategie an.

	Codewörter		
n	m=1	m=2	m=3
0	0·	0·0	0·00
1	10·	0·1	0·01
2	110·	10·0	0·10
3	1110·	10·1	10·00
4	11110·	110·0	10·01
5	111110·	110·1	10·10
6	1111110·	1110·0	110·00
7	11111110·	1110·1	110·01
8	111111110·	11110·0	110·10
9	1111111110·	11110·1	1110·00

Tabelle 3.1.: Golomb-Codetabelle

unterliegen, wenn für den Parameter m gilt:

$$m = \left\lceil -\frac{1}{log_2 \rho} \right\rceil . \tag{3.8}$$

3.1.4. Golomb-Rice-Code

Eine besondere Gruppe der Golomb-Codes sind die Tabellenspalten mit der Eigenschaft

$$m = 2^k. \tag{3.9}$$

Sie werden Rice-Codes bzw. Golomb-Rice-Codes (GR-Codes) genannt und gehen auf die 1977 von Robert F. Rice vorgestellte Arbeit [24] zurück |[6]. Trotz der starken Reduzierung der Codetabellen und der damit einhergehenden minimal schlechteren Kompressionsrate, ergeben sich besondere Vorteile bei der Implementierung der GR-Codes. Aus der Division in Gleichung (3.5) wird eine einfache Bitshift-Operation

$$q = n >> k, \tag{3.10}$$

deren Wert unär codiert wird. Die unäre Codierung ist wiederum sehr effizient mit einer Bitshift-Operation umsetzbar. Der Rest der Division wird durch eine

[6] nicht zu Verwechseln mit dem Rice-Codierer

Modulooperation abgebildet.

$$r \;=\; n \bmod k \tag{3.11}$$

Modulooperationen sind auf der Hardwareebene sehr einfache Aufgaben, die gelöst werden, indem lediglich die letzten k bit direkt ausgegeben werden.

Adaptive Golomb-Rice-Codes

In analoger Weise zu Gleichung (3.8) stellt sich die Frage, wie der optimale Parameter k ermittelt werden kann und welche Bedingungen gelten müssen. Als Symbolverteilung für ein unendliches Alphabet wird die geometrische Verteilung vorausgesetzt. Die Länge eines Codewortes ergibt sich aus:

$$l_n \;=\; 1 + k + \left\lfloor \frac{n}{2^k} \right\rfloor . \tag{3.12}$$

Daraus kann die durchschnittliche Länge eines Codewortes ensprechend der Gleichung (3.3) berechnet werden.

$$\bar{l} \;=\; \sum_{n=0}^{\infty} l_n (1-\rho)\rho^n \tag{3.13}$$

$$\;=\; 1 + k + (1-\rho) \sum_{d=0}^{\infty} d \cdot \sum_{n=0}^{2^k-1} \rho^{2^k \cdot d + n} \tag{3.14}$$

$$\;=\; k + \frac{1}{1-\rho^{(2^k)}} \tag{3.15}$$

Die Gleichung (3.15) sagt aus, dass die durchschnittliche Codewortlänge $\bar{l}$ immer gegen einen festen Wert konvergiert, der von den Paramtern ρ und k abhängig ist. Ein optimaler Code ist gegeben, wenn $\bar{l}$ minimal ist, d.h. ein optimaler Wert k^* bestimmt werden kann.

$$k_\rho^* \;=\; \arg\min_{k \geqslant 0} \{\bar{l}\} \tag{3.16}$$

Eine vollständige Herleitung ist in dem Artikel [13] von Kiely zu finden. Es besteht die Möglichkeit, den zur Laufzeit unbekannten Parameter ρ der Modellverteilung durch eine Maximimum-Likelihood-Schätzung $\widehat{\rho}$ wie in Abschnitt 2.22 vorgestellt zu berechnen [7]. Damit ist eine Grundlage geschaffen, um ein

[7] siehe dazu Salomon [28]

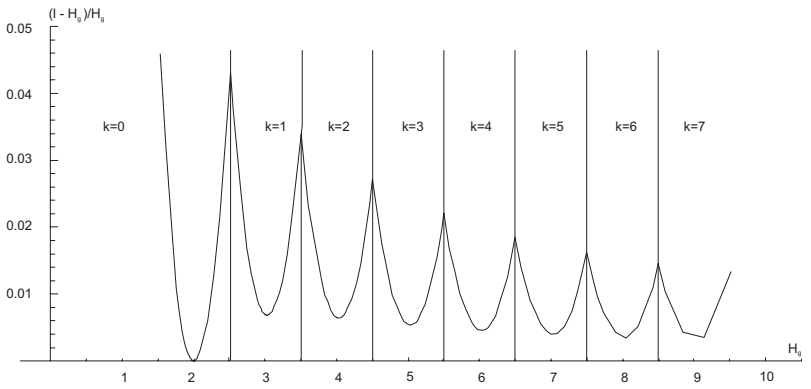


Abbildung 3.2.: relative Codierungsredundanz der Golomb-Rice-Codierung als Funktion zur Entropie H_g einer Signalquelle mit geometrischer Verteilung

optimales k_ρ^* zu ermitteln. Abbildung 3.2 veranschaulicht für verschiedene k_ρ^* die relative Codierungsredundanz der Golomb-Rice-Codierung gegenüber einer Signalquelle mit geometrisch verteiltem Alphabet. Es ist zu erkennen, dass die relative Redundanz für $1 \leqslant H_g \geqslant 3$ am größten ist und im Maximum ungefähr 4.5 % beträgt. Im Vergleich zu der relative Redundanz des Golomb-Codes ist der Golomb-Rice-Code im Schnitt $\approx 1 - 2$ % schlechter.

3.1.5. Längenlimitierte Codes

Anhand Tabelle 3.1 Spalte $k = 0$ ist eindeutig zu erkennen, dass kleine k für große n von Nachteil sind. Angenommen, das Alphabet der Signalquelle kennt $2^8 = 256$ Symbole, so ist es im ungünstigsten Fall möglich, dass ein Symbol $s_i = 255$ mit gewähltem $k = 0$ ein Codewort c_i der Länge $l_i = 256$ erzeugt. Die Darstellung von s_i wird um $N - \lceil log_2(N) \rceil$ bit expandiert. Für ein Alphabet mit $N = 2^{16}$ Symbolen werden 65.520 bits zusätzlich benötigt. Insbesondere zu Beginn eines Kompressionsdurchlaufs würde ein einfaches adaptives Verfahren unter dem unnötigen Speichermehrverbrauch leiden. Mit zunehmender Dauer des Durchgangs passt der Parameter k sich für große n an und die Häufigkeit einer Datenexpansion nimmt ab. Für Implementierungen ist es dennoch unerlässlich, eine maximale Bitbreite für die Codewörter zu definieren. Mit Hilfe sogenannter längenlimitierter Codes ist eine obere Schranke für die Codewortlänge l_i bestimmbar.

	Codewörter			
n	k=0	k=1	k=2	k=3
0	0·	0·0	0·00	0·000
1	10·	0·1	0·01	0·001
2	110·	10·0	0·10	0·010
3	1110·0010	10·1	0·11	0·011
4	1110·0011	110·0	10·00	0·100
5	1110·0100	110·1	10·01	0·101
6	1110·0101	1110·0101	10·10	0·110
7	1110·0110	1110·0110	10·11	0·111
8	1110·0111	1110·0111	110·00	10·000
9	1110·1000	1110·1000	110·01	10·001
10	1110·1001	1110·1001	110·10	10·010
11	1110·1010	1110·1010	110·11	10·011
12	1110·1011	1110·1011	1110·1011	10·100
13	1110·1100	1110·1100	1110·1100	10·101
14	1110·1101	1110·1101	1110·1101	10·110
15	1110·1110	1110·1110	1110·1110	10·111

Tabelle 3.2.: JPEG-LS-Codetabelle

JPEG-LS [8], der bekannteste Vertreter der modernen verlustfreien Kompressionsverfahren, verwendet einen limitierten GR-Code, der auf folgende Weise entsteht. Durch Vergleich mit einer um k binär verschobenen Konstante wird überprüft, ob ein Symbol n aus dem Bereich von $\{0, \ldots, 2^N - 1\}$ mit dem Golomb-Rice-Code und Parameter k oder als Binärzahl codiert wird. Das daraus resultierende Codewort c_i hat maximal l_{max} Bitstellen.

$$B = log_2(N)$$

$$c_i = \begin{cases} \text{Golomb-Rice-Code(n,k)} & \text{, falls } n < (l_{max} - B - 1) \cdot 2^k; \\ \text{unär}(l_{max} - B - 1) \ \& \ \text{binär}(i - 1) & \text{, sonst} \end{cases}$$

Für $k = 2$ aus Tabelle 3.2 ist zu erkennen, dass anstelle der 6 bit breiten äquivalenten Golomb-Rice-Codewörter, JPEG-LS eine unnötig lange 8-bit-Ersetzung für $n > 11$ durchführt. Eine verbesserte Darstellung ist in der Arbeit [38] von Roman Starosolski beschrieben worden. Ähnlich des Codierers von JPEG-LS ist

[8] Der Standard JPEG-LS basiert auf dem Verfahren LOCO-I. LOCO-I wurde von den Mitarbeitern der HP-Labs Weinberger, Seroussi und Shapiro 1996 entworfen und in dem Artikel [43] vorgestellt. Für eine genauere Beschreibung sei auf Kapitel 4 verwiesen.

ein Vergleich und eine zusätzliche Minimumoperation nötig.

$$\pi_k \quad = \quad \min\{(l_{max} - B) \cdot 2^k, N - 2^k)\} \tag{3.17}$$

$$c_i = \begin{cases} \text{Golomb-Rice-Code(n,k)} & \text{, falls } n < \pi_k; \\ \frac{\pi_k}{2^k} * \text{Einsen } \& \; bin\ddot{a}r(i - \pi_k) & \text{sonst} \end{cases}$$

Zusätzlich müssen die Bedingungen

$$l_{max} > N \tag{3.18}$$

und

$$0 \leqslant k \leqslant N \tag{3.19}$$

erfüllt sein. In Tabelle 3.3 ist die neue Codefamilie für $k = 0, 1, 2$ abgebildet, die im direkten Vergleich zu JPEG-LS kürzere oder gleich lange Codewörter erzeugt, jedoch keine längeren. Eine erneute Optimierung erhält man, indem der Rest r mit dem Truncated Binary Encoder verarbeitet wird. Diese Form erzeugt für bestimmte k und n noch einmal kürzere Codewörter. Eine bessere Ausnutzung der zur Verfügung stehenden Bits für r wird durch eine Zweiteilung des Zahlenbereichs für r erreicht. Der Wert von q wird weiterhin in unärer Darstellung codiert, während der Rest auf spezielle Art binär repräsentiert wird. Die ersten $2^{\lceil log_2 k \rceil} - k$ Zahlenwerte werden in $\lfloor log_2 k \rfloor$-bit-Darstellung codiert, der übriggebliebene Zahlenbereich wird in $\lceil log_2 k \rceil$ bit codiert. Dieses Schema ist auch unter dem Begriff Truncated Binary Encoding (TBE) oder Adjusted Binary Encoding bekannt. Die Ursache für die Verkürzung der Codes liegt darin begründet, dass für bestimmte k die Codes unvollständig sind bzw. der gesamte Darstellungsraum nicht maximal ausgeschöpft wird. Messungen aus Diagramm 5.3 zeigen jedoch, dass der Gewinn marginal ist (weniger als 0.01 bit pro pixel für $l_{max} > 16$) und der Aufwand für einen zusätzlichen Vergleichsoperator nicht gerechtfertigt ist.

Qualität modifizierter Golomb-Rice-Codes

Die vorgestellten Modifikationen haben zur Folge, dass das Alphabet der Signalquelle endlich ist. Daraus ergibt sich die Konsequenz, dass der modifizierte

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

Tabelle 3.3.: Modifizierte JPEG-LS-Codetabelle (Quelle: [38])

Golomb-Rice-Code kein optimaler präfixfreier Code im Sinne der Codierungsredundanz sein muss. Merhav, Weinberger und Seroussi haben in [20] bewiesen, dass die relative Abweichung von einem optimalen Golomb-Rice-Code im Maximum $\approx 4\,\%$ beträgt. Diese starke Abweichung tritt insbesondere bei Symbolquellen mit niedriger Entropie auf. Dies deckt sich mit den Überlegungen aus Abschnitt 3.1.4.

3.1.6. Arithmetische Codierer

Wie bereits erwähnt, kann die durchschnittliche Länge $\bar{l}$ nicht kleiner als eins werden, da die Symbole einzeln durch Codewörter ersetzt werden. Arithmetische Codierer überwinden dieses Handicap, indem sie die gesamte Eingabe als Block betrachten und einen Code für diesen erzeugen. Somit ist ihnen die Möglichkeit gegeben, eine deutlich bessere Annährung an die Quellenentropie zu finden. Das Prinzip der arithmetischen Codierung soll nur kurz vorgestellt werden. Der Algorithmus liest von der Eingabe Symbol für Symbol und nutzt deren kumulierte Verteilung um ein bestimmtes Intervall einzugrenzen. Ein Intervall wird durch eine Unter- und Obergrenze bestimmt. In einer inhaltlich äquivalenten Angabe

wird oftmals auch nur die Untergrenze und eine Länge des Intervalls angegeben.

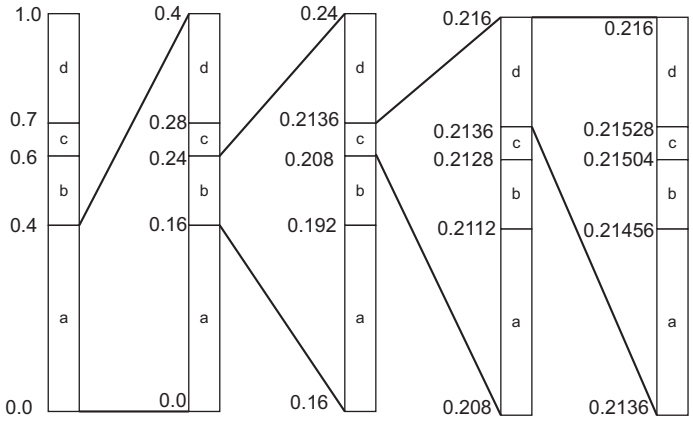


Abbildung 3.3.: Intervallzerlegung der arithmetischen Codierung (Quelle: [40])

Ein Beispiel aus dem Buch [40] soll als Anschauung dienen. Das Alphabet der Signalquelle kennt vier Symbole $s = \{a, b, c, d\}$ mit den entsprechenden Auftretenswahrscheinlichkeiten $\{0.4, 0.2, 0.1, 0.3\}$ und der kumulierten Verteilungsfunktion

$$F(x) = \sum_{x_i \leqslant x} p_i \tag{3.20}$$

$F(x) = \{0, 0.4, 0.6, 0.7, 1\}$. Das Startintervall liegt zwischen $(0, 1]$. Für jedes Symbol s_i werden drei Berechnungsschritte durchgeführt.

- $oI :=$ obere Intervallgrenze
- $uI :=$ untere Intervallgrenze
- $oI^* :=$ neue obere Intervallgrenze
- $uI^* :=$ neue untere Intervallgrenze

$$\begin{aligned} \triangle &= oI - uI \\ oI^* &= uI + \triangle \cdot F(i+1) \\ uI^* &= uI + \triangle \cdot F(i) \end{aligned}$$

Mit jedem gelesenen Symbol wird das Teilintervall kleiner. Die Codierung von Symbolen mit hoher Auftretenswahrscheinlichkeit bewirkt eine weniger starke Intervallverkürzung als solche mit seltenem Vorkommen. Nachdem alle Symbole eines Blocks gelesen worden sind, muss ein Codewort, das zwischen den Intervallgrenzen liegt, gesendet werden. Da die Intervallgrenzen

$$0.2136 \leqslant Code \leqslant 0.216$$

gebrochen rationale Zahlen sind, werden die Bits als negative Potenz zur Basis zwei interpretiert. Tabelle 3.4 illustriert die Entstehung des Codeworts, das an die Ausgabe als 00110111 gesendet wird.

binär	dezimal	Bit	Summe
2^{-1}	0.5	0	0
2^{-2}	0.25	0	0
2^{-3}	0.125	1	0.125
2^{-4}	0.0625	1	0.1875
2^{-5}	0.03125	0	0.1875
2^{-6}	0.015625	1	0.203125
2^{-7}	0.0078125	1	0.2109375
2^{-8}	0.00390625	1	0.21484375

Tabelle 3.4.: Ausgabecodewort der arithmetischen Codierung (Quelle: [40])

Der erste praktisch einsetzbare arithmetische Codierer wurde 1987 von Witten, Neal und Cleary vorgestellt und sukzessiv verbessert (siehe dazu eine aktuelle Abhandlung in [21]). Anstelle von Wahrscheinlichkeiten, die wie bereits erwähnt, zur Laufzeit nicht bekannt sind, wird in dem Verfahren mit absoluten Häufigkeiten gearbeitet, wodurch die arithmetische Codierung bestens für ein adaptives Vorgehen geeignet ist. Ähnlich zu den vorgestellten adaptiven Golomb-Rice-Codierern wird versucht, aus den bereits verarbeiteten Symbolen eine Abschätzung für die Auftretenswahrscheinlichkeiten der Symbole zu erhalten. Aus den Zählerständen für die Häufigkeit der einzelnen Symbole $c(s_i)$ werden die kumulierten Häufigkeiten

$$C(x) \quad = \quad \sum_{i \leqslant x} c(s_i) \tag{3.21}$$

in einer sortierten Liste $L = \{C(1), C(2), \ldots, C(N)\}$ festgehalten. Aus denen wird wiederum die kumulierte Verteilung der Häufigkeiten durch eine Division

$$F(i) \quad = \quad \frac{C(i)}{C(N)} \tag{3.22}$$

berechnet. In jedem Codierungsdurchgang müssen für jedes Symbol mindestens zwei Additionen, zwei Multiplikationen (siehe Gleichungen 3.21 und 3.21) und falls nötig eine Shift-Operation zur Renormalisierung der Intervallgrenzen durchgeführt werden. Am Beispiel aus Tabelle 3.4 wird deutlich, dass nach wenigen Symbolen viele Nachkommastellen zur genauen Erfassung der Intervallgrenzen nötig sind. Eine Idee von Witten, Neal und Cleary besteht darin, die ersten Bits auszugeben, sobald diese eindeutig bestimmt sind. Nur ein sehr geringer Verlust an Kompressionsgüte tritt auf, wenn anstelle exakter Fließkommaarithmetik die Nachkommastellen der Intervallgrenzen mit Integerwerten approximiert werden.

Die größte Schwäche der adaptiven arithmetischen Codierer besteht in dem Aufwand, der bei der Aktualisierung des Models entsteht. Mit jedem Symbol, das zur Codierung gelesen wird, ändert sich auch der ihm assoziierte Zählerstand, der die Häufigkeit des Auftretens misst. Automatisch ist auch die Liste der kumulierten Häufigkeiten veraltet. Für den ungünstigsten Fall, in dem die Häufigkeiten gleichverteilt sind, müssen für jedes Symbol im Durchschnitt die Hälfte der kumulierten Verteilungen neu berechnet werden. Das heißt, dass es für große Alphabete mit 2^{16} Symbolen bis zu 32768 Divisionen pro Symbol sein können. Eine praktische Umsetzung ist quasi ausgeschlossen.

Eine Minderung des Rechenaufwandes ist durch eine etwas komplexere Aktualisierungsprozedur möglich. Zu Beginn eines jeden Eingabeblocks ist aufgrund der wenigen bisher aktualisierten Häufigkeiten die Schätzung über die Symbolverteilung ungenau. Ab einem gewissen Punkt jedoch, ist ein quasi stationäres Verhalten zu beobachten, sodass bei nur marginaler Verschlechterung der Kompressionsrate nicht mit jedem Symbol die kumulierten Häufigkeiten neu berechnet werden müssen. Für Softwareimplementierungen ergeben sich durch diesen Kompromiss teilweise erhebliche Geschwindigkeitsvorteile. Moffat, Neal und Witten stellen in [21] ein vereinfachtes Verfahren vor, das ohne Multiplikationen und Divisionen auskommt. Der Preis ist jedoch eine sehr hohe Anzahl an

Additionen und Shift-Operationen, sodass der Rechenaufwand weiterhin hoch bleibt. Einer Verbreitung der sog. Shift-Add-Coder steht häufig eine zu starke Degradierung der Kompressionsrate im Verhältnis zu einer nur mäßigen Veringerung der Komplexität im Weg. Als Folge dessen, sind Verfahren, die auf Tabellenabfragen basieren, in den Vordergrund der Forschung gerückt.

3.1.7. Binäre arithmetische Codierer

Ein häufig zitiertes Beispiel für einen sog. binären Codierer ist der Q-Coder, der von Pennebaker et al. entwickelt und in dem Artikel [23] vorgestellt wurde. Das Verfahren verwendet ein Alphabet von nur zwei Symbolen und wird aus diesem Grund binärer arithmetischer Codierer genannt. Verursacht durch das kleine Alphabet, muss die Aktualisierung der kumulierten Verteilung lediglich über den Vektor $F(x) = P(s_i \leqslant x) = \{0, p(0), 1\}$ stattfinden. Die Ermittlung der neuen Intervallgrenzen wird stark vereinfacht. Obwohl der Q-Coder ursprünglich für die Codierung von binären Bildern entwickelt worden ist, kann das Verfahren auch für Alphabete mit $N > 2$ verwendet werden. Während der Binearisierung werden die Symbole den Blättern eines binären Baumes zugeordnet. Die Eingabe für den binären arithmetischen Codierer entsteht durch traversieren des Baumes von der Wurzel zum Symbolblatt. An diesem Punkt wird deutlich, dass der binäre Codierer zwar weniger komplex ist, aber für jeden besuchten Knoten |[9] (zur besseren Unterscheidung auch Bins gennannt) ausgeführt werden muss. Die Anordnung der Knoten innerhalb des Baumes spielt insofern eine Rolle, als dass Symbole mit hoher Häufigkeit nahe der Wurzel sein sollten, um die durchschnittliche Zahl der Bins zu minimieren. Dabei erhöht die Sortierung des Baumes |[10] die Komplexität des Verfahrens deutlich. Die binären Codierer sind dennoch sinnvoll, da sich die Komplexität des Verfahrens durch die Verwendung eines Binärbaums in $O(log_2(N))$ befindet |[11].

Hardwareumsetzung

Im Zusammenspiel mit den oben vorgestellten binären Entscheidungsbäumen, sind binäre arithmetische Codierer dazu geeignet, in einer parallelen Anordnung

[9] von der Wurzel zum Blatt

[10] Die binären Bäume können möglicherweise nach verschieden Verfahren der Graphentheorie vorsortiert werden.

[11] Siehe hierzu die Arbeiten von Fenwick [4] und Said [27]

als Hardwaremodule realisiert zu werden. Erste Konzepte und Implementierungen beschränken sich zumeist auf kleine Alphabete mit maximal vier Symbolen. Hauptsächlich werden durch feste Ersetzungstabellen, welche die arithmetischen Operationen (und deren Ergebnis) nachbilden, sog. quasi (binäre) arithmetische Codierer implementiert. Howard und Vitter stellen solch einen Codierer in der Arbeit *Practical Implementations of Arithmetic Coding* [8] vor. Das Grundprinzip ist einfach und besteht darin, dass die Nachkommastellen der Intervallgrenzen als Zustände eines endlichen Zustandsautomaten betrachtet werden. Nach einer Minimierung der Zustände lassen sich die Zustandsübergänge in Tabellen ablegen, die vorweg berechnet wurden. Durch Hinzunahme von Binärbäumen (in der Literatur Verteilungsbäume genannt) sind auch größere Alphabete möglich. Vertreter der table-driven (binären) arithmetischen Codierer sind der Q-coder |[12] und seine Abkömmlinge der QM-Coder |[12] und der MQ-Coder |[13], der im Standard JPEG 2000 Verwendung findet.

3.2. Dekorrelation

Durch Dekorrelation wird versucht, die Intersymbolredundanz zu minimieren. Bestehen in einem Signal statistische Abhängigkeiten zwischen den Symbolen, so ist es möglich, N_v Quellsymbole zusammenzufassen und auf neu eingeführte Symbole abzubilden. Die Entropie des neuen Alphabets nimmt dabei ab, je mehr korrelierte Symbole zusammengefasst werden. Eine wirkliche Reduktion der Entropie findet nicht statt. Lediglich eine Verbesserung der Abschätzung der Quellenentropie ist erreicht worden, solange die Menge an Information, die die Quelle abgibt, unverändert bleibt. Durch das Erkennen von Strukturen innerhalb der Quelldaten ist es möglich, die statistische Unsicherheit der Signalquelle zu reduzieren. Seien beispielhaft X, Y zwei Zufallsgrößen, $H(X)$ die Entropie der Zufallsgröße X und $H(Y|X)$ die bedingte Entropie von X und Y, dann wird $H(X,Y)$ mit

$$H(X,Y) \;=\; H(X) + H(Y|X) \tag{3.23}$$

als Verbundentropie zweier Zufallsgrößen bezeichnet. Sie spiegelt den mittleren Informationsgehalt der verbundenen Symbole wieder. Eine Herleitung von (3.23)

[12] patentrechtlich geschützt
[13] freie Version

findet sich in den Arbeiten von Sayood [31] und Strutz [40]. Zur Verdeutlichung von (3.23) sollen zwei Grenzfälle betrachtet werden. Sind zum einen X und Y nicht korreliert, folgt das $p(y|x) = p(y)$ gilt und die Verbundentropie $H(X,Y)$ aus der Summe der Einzelentropien zusammengesetzt ist.

$$H(X,Y) \quad = \quad H(X) + H(Y). \tag{3.24}$$

Wird zum anderen eine totale Abhängigkeit beider Zufallsgrößen zueinander ermittelt, ist die Verbundentropie gleich der Entropie von X

$$H(X,Y) \quad = \quad H(X), \tag{3.25}$$

da die bedinge Entropie $H(Y|X)$ den Wert Null annimmt. Am Ende der Berechnung muss die Verbundentropie mit der Anzahl zusammengefasster Symbole N_V noch normiert werden.

$$\frac{H(X,Y)}{N_V}$$

$$\frac{H(X)}{N_V} \leqslant \quad H(X,Y) \quad \leqslant H(X) \tag{3.26}$$

Dies bedeutet, dass je mehr Symbole, die statistisch voneinander abhängen, zusammengefasst werden, umso mehr kann die Entropie gesenkt werden.

Lauflängen

Die Lauflängen-Codierung ist ein Verfahren, das zur Präcodierung geeignet ist. Eine allgemeine Form der Lauflängen-Codierung beinhaltet die Zusammenfassung von gleichen Symbolen in einem Token (Tripel). Übermittelt werden nur das Symbol, dessen Lauflänge und ein Steuerzeichen. Der Decoder benötigt das Steuerzeichen, um aus dem normalen Decodierablauf eine Lauflängenangabe zu erkennen. Sollte ein Symbol sehr häufig auftreten, jedoch nur selten in einer Folge, würde die Lauflängen-Codierung versagen und sogar unnötig zusätzlichen Speicher bzw. Übertragungskapazität verbrauchen. Stattdessen kann eine Bit-Markierung für eindimensionale und eine Vierbaum-Codierung (engl.: quadtree coding) für zweidimensional Signale in Betracht gezogen werden. Das Sondersymbol wird in beiden Verfahren implizit übertragen, d.h. das lediglich

eine Information darüber übermittelt wird, ob das Sonderzeichen vorliegt oder nicht. Alle anderen Symbole werden wie gehabt gesendet.

Wörterbücher

Neben identischen Symbolen, die sich in unmittelbarer Nähe zu einander befinden, können auch wiederkehrende Sequenzen unterschiedlicher Zeichen zu Phrasen zusammengefasst und in einem Speicher abgelegt werden. Diese Form der Präcodierung geht auf die Arbeiten von Ziv und Lempel zurück. Mittelpunkt der Verfahren ist ein Speicher (auch Wörterbuch genannt) der einen Index auf die Sequenzen ausgibt. Das Wörterbuch kann zu Beginn leer sein und während der Eingabe der Symbole gefüllt werden oder vorinitialisiert sein. Die Größe des Wörterbuchs hat Einfluss darauf, mit welcher Wahrscheinlichkeit eine Phrase wiederentdeckt wird, was einen positiven Effekt auf die Kompressionsrate haben kann. Durch die notwendige Erweiterung des Speicherindexes zur Adressierung des Wörterbuches, werden die positiven Auswirkungen relativiert.

Blocksortierung

Ein relativ neues Verfahren wurde von Burrows und Wheeler 1994 [2] entwickelt |[12]. Anders als sequentielle Verfahren, wird die Eingabe blockweise verarbeitet. In einer Matrix werden die Symbole des Eingabeblocks zyklisch rotiert und zeilenweise abgespeichert. Die Zeilen der Matrix werden anschließend lexikographisch sortiert und die letzte Spalte der Matrix inklusive eines Indizes, der die Zeilennummer des ursprünglichen Eingabeblocks in der sortierten Matrix wiedergibt, ausgegeben. Die Eingabesymbole sind nun in Gruppen zusammengefasst, in denen ähnliche Symbole dicht beieinander liegen, wodurch eine gute Ausgangsposition für weitere Präcodierer gegeben ist. Häufig folgt der Burrows-Wheeler-Transformation (BWT) eine Move-To-Front-Codierung, die kürzlich aufgetretenen Symbolen kleine Indizes zuweist. Diese Indizes können anschließend mit dem Huffman-Verfahren oder arithmetisch codiert werden. Abschließend sei angemerkt, das die BWT gute Kompressionsergebnisse für Blockgrößen ab 100 kbyte erzielt. Zu berücksichtigen ist dabei der steigende Sortieraufwand.

[12] Aufgrund der sehr guten Kompressionsergebnisse wurde eine Implementierung der Burrows-Wheeler-Transformation mit in die Auswertung in Kapitel 6 aufgenommen.

3.3. Prädiktion

Eine weitere Möglichkeit die Intersymbolredundanz zu beseitigen, besteht darin, durch Beobachtung weniger Symbole in der unmittelbaren Nachbarschaft des aktuell zu encodierenden Symbols Vorhersagen auf Basis eines Modells zu treffen. Die Schätzung $v[n]$ des Modells zum Zeitpunkt n und der von der Signalquelle ausgegebene Wert $x[n]$ werden als Differenz bzw. als Schätzfehler (Residuum) $e[n]$ verrechnet.

$$e[n] \quad = \quad x[n] - v[n] \tag{3.27}$$

Der Grund für dieses Vorgehen liegt in der verstärkten Konzentration der Amplituden des Signals. Im Vergleich zum ursprünglichen Signal $X[n]$ ergeben sich so verbesserte Bedingungen für die Entropiecodierung. Im statistischen Sinne entspricht dies einer Verringerung der Varianzen σ_e^2 der Fehler gegenüber der Varianz σ_x^2 des originalen Signals. Ausgehend von Gleichung (3.27) können einige qualitative Aussagen über das Verhalten der Varianz σ_e^2, die es zu minimieren gilt, in Abhängigkeit zu σ_x^2 und der Autokorrelation von $x[n]$ gemacht werden. Für den einfachen linearen Prädiktor $v[n] := x[n-1]$ mit nur einem betrachteten Vorgänger folgt aus Gleichung (3.27)

$$\sigma_e^2 \quad = \quad \sigma_x^2 \cdot (2 - 2 \cdot r_{xx}[1]). \tag{3.28}$$

Der Wert $r_{xx}[1]$ ist der Autokorelationskoeffizient, der eine Auskunft über die Ähnlichkeit des Signals $X[n]$ bei einer Verschiebung um eine Stelle zu sich selbst gibt und aus

$$r_{xx}[m] \quad = \quad \frac{K_{xx}[m]}{K_{xx}[0]} \tag{3.29}$$

mit

$$K_{xx}[m] \quad = \quad E\left[x_n \cdot x_{n-m}\right] \tag{3.30}$$

berechnet wird (siehe Strutz [40]).

Gleichung (3.28) zeigt, dass ein Korrelationskoeffizient von $r_{xx} > 0.5$ eine Verminderung der Varianz des Fehlers gegenüber der Varianz des Originalsignals

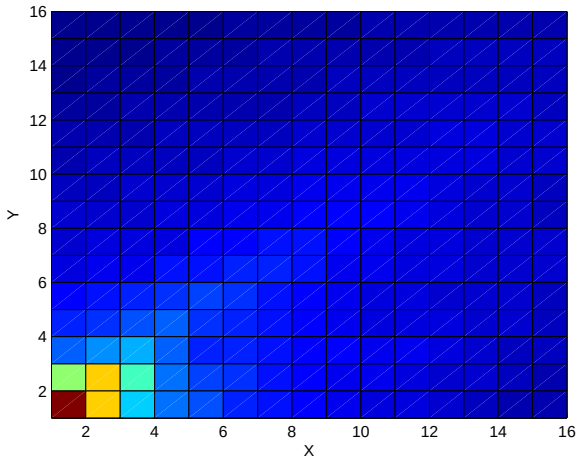


Abbildung 3.4.: Qualitative Darstellung der Korrelation benachbarter Pixel anhand des 8-bit-Grauwertbildes A.4(d) (rot:= starke Abhängigkeit, blau := geringe Abhängigkeiten)

bewirkt. Anhand der Abbildung 3.4 sei der statistische Zusammenhang der einzelnen Pixel auf Basis des Kameramannbildes A.5(b) qualitativ verdeutlicht. Die Korrelation wurde durch verschieben der Zeilen und Spalten um 16 Stellen zueinander ermittelt.

3.3.1. Prädiktion zweidimensionaler Signale

Für zweidimensionale Signale erfolgt eine Prädiktion identisch zum vorhergehenden Abschnitt aus einer gewichteten Überlagerung mehrerer Vorgänger, die als Symbole A,B,C,D einen kausalen Zusammenhang für den aktuellen Pixelwert X liefern (siehe Anhang A.1 Lossless JPEG). Aufgrund der sich häufig schnell

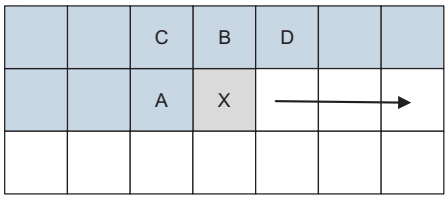


Abbildung 3.5.: Vorgängersymbole A,B,C,D ... für die Prädiktion

ändernden Signalstatistik kann eine bessere Vorhersage unter Einbeziehung von

Strukturinformationen, zum Beispiel einer Texturbeobachtung, getroffen werden. So verwendet das standardisierte Verfahren für verlustfreie Kompression JPEG-LS einen von drei Prädiktoren auf Basis einer einfachen Kantenerkennung (engl.: median edge detector).

$$v = \begin{cases} min(a,b), & \text{wenn } c \geqslant max(a,b); \\ max(a,b), & \text{wenn } c \leqslant min(a,b); \\ a+b-c, & \text{sonst.} \end{cases} \tag{3.31}$$

Prädiktionsschema 3.31 liefert $v = a$ falls eine horizontale Kante und $v = b$ falls eine vertikale Kante entdeckt wird. Für alle anderen Fälle wird $v = a + b - c$ als Vorhersage verwendet. Diese Form von Prädiktion wird als nichtlinear bezeichnet. Obwohl in gewisser Weise eine eindimensionale Verarbeitung eines zwei-

(a) Originalbild (b) Prädiktionsfehler

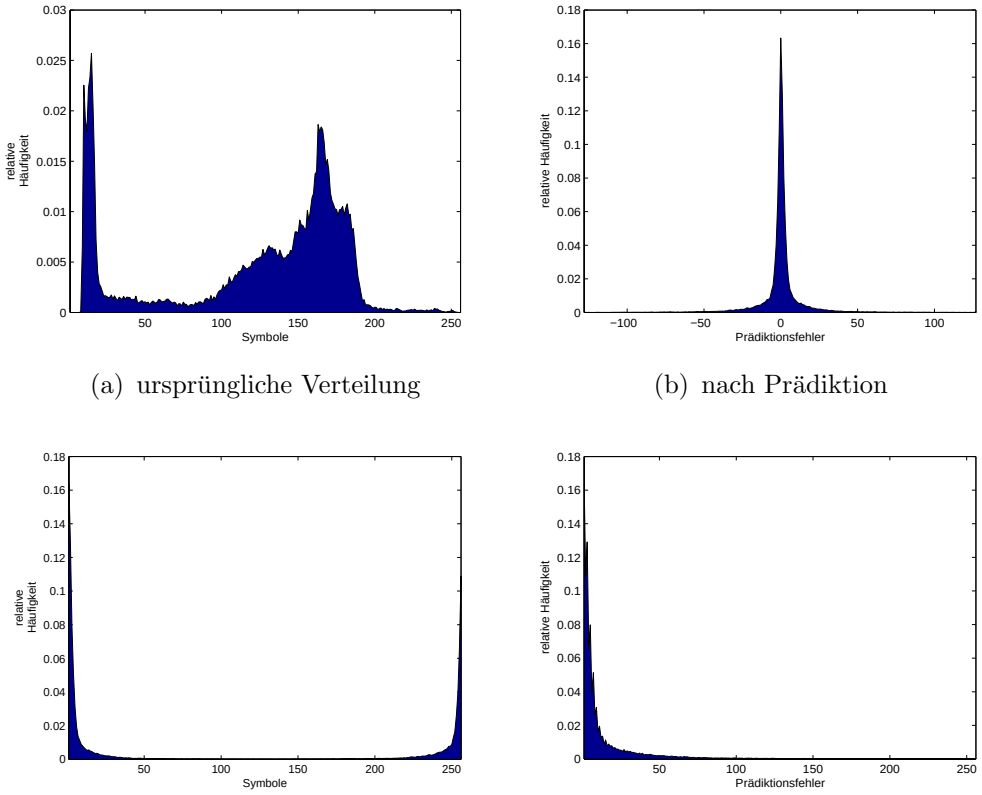
Abbildung 3.6.: Linearer Prädiktor $\frac{(a+b)}{2}$ auf das 8-bit-Grauwertbild A.5(b) angewendet

dimensionalen Bildes durchgeführt wird, erzielen Verfahren, die auf Prädiktion beruhen, erstaunlich gute Ergebnisse.

3.3.2. Verteilung der Residuen

War die Symbolverteilung zu Beginn der Präcodierung noch relativ gleichverteilt, so sind nach der Prädiktion die Symbole um einen Punkt gehäuft. Diese Form der Verteilung kann sehr gut durch eine geometrische Verteilung mit dem Parameter ρ modelliert werden. Am Beispiel des 8-bit-Grauwertbildes A.5(b) ist

die veränderte Symbolverteilung gut zu erkennen. Die Abbildung 3.7(b) zeigt eine deutliche Konzentration der Prädiktionsfehler. Weitere Berechnungsschritte müssen folgen, wenn eine Codierung der Prädiktionsfehler mit dem häufig verwendeten präfixfreien Golomb-Rice-Code durchgeführt werden soll. Eine



(a) ursprüngliche Verteilung

(b) nach Prädiktion

(c) nach Modulooperation

(d) nach Umsortierung

Abbildung 3.7.: Einfluss einer linearen Prädiktion auf die Symbolverteilungen

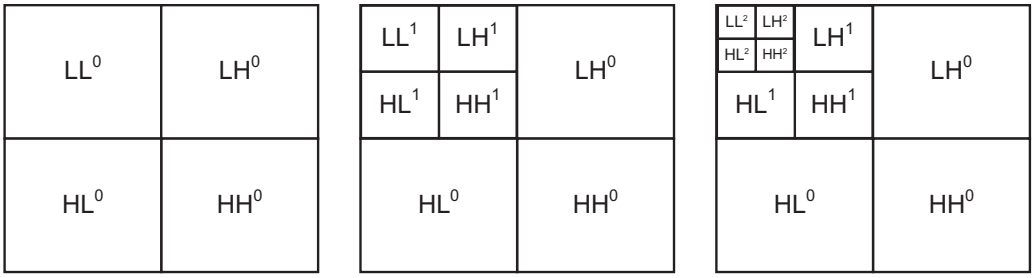
Modulooperation im Anschluss der Prädiktion verschiebt die Prädiktionsfehler in einen nicht negativen Wertebereich, sodass eine notwendige Bedingung der Golomb-Rice-Codierung erfüllt ist. Der Nachteil der Modulooperation besteht darin, dass der geometrische Verlauf der Verteilung durch die Anwendung des Operators verloren geht (siehe Abbildung 3.7(c)). Eine Verschiebung der Werte ineinander restauriert die geometrische Verteilung. Abbildung 3.7(d) veranschaulicht das Endergebnis nach Prädiktion, Modulooperation und Umsortierung.

3.4. Transformation

Das Grundprinzip jeder Transformation ist der Versuch, durch Umverteilung der Varianzen und Kovarianzen auf Basis einer orthogonalen Transformationsmatrix eine optimale Dekorrelierung des Eingangssignals zu erreichen. Die Karhunen-Loève-Transformation liefert solch ein vollständig dekorreliertes Signal. Die Berechnung ist jedoch zu aufwendig, als dass sie von praktischer Bedeutung wäre, da die Basisfunktionen für die Transformation vom Eingangssignal abgeleitet werden. Transformationen mit fester Basisfunktion liefern zwar suboptimale Ergebnisse, sind jedoch praktisch umsetzbar. Grundlage des bekannten JPEG Kompressionsstandards ist die sog. diskrete Kosinus-Transformation (DCT). Sie beruht ähnlich der Fouriertransformation auf einem harmonischen Transformationskern, weißt jedoch bessere Eigenschaften als letztere hinsichtlich der Datenkompression auf. Der Name der DCT bezieht sich auf die kosinusförmige Basisfunktion, die auch als Transformationskern bezeichnet wird. Aufgrund der noch besseren Eigenschaften der Wavelettransformation hat diese die DCT aus dem JPEG Standard verdrängt. Die Basisfunktionen der Wavelettransformation werden aus einer Mutterwelle (engl.: mother wavelet) durch Skalierung und Verschiebung abgeleitet. Das einfachste Wavelet ist das Haar-Wavelet, das eine geringe Komplexität bezüglich seiner Anwendung aufweist, jedoch aufgrund seiner groben Struktur schlechtere Transformationsergebnisse liefert. Viele verschiedene Wavelettransfomrationskerne sind seit der Arbeit von Debauchies vorgeschlagen worden, wobei das nach ihr benannte Debauchies-5/3-Wavelet am vorteilhaftesten für die verlustfreie Kompression ist. Dieser Transformationskern wird ebenfalls vom Standard JPEG2000 verwendet. Aufgrund der Analogie zwischen Filterbänken und der Wavelettransformation, kann letztere als eine Filterung mit einem Hochpass und einem Tiefpass verstanden werden. Die Filter für das Debauchies-5/3-Wavelet sind in Gleichung (3.32) aufgestellt. Die Werte a und d werden oftmals als Approximation und Detail der diskreten Eingangswerte $x[n]$ zum Zeitpunkt n bezeichnet.

$$\begin{aligned} a &= \frac{1}{8}(-x[2n-2] + 2x[2n-1] + 6x[2n] + 2x[2n+1] - x[2n+2]) \\ d &= \frac{1}{4}(x[2n] - 2x[2n+1] + x[2n+2]) \end{aligned} \tag{3.32}$$

Durch Unterabtastung und zweifache Ausführung für die Zeilen und Spalten



(a) (b) (c)

Abbildung 3.8.: Verschiedene Stufen der Wavelettransformation

können zweidimensionale Signale gefiltert werden. Abbildung (3.9) verdeutlicht die hintereinander ablaufende Ausführung der Transformation auf die Zeilen des Bildes und auf die Spalten der ersten Transformationsergebnisse. Das Gesamtergebnis ist eine Matrix von der Größe des ursprünglichen Bildes mit vier gleich großen Teilmatrizen, welche die Werte LL, LH, HL und HH enthalten. Durch erneute Ausführung der Transformation auf vorhergehende Transformationsergebnisse der LL-Submatrix, wird das ursprüngliche Bild in immer höhere Teilbänder zerlegt |[13]. Im Verlauf von hohen zu tiefen Teilbändern sind immer weniger Detailkoeffizienten ungleich null. Dadurch wird eine Datenkompression erleichtert (siehe Abbildung A.2(a)). Eine effizientere Methode, um die

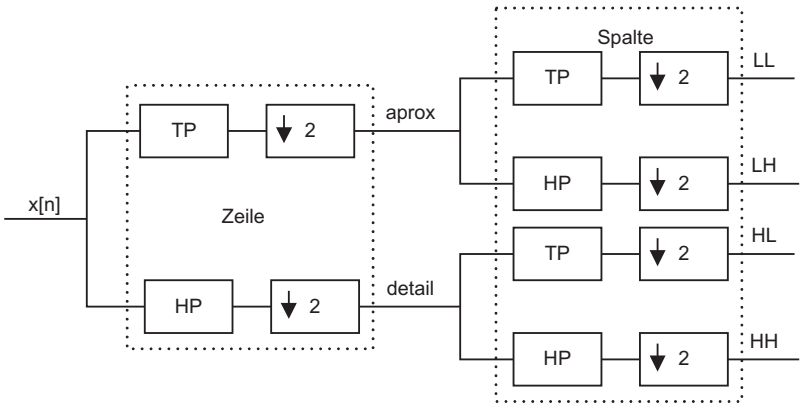


Abbildung 3.9.: Zweidimensionale Wavelettransformation mit Filter

Approximations- und Detailwerte zu berechnen, ist durch das Lifting-Schema von Sweldens [41] gegeben. Die Zahl der arithmetischen Operationen reduziert

[13]steigende Indexzahl

sich dabei von sieben Additionen/Subtraktionen und vier Bitshift-Operationen zu vier Additionen/Subtraktionen und zwei Bitshift-Operationen. Eine zusätzliche Ersparnis ergibt sich bei der Berechnung der Approximationswerte durch die Einbeziehung bereits ermittelter Detailwerte aus vorhergehenden Stufen.

$$a = x[2n] - \frac{1}{4}(d[2n-1] + d[2n+1]) \tag{3.33}$$

$$d = -x[2n+1] + \frac{1}{2}(x[2n] + x[2n+2]) \tag{3.34}$$

Letztlich müssen die Koeffizienten, die als Ergebnis aus der Wavelettransformation entstanden sind, effizient codiert werden, da eine Transformation noch keine Daten komprimiert. Die in Abschnitt 3.2 vorgestellte Quadtree-Codierung könnte verwendet werden. Für die Wavelettransformation ist jedoch der EZW-Algorithmus |[14] besser geeignet, da er die örtlichen Beziehungen zwischen den verschiedenen Teilbändern ausnutzt (siehe Abbildung A.2(b)). Neben der örtlichen Abhängigkeit der Koeffizienten wird auch die Wahrscheinlichkeit, dass die Werte von tieferen Teilbändern zu höheren abnehmen, berücksichtigt. Weiterhin ist der EZW-Algorithmus einer der ersten, die eine progressive Übertragung ermöglichen. Diese spezielle Form der Codierung liefert einen Datenstrom, der mit zunehmender Übertragungsdauer der komprimierten Daten, die Qualität des restaurierten Bildes erhöht. Eine vollständige Rekonstruktion, die für eine verlustfreie Kompression nötig ist, wird durch eine vollständige Übertragung erreicht. Nachteil des EZW und des darauf aufbauenden Verfahrens SPIHT |[15] ist die Vielzahl an Speicheroperationen, die zur Berechnung des Codes nötig sind.

3.5. Kriterien zur Kompressionsbewertung

Vergleichskriterien sind unerlässlich, um die Güte verschiedener Kompressionsverfahren zu bewerten. Dabei gibt es mehrere Möglichkeiten, die Effektivität des zu überprüfenden Kompressionsverfahrens zu bestimmen. Etabliert hat sich die Angabe der Kompressionsrate.

[14] Der EZW-Algorithmus wurde in der Arbeit *Embedded image coding using zerotrees of wavelet coefficients* [34] von Shapiro vorgestellt.

[15] Das Verfahren SPIHT (engl.: Set Partitioning in Hierarchical Trees) wurde in der Arbeit *A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees* [29] von Said und Pearlman gezeigt.

$$\text{Kompressionsrate} \quad = \quad \frac{L_u}{L_c} \tag{3.35}$$

Gleichung (3.35) stellt ein einfaches Verhältnis zwischen der Anzahl der verwendeten Bits im unkomprimierten und im komprimierten Zustand her. Häufig wird eine Angabe in Bits pro Symbol, wobei ein Symbol ein Pixel bedeutet, verwendet. Die Einheit ist Bits pro Pixel (bpp).

$$\text{Bitrate} \quad = \quad \frac{L_u}{N} \tag{3.36}$$

Ein Nachteil der Angaben aus den Gleichungen (3.35) und (3.36) ist der, dass sie nicht additiv sind. In der Analyse von Kompressionsverfahren ist es oftmals wünschenswert, diese in Subkomponenten zu zerlegen und deren Eigenschaften auf das Gesamtsystem in einer additiven Weise zu betrachten. Mit der Angabe des Kompressionsgewinns

$$\text{Kompressionsgewinn} \quad = \quad 100 \cdot ln\left(\frac{L_u}{L_c}\right) \tag{3.37}$$

kann man diese Unzulänglichkeit umgehen. Da in dieser Arbeit nur verlustfreie Verfahren betrachtet werden, werden Qualitätsmaße, die Rekonstruktionsfehler berücksichtigen, vernachlässigt.

4. Stand der Technik

In den vorangegangen Kapiteln sind die statistischen Modelle und Prinzipien zur Präcodierung von Symbolen vorgestellt worden. Dabei ging es um eine Minimierung der statistischen Abhängigkeiten der Symbole zueinander. Der Präcodierung schließt sich die Entropiecodierung an, zu deren Gruppe zum einen die rechenintensive arithmetische Codierung und zum anderen die Codierung mit präfixfreien Codes gehören. In dem nun folgenden Abschnitt werden vier verlustfreie Kompressionsverfahren vorgestellt, die aus der großen Zahl verschiedener Implementierungen in einer Vorauswahl heraus gefiltert wurden. Die Selektion basiert dabei vorrangig auf der Komplexität des Verfahrens ohne die Kompressionsrate außer Acht zu lassen. Einfluss auf die Vorauswahl hatten ebenfalls Patentrechte und insbesondere die Dokumentation der Algorithmen. JPEG-LS/LOCO-I ist der aktuelle Standard (Stand: 2007) für verlustfreie Datenkompression und dient gleichzeitig als Referenz bezüglich der Kompressionsrate und der Komplexität. SFALIC ist ein Hybrid aus JPEG-LS und einer einfachen differentiellen Kompression wie zum Beispiel DPCM. Der Algorithmus von SFALIC besticht durch eine hohe Datenrate bei guten Kompressionsergebnissen. Das Verfahren CCSDS 122.0 des Beratungsgremiums Consultative Committee for Space Data Systems wurde speziell für den Luft- und Raumfahrtsektor entworfen und basiert auf einer Wavelettransformation. Der letzte hier vorgestellte Algorithmus ist das bekannte Gzip/deflate. Das Verfahren wurde aufgrund der bereits existierenden kommerziellen und akademischen Hardwareimplementierungen ausgewählt, mit denen sehr hohe Datenraten möglich sind.

4.1. JPEG-LS/LOCO-I

Mit Einführung des neuen Standards JPEG-LS für verlustfreie und beinahe verlustfreie Kompression setzte die Gremien der ISO/ITU auf ein Verfahren, dass das wenig verbreitete lossless JPEG ersetzen sollte. Aus einer Reihe von Vorschlägen wurde LOCO-I [43] als Basisalgorithmus für den neuen verlustfreien

Standard ausgewählt. Ausschlaggebend war die geringe Komplexität des Verfahrens im Vergleich zu dem besser komprimierenden CALIC-Algorithmus und die guten Kompressionsergebnisse, die nur unwesentlich schlechter als die der arithmetischen Codierer sind. LOCO-I basiert auf einer nichtlinearen Prädiktion (siehe Abschnitt 3.31) mit einer anschließenden kontextadaptiven Golomb-Rice-Codierung. Zu Beginn eines Durchlaufs werden die vorangegangenen Pixel

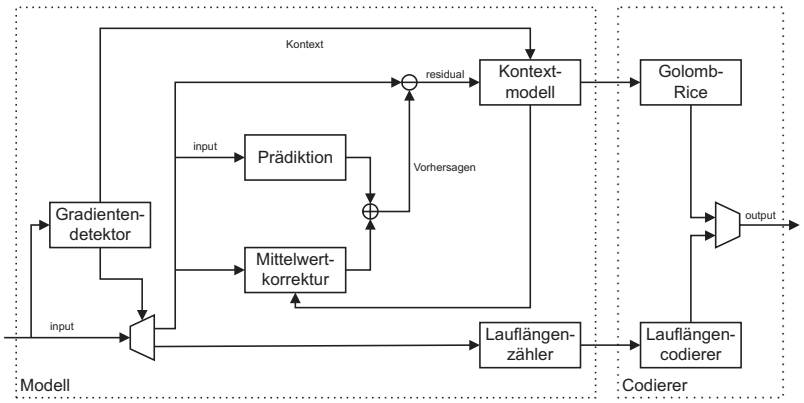


Abbildung 4.1.: Schema des JPEG-LS Datenflusses

mit einem Gradientendetektor analysiert. Dabei wird entschieden, ob ein relativ homogener Verlauf vorliegt und der Lauflängenmodus aktiviert wird oder weiterhin im sog. regulären Modus der nichtlinearen Prädiktion verbleibt. Gleichzeitig wird durch den Gradientendetektor der Kontext zur Adaption des Modells und der Golomb-Rice-Codierung ermittelt. Durch Quantisierung der drei Gradienten reduziert sich die Zahl der möglichen Kontexte auf 729. Durch das Weglassen des Vorzeichens verringert sich die Anzahl noch einmal um die Hälfte. Für jeden Kontext werden die absoluten Werte der Prädiktionsfehler, die Häufigkeit mit der der Kontext aufgerufen worden ist und zwei Werte zu Mittelwertkorrektur gespeichert. Zur Berechnung des Parameters k der Golomb-Rice-Codierung wird angenommen, dass für den mittleren Prädiktionsfehler genau k Bits zur Speicherung benötigt werden. Die Mittelwertkorrektur im regulären Modus sorgt dafür, dass die Prädiktionsfehler von einem Mittelwert befreit werden, der durch die Gruppierung in Kontextmengen entstehen kann.

4.2. SFALIC

Ähnlich zu JPEG-LS ist SFALIC ein sequentielles Kompressionsverfahren. Es beruht auf einer linearen Prädiktion mit anschließender kontextadaptiver Golomb-Rice-Codierung. In der Veröffentlichung [37] wird der Algorithmus von Starosolski für die Anwendung in bildgebenden Verfahren der Medizintechnik vorgestellt. Aufnahmen aus CT, MRT etc. ähneln den Sensordaten der Fernerkundung dahingehend, dass sie in großen Mengen anfallen und viel Übertragungs- und Speicherkapazität aufgrund ihrer Dimension und Bittiefe benötigen. In diesem Zusammenhang ist SFALIC (engl.: simple, fast adaptive lossless image compression) mit der Zielgabe entwickelt worden, eine hohe mögliche Datenrate während der Kompression zu erlauben. Nichtsdestoweniger konnte eine allzustarke Degeneration der Kompressionsrate verhindert werden. Neun Prädiktoren stehen zur Auswahl, wovon die ersten acht dem abgelösten Lossless-JPEG-Standard entnommen sind (siehe Anhang A.1). Die Auswahl des Prädiktors v erfolgt statisch zu Beginn der Kompression, wobei an den Bildrändern die Prädiktoren angepasst werden. Während der Prädiktion kann es dazu kommen, dass der Prädiktionsfehler $e[n] = x[n] - v[n]$ den Wertebereich von $[0, (2^N - 1)]$ verlässt. Um die in Abbildung 3.7(b) dargestellte Verteilung der Prädiktionsfehler über dem Bereich von $[-2^N + 1, 2^N - 1]$ zu vermeiden, wird in einem Folgeschritt $e_{mod}[n] = e[n]\ mod\ 2^N$ berechnet. Die Abbildungsvorschrift

$$e_{reorder}[n] = \begin{cases} 2 \cdot e_{mod}[n] & \text{, falls } e_{mod}[n] < 2^{N-1}; \\ 2(2^N - e_{mod}[n]) - 1 & \text{, sonst} \end{cases} \tag{4.1}$$

bewirkt, dass die für eine Golomb-Rice-Codierung ungünstige Verteilung von $e_{mod}[n]$ (Abbildung 3.7(c)) einen monoton fallenden Verlauf wie in Abbildung 3.7(d) annimmt.

Howard und Vitter haben eine Methode in der Arbeit [9] entwickelt, die ein einfaches statistisches Modell der Daten adaptiv erstellt. Der Kontext, in dem sich der aktuelle Pixel $x[n]$ befindet und auf Grundlage dessen der Parameter k der Golomb-Rice-Codierung ermittelt wird, ist durch den Prädiktionsfehler des vorhergehenden Pixels $x[n-1]$ bestimmt. Für jeden der 2^N Kontexte, werden N Zähler z_i bereitgehalten. Der Zählerstand des Zählers z_i gibt die kumulierte absolute Codewortlänge an, die erreicht wird, wenn alle Symbole mit dem Pa-

rameter $k = i$ codiert werden. Entsprechend der Gleichung (3.16) wird für die Codierung des aktuellen Prädiktionsfehlers das i gewählt, das die Codewortlänge minimiert. Das heißt, dass der Zählerindex i mit dem kleinsten Zählerstand als neuer Codierungsparameter für das $n + 1$-te Symbol an den Golomb-Rice-Codierer übergeben wird. Anschließend werden die Zählerstände um $l_i(x[n])$ erhöht. SFALIC führt gegenüber dem Verfahren von Howard und Vitter mehrere Verbesserungen ein, die zum einen eine schnellere Anpassung des Parameters k erlauben und zum anderen den notwendigen Speicherbedarf für die Zählerstände minimieren. Zu Beginn der Kompression kann es vorkommen, dass durch gleiche Zählerstände ein zu kleiner Zählerindex i gewählt wird, sodass die Golomb-Rice-Codierung mit kleinem k eine unerwünschte Datenexpansion verursacht. SFALIC wirkt dem entgegen, indem bei mehreren minimalen Zählerständen der Zähler mit dem größten i den Parameter der Golomb-Rice-Codierung bestimmt. Aufgrund des exponentiell fallenden Verlaufs der Verteilung der Residuen (siehe Abbildung 3.7(d)) kommen einige Kontexte seltener vor als andere. Durch die Gruppierung der Kontexte in sog. Kontextmengen, welche die Verteilung der Prädiktionsfehler wiederspiegeln, kann der Speicheraufwand von $\mathcal{O}(2^N)$ auf $\mathcal{O}(N^2)$ gesenkt werden.

4.3. CCSDS 122.0

CCSDS ist ein Zusammenschluss mehrerer nationaler und internationaler Raumfahrtagenturen zur Etablierung gemeinsamer Standards im Luft- und Raumfahrtsektor. Die Ausgabe CCSDS 122.0 [36] ist die neueste Empfehlung zum Thema verlustfreier und verlustbehafteter Kompression. Im Vergleich zum älteren Standard 121 basiert CCSDS 122.0 nicht mehr auf dem Rice-Verfahren, sondern auf einer diskreten Wavelettransformation und einer Bitebenencodierung. Dies hat zum Vorteil, dass eine blockorientierte progressive Übertragung möglich ist. Im Falle einer verlustbehafteten Kompression wird das Debauchies-9/7-Wavelet in einer Fließkommadarstellung als Transformation vorgeschlagen, deren Transformationsergebnisse anschließend quantisiert werden. Für eine verlustfreie Datenkompression muss eine ganzzahlige Version des Debauchies-9/7-Wavelets verwendet werden |[1]. Die anschließende Codierung der Transformationsergebnisse geschieht mit einem vereinfachten Zero-Tree-Verfahren.

[1] Warum das Debauchies-5/3-Wavelet verworfen wurde, kann nicht nachvollzogen werden.

Abbildung 4.2.: Datenflussdiagram des CCSDS 122.0 Standards [36]

4.4. Gzip/deflate

Gzip/deflate ist eine sehr populäre Implementierung einer auf ein Wörterbuch basierenden Präcodierung in Kombination mit einer Huffman-Codierung. Das Verfahren geht auf den Deflate-Algorithmus zurück, der eine Verbund aus modifizierter LZ77 und Huffman-Codierung darstellt und aufgrund von patentrechtlichen Problemen entworfen wurde. Ein in zwei verschiedene große Teile gegliederter Speicher, den die Eingabesymbole in FIFO-Ordnung passieren, dient zum einen als Wörterbuch und zum anderen als Look-ahead-Buffer. Beginnend mit dem ersten Symbol im Look-ahead-Speicherbereich wird nach einem identischen Symbol im Wörterbuch gesucht. Ist dieses gefunden worden, wird das aktuelle mit dem nachfolgenden Symbol des look-ahead Speicherbereichs verknüpft und nach der Zeichenkette innerhalb des Wörterbuchs gesucht. Dieser Vorgang wird solange wiederholt, bis die Suche fehlschlägt. Ausgegeben wird die Stelle im Wörterbuch und die Länge des gefundenen Strings. Die statische Huffman-Codierung verwendet zwei verschiedene Codetabellen für jeweils den Wörterbuchindex und die Länge der Zeichenkette.

5. Implementierung

Im vorherigen Abschnitt wurden verschiedene Kompressionsverfahren vorgestellt, von denen LOCO-I aufgrund seiner guten Kompressionrate und seiner vereinfachten Algorithmen zum Standard JPEG-LS für ausschließlich verlustfreie Kompression erklärt wurde. SFALIC verringert die Komplexität gegenüber JPEG-LS noch einmal, ohne eine zu starke Degration der Kompressionrate zu verursachen. Es sei noch einmal erwähnt, dass die Sensordaten mit einem Pixel pro Takt prozessiert werden müssen. Vier bzw. acht Kanäle sollen auf einem FPGA (engl.: field programmable gate array) realisiert werden. Hinsichtlich des MFC-Projektes besteht die Hoffnung, mittels Kompression der Sensordaten eine allgemeine Verbesserung der Leistungsfähigkeit zu erzielen. Beide Systeme sind durch Zeilenkameras gekennzeichnet, deren Bittiefe bis zu 14 bit beträgt. Je nach Einsatzort werden die Sensoren mit bis zu 10 kHz ausgelesen. Die Anzahl der Pixel je Datenausgang des Sensors kann bis zu 6.000 betragen. Dies entspricht einer möglichen Datenrate von bis zu 60 Msamples/s.

5.1. Hardwareabschätzung

Im folgenden Abschnitt wird erörtert, welches der in Kapitel 4 vorgestellten Verfahren am besten für eine Realisierung geeignet ist. Dies geschieht zum einen durch eine Abschätzung des Ressourcenverbrauchs, wobei auf hardwarespezifische Besonderheiten und Engpässe eingegangen wird. Zum anderen wurden Simulationen durchgeführt, welche die Kompressionsrate der einzelnen Algorithmen für die Sensordaten von Fernerkundungssatelliten und des MFC-Projektes näher beleuchten. Die Ergebnisse der Simulationen finden sich im Kapitel 6 wieder. Die Zusammenführung der Simulationsergebnisse und der Hardwareabschätzung führen zu der Matrix 5.2 auf deren Grundlage die Entscheidung für eine Implementierung fällt. Bevor jedoch das endgültige Ergebnis vorgestellt wird, muss eine genauere Hardwareabschätzung erfolgen.

5.1.1. Transformationen

Zur Berechnung einer Debauchies-5/3-Wavelettransformation sind nach dem Lifting-Schema vier Additionen und zwei Bitshift-Operationen nötig. Für die Ausführung des Debauchies-9/7-Wavelets werden sieben Additionen/Subtraktionen und drei Bitshift-Operationen gebraucht. Aufgrund der nur marginal besseren verlustfreien Kompression des Debauchies-9/7-Wavelets gegenüber dem Debauchies-5/3 ist letztere Transformation wegen der um 75 % geringeren Komplexität vorzuziehen. Als problematisch für die Datenrate gilt bei der Wavelettransformation der Speichertransfer der Zeilen und Spalten, sowie der Zwischenergebnisse. Mit einer Zerlegung des Eingangsbildes in Teilblöcke der Größenordnung von 16×16 bis zu 64×64 Pixel und der Hinzunahme von SRAM-Speichermodulen, können die Zeilen der CCDs in Koeffizienten der Transformation umgewandelt, zwischengespeichert und parallel verarbeitet werden. Durch Veränderungen in der Anordnung der Berechnungseinheiten kann der Speichertransfer reduziert werden.

Ein von EADS Astrium entwickeltes Kompressionsmodul für verlustfreie und verlustbehaftete progressive Kompression mit dem Namen CWIC (engl.: constant rate image compressor) orientiert sich stark an SPIHT. Zwei ASICs und externer SRAM-Speicher sind für ein Kompressionsmodul nötig und erzielen eine Datenrate von 11 Msamples/s, wobei ein Sample in 16 bit codiert sein kann. Aufgrund der Neuausrichtung der Luft- und Raumfahrtgemeinschaft hin zum neuen Standard CCSDS 122.02 Ende des Jahres 2005 wurde von EADS Astrium ein neues Kompressionsmodul auf der Basis zweier Actel FPGAs entwickelt, die ebenfalls auf externe Speicher angewiesen sind. Aufgrund der geringeren Komplexität von CCSDS 122.0 sind Datenraten von über 40 Msamples/s möglich.

Weitere Information liefert eine Dissertation [25] aus dem Jahr 2002 zum Thema Waveletkompression auf der Grundlage des Debauchies-5/3-Wavelet und modifizierter Zero-Tree-Codierung mit optionaler arithmetischer Codierung auf der Basis eines FPGA-Entwicklungsboards. Die Ergebnisse dieser Arbeit und der hohe Hardwareaufwand des CWIC-Kompressionsmoduls sind ein Beleg dafür, dass eine Realisierung auf der Basis einer Transformation mit anschließender Codierung als nicht realisierbar erachtet wird. Für das MFC-Projekt kommt die Transformation aufgrund fehlender externer Speicher ebenfalls nicht in Frage.

5.1.2. Wörterbücher

Der Algorithmus von Gzip/deflate benötigt für den Look-ahead-Buffer und das Wörterbuch einen Speicher, über dem viele Suchoperationen durchgeführt werden müssen. In diesem Zusammenhang wird Content-Addressable-Memory (CAM) verwendet. Anders als bei einem gewöhnlichen RAM, wird dem CAM ein Datenwort übergeben, nach diesem im Speicher gesucht wird. Nach erfolgreicher Suche gibt dieser den Index der Fundstelle im Speicher aus. FPGAs sind wegen ihrer dynamischen Strukturen in der Lage verschiedene CAM-Konfigurationen nachzubilden. Speziell in den zur Verfügung stehenden Xilinx FPGAs kann der assoziative Speicher mit Hilfe der Shiftregister (SLR16), SRAM (Select Blockram) und der Look-up-Tabellen der Logikblöcke realisiert werden. Ein frei nutzbarer CAM-IP-core für Xilinx FPGAs auf der Basis von SRAM ist in der maximalen Konfiguration von 512 bit Breite und 4096 Tiefe einsetzbar. Der assoziative Speicher ist somit recht klein aber dennoch verwendbar. Aufgrund des ausreichend vorhandenen SRAM beim Virtex2 und beim Virtex4 FPGA sind genügend Ressourcen für vier bzw. acht Kompressionskanäle vorhanden. Auch wegen der geringen Latenz von einem Takt für Leseoperationen und zwei Takten für Schreiboperationen des Assoziativspeichers wird eine Realisierung der Gzip/deflate-Kompression für möglich erachtet und der Aufwand als relativ gering eingeschätzt.

5.1.3. Prädiktion

Die Testbilder der DLR-Serie weisen bezüglich der Korrelation benachbarter Pixel verschieden starke Ausprägungen auf. Während im MFC-Testbild A.4(f) für Nachbarpixel mit mehr als zehn Stellen Entfernung noch starke Korrelationen feststellbar sind, zeigen sich im Testbild A.4(d) moderatere Abhängigkeiten (siehe Abbildung 3.4).

Die Wahl des Prädiktors bzw. dessen Parameter hat einen entscheidenden Einfluss auf die Qualität der Dekorrelation des Eingangssignals und der Effizienz der Codierer. Da sich komplexere Prädiktoren mit dynamischer Nachjustierung aufgrund der hohen Anforderungen nicht eignen, wurde das Hauptaugenmerk

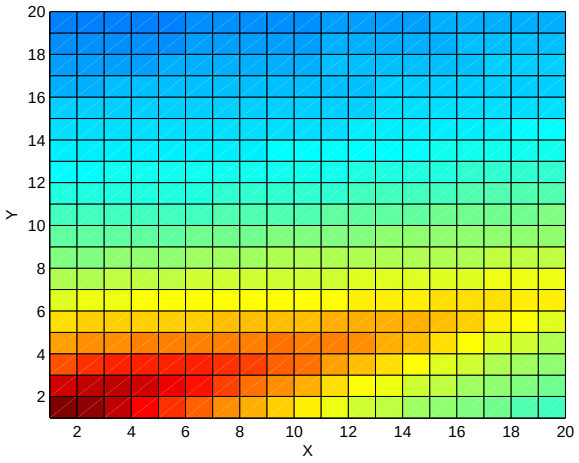


Abbildung 5.1.: Autokorrelationskoeffizienten r_{xx} von A.4(f) qualitativ (rot= starke Abhängigkeit, blau = geringe Abhängigkeiten)

auf etablierte Prädiktoren gelenkt. In durchgeführten Simulationen hat sich der Prädiktor

$$v = \frac{(A+B)}{2} \tag{5.1}$$

mit dem besten Verhältnis aus Komplexität und Effektivität durchgesetzt. So konnte der Prädiktor aus Gleichung (5.1) auch gegen den nichtlinearen Prädiktor MED aus dem Standard JPEG-LS bestehen. Für die Waterloo-Testserie

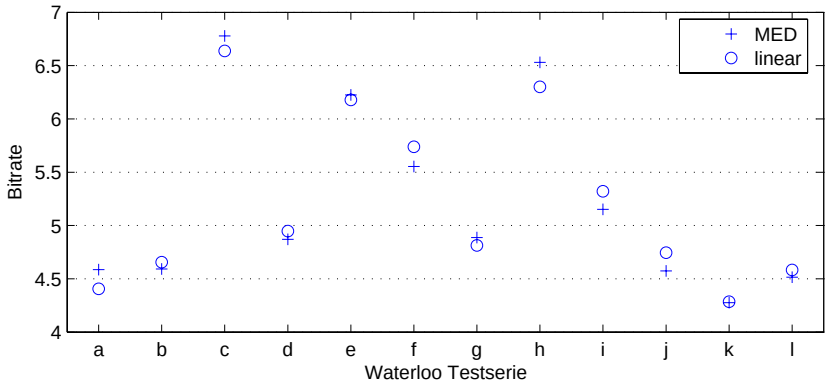


Abbildung 5.2.: JPEG-LS Prädiktor im Vergleich zum Prädiktor $v = \frac{(A+B)}{2}$. Der Test wurde mit den Grauwertbildern aus Abschnitt A.5 durchgeführt.

verschlechterte sich die Kompressionsrate im Durchschnitt um weniger als 0,03 bpp (siehe Abbildung A.2). Für die DLR-Testbilder (8 bit) wurde sogar nur eine Verringerung um 0.0061 bpp gemessen. Dementsprechend wird die Implementierung des linearen Prädiktors aus Gleichung (5.1) bevorzugt und mit einem geringen Realisierungsaufwand bewertet.

5.1.4. Arithmetische Codierung

Wie bereits erwähnt, liegt der große Vorteil der arithmetischen Codierer darin begründet, dass sie für jede Wahrscheinlichkeitsverteilung von zueinander unabhängigen Symbolen einer Signalquelle eine optimale Codierung finden. Durch die Methode der Intervalleingrenzung wird ein Code generiert, der vollständig frei von Redundanz ist. Aufgrund der im theoretischen Ansatz vorhandenen komplexen Rechenoperationen müssen von Anfang an Abstriche gemacht werden. Aufgrund der einfacheren Handhabung von binär arithmetischen Codierern sind mehrere Hardwareimplementierungen auf dieser Basis entstanden. Eine etwas ältere tabellarische Übersicht aus dem Jahr 2001 ist dem Artikel von Stefo et al. [39] entnommen, zu welcher neuere Implementierungen ergänzt wurden.

Implementierung	Kontext-tiefe	Taktrate	bit/Takt	Technologie
Marks [18]	7	75 Mhz	0,85	ASIC, 0.35 μm
Kuang	10	25 Mhz	0,12	ASIC in 0.8 μm
Stefo [39]	0	32 Mhz	8	FPGA
CABAC [35]	2	263 Mhz	0,3	ASIC, 0.18 μm
CABAC [17]	2	100 Mhz	1	FPGA
CABAC [17]	2	255 Mhz	1	ASIC, 0.25 μm

Tabelle 5.1.: Ausgabecodewort der arithmetischen Codierung (Quelle: [39])

Aufgrund kommerzieller Anreize konzentriert sich ein Forschungsschwerpunkt der letzten Jahre auf den Standard ITU-T | ISO/IEC H.264/AVC zur Kompression von Videodaten. Interessant ist das Modul CABAC, das für kontextadaptiver binärer arithmetischer Codierer steht. Zwei aktuelle Implementierungen sind der Tabelle 5.1 hinzugefügt worden. CABAC ist ein Vertreter der tabellenbasierten Verfahren und ähnelt dem MQ-Coder. Trotz der teilweise besseren Kompressionsraten gegenüber präfixfreien Codes wird die Komplexität als zu

hoch angesehen, als dass die erforderlichen Datenraten eingehalten werden können (siehe hierzu die Quellen [22] und [26]). Dies gilt insbesondere für Alphabete mit bis zu 2^{16} Symbolen. Nur in einem parallelen Ansatz, der viel Chipfläche bzw. Chips verbraucht, könnten die Anforderungen an die Datenrate (16 bit/Takt) erfüllt werden.

5.1.5. Präfixfreie Codes

Präfixfreie Codes, wie der Huffman-Code oder der Golomb-Rice-Code, sind sehr schnell in ihrer Ausführung und leicht zu erzeugen. Dies gilt insbesondere für den Golomb-Rice-Code mit dessen sehr günstigen Eigenschaften für eine Hardwareimplementierung. In Abschnitt 3.1.4 wurde der Golomb-Rice-Code diesbezüglich ausführlich beschrieben. Da für große Alphabete der Huffman-Code mit volladaptiven Modellen zu komplex ist, wird der modifizierte Golomb-Rice-Code aus Abschnitt 3.1.5 trotz seiner schlechteren Kompressionsergebnisse vorgezogen, wobei die Komplexität des Golomb-Rice-Codes als gering eingestuft wird. Exemplarisch sind in Abbildung 5.3 und dem vergrößerten Ausschnitt in Ab-

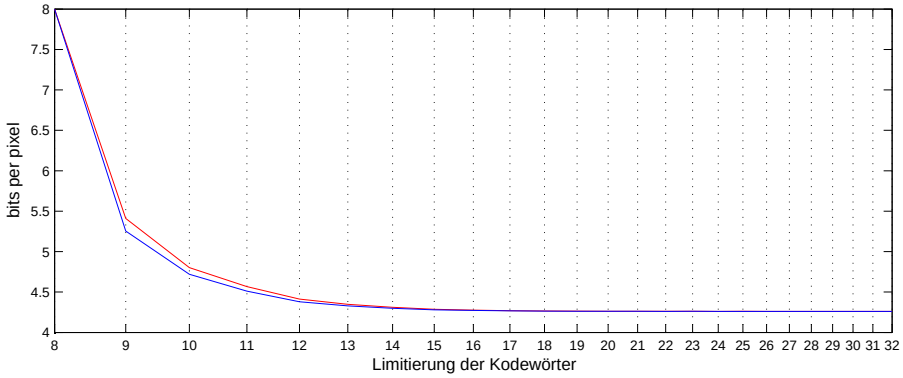


Abbildung 5.3.: Bitraten des 8-bit-Grauwertbildes A.5(k) mit vollständiger modifizierter und unvollständiger modifizierter Golomb-Rice-Codetabelle als Funktion der maximalen Codewortlänge l_{max}

bildung A.1 die Ergebnisse zweier Simulationen abgetragen. Zum einen wurde untersucht, inwiefern ein vollständiger modifizierter Golomb-Rice-Code gegenüber dem Minderaufwand eines unmodifizierten Golomb-Rice-Codes sinnvoll ist. Zum anderen wurde das Verhalten beider Codevarianten auf eine Veränderung

der Codelimitierung gemessen. Dabei hat sich gezeigt, dass der Mehraufwand für die Vervollständigung des längenlimitierten Golomb-Rice-Codes nicht lohnend ist. Bezüglich der Längenlimitierung kann gesagt werden, dass für 8-bit-Grauwertbilder ein Wert größer 16 bit keine nennenswerte Steigerung der Kompressionsrate liefert. Ähnliches gilt für 16 bit Grauwertbilder. Eine maximale Codewortlänge von 24 bit gilt hier als ausreichend (siehe Abbildung A.3(a) und A.3(b)).

5.2. Wahl der Implementierung

Im vorherigen Abschnitt wurde gezeigt, dass die Verfahren, deren Präcodierung auf der Basis einer Wörtbucherindizierung oder Prädiktion stattfindet, weniger komplex sind und dementsprechend weniger Ressourcen verbrauchen. Andererseits ist eine Transformation, insbesondere die Wavelettransformation, eine Voraussetzung für eine progressive Übertragung. Ein Merkmal, das für die Datenfernübertratung sehr interessant ist. Desweiteren bietet die Wavelettransformation die besten Signal-Rausch-Verhältnisse am Ende einer Decodierung. Da in dieser Arbeit jedoch nur verlustfreie Verfahren betrachtet werden, verliert die Wavelettransformation an Bedeutung. Weiterhin wurde gezeigt, dass präfixfreie Codes der arithmetischen Codierung im Grad der Komplexität weit überlegen sind. Ihr Nachteil ist jedoch, dass die Codierungseffizienz rapide fällt, sobald die statistischen Voraussetzungen nur noch ungenügend erfüllt sind. Welches Verfahren als Vorlage für eine Implementierung herangezogen werden soll, wird anhand einer Bewertungsmatrix ermittelt. Aufgrund der geforderten Echzeitfähigkeit wird dem Realisierungsaufwand bzw. der Komplexität gegenüber der Kompressionsrate eine höhere Bedeutung zugestanden. Daraus folgt, dass die Komplexität mit 1,5 gewichtet wird. Das Gewicht der Kompressionsrate beträgt Eins. Die Skala reicht von Eins bis Drei, wobei Drei der beste Wert ist. Mit 6,5 Zählerpunkten ist das Ergebnis der Bewertung von JPEG-LS, SFALIC,

	JPEG-LS	SFALIC	CCSDS 122.0	Gzip/deflate
Komplexität	2	3	1	3
Kompressionsrate	3	2	3	1
Ergebnis	6	6,5	4,5	5,5

Tabelle 5.2.: Auswahlverfahren für das zu realisierende Kompressionsverfahren

CCSDS 122.0 und Gzip/deflate zu Gunsten von SFALIC ausgefallen (siehe Tabelle 5.2).

5.3. Hardwareimplementierung

Im folgenden Abschnitt wird die Hardwareimplementierung des SFALIC-Algorithmus vorgestellt, die aufgrund von geringfügigen Abweichungen zur besseren Unterscheidung zum Original SFALIC-HW genannt wird. SFALIC-HW wurde zusammen mit dem Hardwarebetriebssystemkonzept von Krutz [16] entwickelt. Die Vorteile, die sich daraus ergebenen, sind eine Abstraktion der Benutzerlogik von der Hardwareinfrastruktur und eine vereinfachte Wiederverwendung der Module bei veränderter Hardwaregrundlage. Die Freiheiten einer Hardwarebeschreibung bleiben, anders als bei C-ähnlichen Hardwarebeschreibungssprachen wie HandelC, MitrionC etc., erhalten. Abbildung 5.4 zeigt schematisch die Anordnung der einzelnen Module des Betriebssystems und der Benutzerlogik. Das

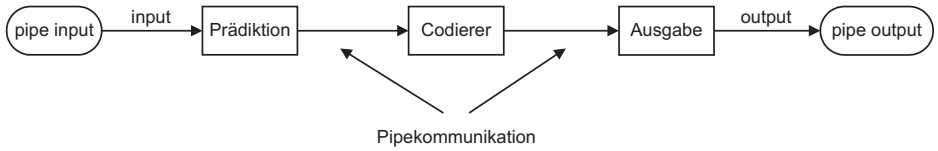


Abbildung 5.4.: SFALIC-HW Module (Schema)

Kompressionsmodul SFALIC-HW wird in drei Teilmodule gespalten, um eine separate Entwicklung und Validierung zu ermöglichen. Über Pipe-Kommunikation des Hardwarebetriebssystems werden die Sensordaten und die Zwischenergebnisse blocksynchron an das nachfolgende Modul übertragen. Das Modul zur Ausgabe der Codewörter reiht die erzeugten Codewörter variabler Länge in 8-bit oder 16-bit-Grenzen ein und stellt eine gewünschte Bit- und Byteordnung her.

5.3.1. Prädiktion

Das für die Prädiktion zuständige Hardwaremodul muss mit zwei Parametern instanziert werden (siehe Codeausschnitt 5.1). Zum einen mit einem Wert für die maximale Breite der Zeilenkamera (`MAX_SIZEX`) und zum anderen mit einem Wert für die Anzahl der Bits je Eingangsdatenwort (`WIDTH`). `WIDTH` ist von 1 bis

16 frei einstellbar. Die Signale `sizeX` und `sizeY` werden zur Laufzeit bestimmt und geben die tatsächliche Größe des zu komprimierenden Bildes wieder.

```
1  entity sfalic_predictor is
2    generic (
3      MAX_SIZEX : integer;
4      WIDTH     : integer
5      );
6    port (
7      sizeX       : in  std_logic_vector(15 downto 0);
8      sizeY       : in  std_logic_vector(15 downto 0);
9
10     -- PIPE Kommunikation Anfang
11         (verkürzte Darstellung, siehe Anhang)
12     -- PIPE Kommunikation Ende
13     );
14 end;
```

Listing 5.1: Schnittstelle des VHDL-Prädiktors

Sobald das erste gültige Datenwort anliegt und dem Prädiktor der Beginn der Abarbeitung angezeigt wird, sind konstant drei Takte nötig, bevor das erste Zwischenergebnis an den Codierer weitergeben werden kann. Da die Pipekommunikation blockweise funktioniert, werden die Eingangsdaten intern in einem FIFO-Speicher zwischengespeichert, sodass sich die Latenz je nach Konfiguration des Zwischenspeichers um weitere ein bis drei Takte erhöht.

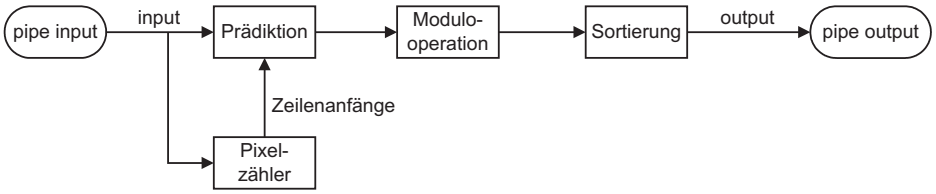


Abbildung 5.5.: Datenfluss des Prädiktors von SFALIC-HW (Schema)

In der ersten Zeile als auch bei den Anfängen jeder weiteren Bildzeile kann der Prädiktor $v_8 = \frac{A+B}{2}$ nicht verwendet werden. Abhängig vom gewählten Parameter `sizeX` wird dem Prädiktor der Beginn einer neuen Zähler signalisiert, woraufhin dieser den passenden Prädiktor $v_1 = 0$, $v_2 = A$,$v_3 = B$ oder $v_7 = \frac{A+B}{2}$ auswählt. Siehe dazu den VHDL-Code in *CD/HDL/src/sfalic_praedictor.vhdl*.

```
1  -- calculate the predictor
2  predictor_calc : process (input_clock, s_reset)
3  begin
4    if s_reset = reset_active then
5        s_Pred0 <= (others => '0');
6        s_Pred1 <= (others => '0');
7        s_Pred2 <= (others => '0');
8        s_Pred7 <= (others => '0');
9        s_pixel_X_pl1 <= (others => '0');
10       sfifo_in_read_ack_pl1 <= invalid;
11   elsif rising_edge(input_clock) then
12       s_Pred0 <= s_pixel_X;
13       s_Pred1 <= s_pixel_A;
14       s_Pred2 <= s_pixel_B;
15       s_Pred7 <= Pred7(s_pixel_A, s_pixel_B, WIDTH);
16       s_pixel_X_pl1 <= s_pixel_X;
17       sfifo_in_read_ack_pl1 <= sfifo_in_read_ack;
18   end if;
19 end process;
```

Listing 5.2: Ausschnitt aus dem Modul VHDL-Prädiktor

Die Modulooperation und die Sortierung der Residuen nach dem Schema aus Abbildung 3.7(d) und der Gleichung (4.1) sind ebenfalls unter *CD/HDL/src/s-falic_praedictor.vhdl* zu finden.

5.3.2. Codierer

Der Codierer ist in Abbildung 5.4 dargestellt und verdeutlicht die Entwurfsentscheidungen, die notwendig sind, um Datenwörter mit einem Pixel pro Takt zu codieren. Zu jedem Takt muss ein gültiger Codierungsparameter k dem Encoder bekannt sein. Da allein die Berechnung von k mindestens einen Takt in Anspruch nimmt und mit jedem Eingangswort ausgeführt wird, können nur in jedem zweiten Takt die Datenwörter codiert werden. Eine parallele Berechnung der Codewörter für jedes k umgeht die zeitliche Beschränkung indem mehr Chipfläche benötigt wird. Ein Multiplexer am Ende der Codierung wählt abhängig vom Parameter k das richtige Codewort aus und leitet es an den Ausgang weiter. Die von den k Encodern ermittelten Codewortlängen $l_k(c_i)$ werden je nach Kontext an die richtige Kontextmenge (Bucket) übertragen. Dort erhöhen sie die entsprechenden Zählerstände. Der Index des Zählers mit dem kleinsten Zählerstand liefert den Codierungsparameter k. Der hier vorgeschlagene Entwurf unterscheidet

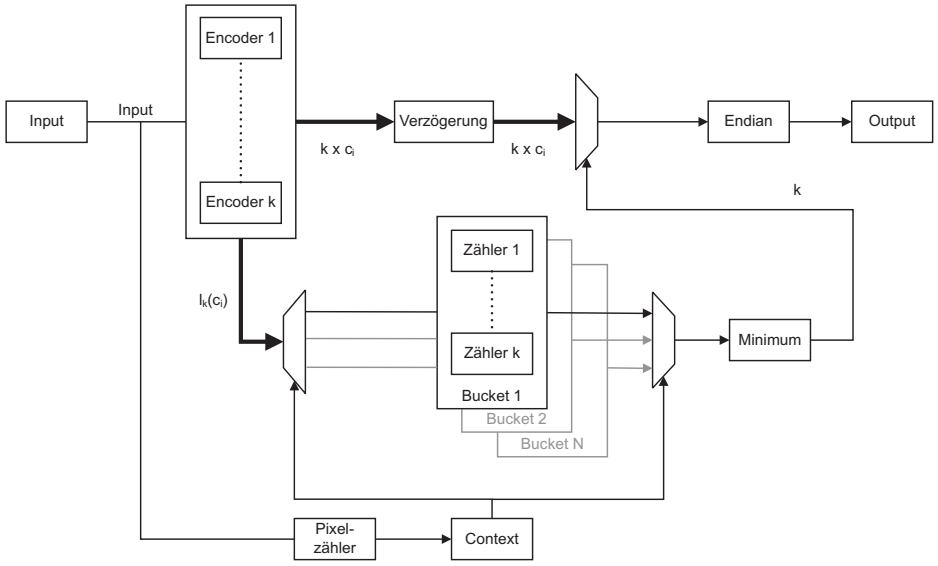


Abbildung 5.6.: Datenfluss des Encodierers von SFALIC-HW (Schema)

sich von der ursprünglichen Version von SFALIC dahingehend, dass die Kontextmengen mit jedem Eingangswort aktualisiert werden. SFALIC verwendet ein adaptives Aktualisierungsschema, das im Laufe der Abarbeitung Aktualisierungen der Kontextmengen weniger häufig durchführt. Die Kompressionraten sind für SFALIC-HW dementsprechend besser. Optional kann eine Kopfzeile mit Zusatzinformation dem komprimierten Datenstrom vorangestellt werden, welche die zur Dekompression notwendige Information wie Größe des Bildes, Zählerschwellwert, Bittiefe und die verwendete maximale Codewortlänge enthält. Dazu muss das Signal `no_header` den Wert Null besitzen. `MAX_SIZEX` und `WIDTH` liefern wie in Codeausschnitt 5.1 Informationen darüber, wie gross die Zeilenkamera ist und wieviel Bits je Eingangsdatenwort erwarten werden. Der Wert des Generics `CNT_WIDTH` legt den Wertbereich der Zähler innerhalb der Kontextmengen fest. Bei welchem Zählerstand die Counterwerte halbiert werden, ist abhängig von `TRESHOLD`. `TRESHOLD` muss sich dabei innerhalb der Grenzen von 0 bis $2^{\texttt{CNT\_WIDTH}} - 1$ bewegen. Da es sich hier um einen Codierer handelt, dessen Codewörter längenlimitiert sind, muss die maximale Codewortlänge durch den Wert des Generics `CODE_LIMIT` angegeben werden. Für die interne Verarbeitung der Eingangsdatenwörter und Zwischenergebnisse wird eine maximale Datenbreite durch `MAX_WIDTH` definiert. In *CD/HDL/src/sfalic_agre.vhdl* werden die Hauptkomponenten `sfalic_encoder` (*CD/HDL/src/sfalic_encoder.vhdl*) und

`sfalic_context_model` (*CD/HDL/src/sfalic_context_model.vhdl*) instanziert. Das Modul `sfalic_encoder` übernimmt die Aufgabe des längenlimitierten Golomb-Rice-Codierers, während in `sfalic_context_model` das dynamische Verhalten des Kompressionsverfahren gesteuert wird. `Sfalic_context_model` benötigt dafür die Module `sfalic_bucket` (*CD/HDL/src/sfalic_bucket.vhdl*) und `sfalic_minimum` (*CD/HDL/src/sfalic_minimum.vhdl*), wobei eine N große Anzahl des Moduls `sfalic_bucket` instanziert wird. Da die Kontexte in einem exponentiellen Schema gruppiert werden $(1, 2, 4, 8 \ldots, N-1)$ |[1], entspricht N aus Abbildung 5.6 der Bitbreite eines Eingangsdatenwortes (`WIDTH`). In der untersten Hierarchieebene von `sfalic_agre` werden innerhalb von `sfalic_bucket` k viele Instanzen der `Sfalic_counter`-Komponente (*CD/HDL/src/sfalic_counter.vhdl*) eingebunden.

```
1  entity sfalic_agre is
2    generic (
3      MAX_SIZEX  : integer;
4      WIDTH      : integer;
5      CNT_WIDTH  : integer;
6      TRESHOLD   : integer;
7      MAX_WIDTH  : integer;
8      CODE_LIMIT : integer
9      );
10   port (
11     sizeX     : in std_logic_vector(15 downto 0);
12     sizeY     : in std_logic_vector(15 downto 0);
13     no_header : in std_logic;
14
15     -- PIPE Kommunikation Anfang
16        (verkürzte Darstellung, siehe Anhang)
17     -- PIPE Kommunikation Ende
18     );
19 end;
```

Listing 5.3: Schnittstelle des VHDL-Golomb-Rice-Codierers

5.3.3. Ausgabe

Wie in den zwei Modulen zuvor, werden auch im letzten Modul die Eingangsdatenwörter in einem FIFO zwischengespeichert. Erst wenn die Codewörter den

[1] siehe Abschnitt 4.2

Zwischenspeicher passiert haben, werden sie in einem vier mal `MAX_WIDTH` großen Ringspeicher abgelegt. Ein Ringspeicher dient dazu, die Bits des präfixfreien Codes in feste Grenzen einzuordnen. Sobald eine Mindestanzahl an Bits geschrieben worden ist, wird mit der Ausgabe der `WIDTH` bit breiten Datenwörter begonnen. Abbildung 5.7 veranschaulicht den Ringspeicher. Mit Hilfe des Generics `BIGENDIAN` kann die Bitordnung bestimmt werden.

```
1  entity sfalic_output_unit is
2    generic (
3      WIDTH     : integer;
4      MAX_WIDTH : integer;
5      BIGENDIAN : valid_type
6      );
7    port (
8      sizeX : in std_logic_vector(15 downto 0);
9      sizeY : in std_logic_vector(15 downto 0);
10
11     -- PIPE Kommunikation Anfang
12        (verkürzte Darstellung, siehe Anhang)
13     -- PIPE Kommunikation Ende
14
15     );
```

Listing 5.4: Schnittstelle des VHDL-Ausgabemoduls

Ringspeicher

Kernbaugruppe des Ausgabemoduls ist der Ringspeicher. Ein Lese- und ein Schreibprozess, die jeweils über einen Index verfügen, die auf die richtigen Schreib- und Leseposition innerhalb des Ringspeichers verweisen. Exemplarisch sei der Schreibprozess aus *CD/HDL/src/sfalic_output_unit.vhdl* kurz erläutert. In zwei `for loop`-Umgebungen werden die Signale am Eingang des Ausgabemoduls auf eine Stelle im Speicher gelenkt. Damit diese Operation korrekt ausgeführt wird, müssen mehrere Bedingungen wie in den Codezeilen 22,27 und 30 des Codeausschnitts 5.5 zu erkennen |[2], erfüllt sein. So darf der Schreibindex den Leseindex nicht überholen, da sonst gültige, noch nicht ausgegeben Codefragmente überschrieben würden. Weiterhin gilt, dass der Leseindex erst dann erhöht wird, wenn genügend Daten in den Ringspeicher eingefügt wurden. In

[2] `s_new_pointer` = Leseindex + Bitbreite des Ausgabecodewortes

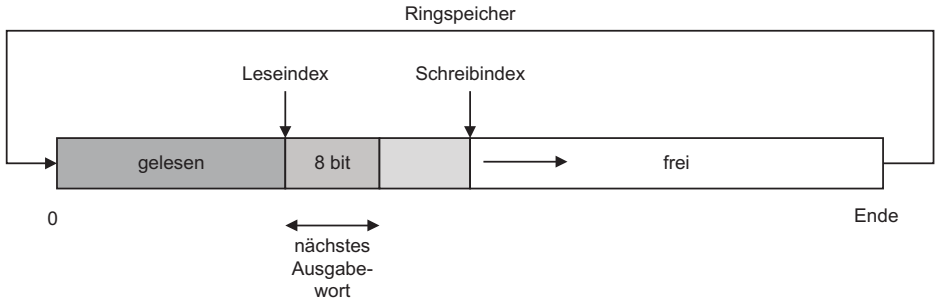


Abbildung 5.7.: Aufbau des Ringspeichers (Schema)

Abbildung 5.7 ist exemplarisch zu erkennen, dass die Bitbreite eines Ausgabewortes 8 bit beträgt und das Ende bzw. der Anfang vom Speicher noch nicht erreicht wurde.

```
1  -- this multiplexes the variable length input to the
       right buffer
2  --location if s_input_data_valid_pl1 = valid  then
3    for I in 0 to c_memory_size - 1 loop
4      for J in 0 to  MAX_WIDTH - 1 loop
5        if s_cond1 = invalid  then
6          -- normal condition
7          if (I < s_new_pointer and I >= s_wr_pointer
               and I-J = s_wr_pointer) then
8              s_memory(I) <= s_input_data_pl1(J);
9          end if;
10       else
11           -- new_pointer has surpassed the
                write_pointer
12         if (I < s_new_pointer) and (J-I = s_delta)
               then
13           s_memory(I) <= s_input_data_pl1(J);
14         end if;
15         if I >= s_wr_pointer and (I-J = s_wr_pointer)
               then
16           s_memory(I) <= s_input_data_pl1(J);
17         end if;
18       end if;
19     end loop;
20   end loop;
21 end if;
```

Listing 5.5: Ausschnitt aus dem VHDL-Ausgabemodul

5.3.4. Datendurchsatz und Latenz

Entwürfe mit bis zu vier voneinander unabhängigen SFALIC-HW-Kompressionsmodulen und 14-bit-Eingangswörtern sind möglich und konnten auf einem Virtex4 (Xilinx XC4VLX80) erfolgreich getestet werden. Die Taktrate betrug 125 Mhz. Werden für vier Kompressionsmodule die Einzeldatenraten in Summation betrachtet, ist eine Eingangsdatenrate von

$$125\ Msamples/s \cdot 4 = 500\ Msamples/s$$

erreichbar.

Aufgrund der sequentiellen Anordnung der Module Prädiktion, Codierung und Ausgabe, können die Verzögerungszeiten der einzelnen Module aufaddiert werden. Innerhalb des Codierers werden nach einem Takt die Codewortlängen $l_k(c_i)$ ermittelt und die Bitorientierung eingestellt. $\lceil log(\text{Bittiefe}) \rceil$ Takte sind zur Berechnung des Minimums der Zähler aus den Kontextmengen nötig, sodass die Latenz des Codierers

$$2 + \lceil log_2(\text{Bittiefe}) \rceil \text{ Takte} + FIFO \tag{5.2}$$

beträgt. Zur Berechnung der Residuen wird eine feste Anzahl von drei Takten benötigt, wobei ebenfalls die Latenz des FIFOs noch zu berücksichtigen ist. Das Modul zur Ausgabe der komprimierten Daten reiht die Codewörter variabler Länge wahlweise in 8-bit oder 16-bit Grenzen ein. In einem zyklischen Speicherbereich mit Lese- und Schreibindex wird das erste Codewort ausgegeben, sobald Lese- und Schreibindex sich entsprechend voneinander entfernt haben. Zusätzlich werden zwei Takte für die endgültige Bit- und Byteanordnung benötigt.

5.3.5. Skalierbarkeit

Nachdem ein Hardwareentwurf mit einem 14-bit-Kompressionsmodul erstellt wurde, sind 35 % der Fläche des Virtex4 verbraucht. Ein geringer Teil ($\approx$ 10 %) geht dabei an die Logik des Hardwarebetriebssystems verloren. Von dem 14-bit-Kompressionsmodul sind vier Instanzen parallel einsetzbar, bevor die Ressourcen des FPGAs verbraucht sind. Abbildung 5.8 ist ein Ausschnitt aus Tabelle 5.3 und veranschaulicht den linearen Zusammenhang zwischen der Bittiefe des

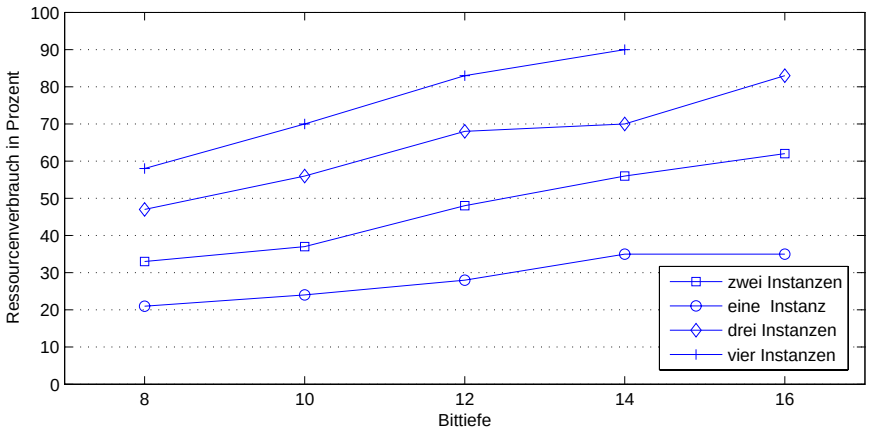


Abbildung 5.8.: Zusammenhang zwischen Bittiefe des Eingangsdatenwortes und dem Ressourcenverbrauch

Eingangsdatenwortes und dem Hardwareaufwand.

Eine ebenfalls lineare Abhängigkeit besteht zwischen der gewählten maximalen Codewortlänge und dem Ressourcenverbrauch. Die Messwerte der Tabelle 5.3 und der dazugehörigen Abbildung 5.8 wurden mit dem Parameter $l_{max} = 16$ für 8-bit-Grauwertbilder und $l_{max} = 24$ für Grauwertbilder größer als 8 bit ermittelt [3].

	SFALIC-HW Instanzen							
Bittiefe	1	2	3	4	5	6	7	8
8 bit	21 %	33 %	47 %	58 %	63 %	78 %	86 %	×
10 bit	24 %	37 %	56 %	70 %	80 %	88 %	×	×
12 bit	28 %	48 %	68 %	83 %	×	×	×	×
14 bit	35 %	56 %	70 %	90 %	×	×	×	×
16 bit	35 %	62 %	83 %	×	×	×	×	×

Tabelle 5.3.: Ressourcenverbrauch von SFALIC-HW für Virtex4 FPGA (4VLX80FF1148)

[3] × = keine Synthese möglich

6. Auswertung

Im Folgenden wird die Kompressionsrate des realisierten Verfahrens gemessenen und mit etablierten Kompressionsverfahren verglichen. Dabei spielt an dieser Stelle die Komplexität der Algorithmen lediglich eine untergeordnete Rolle. Die folgenden Kompressionsprogramme wurden getestet. Aus der Gruppe der sequentiellen prädikativen Kompressoren wurde der Standard JPEG-LS gewählt. Auf das Verfahren CALIC, das besser als JPEG-LS abschneidet, wurde verzichtet, da keine Implementierung erhältlich ist. Ein Vergleich von CALIC mit JPEG-LS und SFALIC bezüglich der Kompressionsrate und der Geschwindigkeit findet sich in der Veröffentlichung von Starosolski [37]. APT führt aus den Ergebnissen einer nichtlinearen Prädiktion eine Hauptkomponentenanalyse durch. Nach der Prädiktion werden die Prädiktionsfehler in einer Kombination aus Lauflängen und dynamischer Huffman Codierung komprimiert. Weiterhin wurden Gzip/deflate als Vertreter der auf einem Wörterbuch basierten Verfahren und S+P als Wavelettransformation auf Basis einer Zero-Tree-Codierung mit anschließender Huffman-Codierung getestet. Bzip2 implementiert eine Burrows-Wheeler-Transformation mit ebenfalls anschließender Huffman-Entropiecodierung. Damit die Messergebnisse nachvollziehbar sind, werden die Eingangsparameter der Kompressionsalgorithmen kurz wiedergegeben.

SFALIC-HW wurde mit einer MatLab-Implementierung simuliert. Die in diesem Test verwendeten Parameter für 8-bit-Bilder sind $l_{max} = 16$ und für 16-bit-Bilder $l_{max} = 24$. Die Testergebnisse basieren auf einer MatLab-Version, die funktional identisch mit der Hardwareimplementierung ist. Das in dieser Arbeit umgesetzte Kompressionsverfahren ist stark an SFALIC angelehnt. An einigen Punkten weicht das Verfahren jedoch von SFALIC ab.

SFALIC Von der Webseite `http://sun.iinf.polsl.gliwice.pl/~rstaros/`

`sfalic/index.html` kann eine in C geschriebene Referenzimplementierung vom Autor Starosolski [37] von SFALIC herunter geladen werden, die nahezu identische Kompressionsergebnisse liefert. Getestet wurde mit der Einstellung –pred 7 –maxclen 24 (–maxclen 16 für 8-bit-Grauwertbilder).

JPEG-LS basiert auf der Implementierung JPEG-LS Reference Encoder - V.1.00 von HP Labs und LOCO-I, die unter der Webseite `http://www.hpl.hp.com/loco` vom 23. Oktober 2007 zu finden ist. Die eingestellten Parameter sind `Ta=18 Tb=67 Tc=276 RESET=64 limit=47`.

Gzip/deflate basiert auf der Version Gzip distribution, Version 1.2.4. und wurde von der Internetseite `http://www.gzip.org/` kopiert. Die Parameter lauten `gzip.exe -c -9`.

ATP wurde von John Robinson entwickelt. Von dessen Internetseite `http://www.intuac.com/userport/john/apt/index.html` wurde eine Implementierung ATP 1.0 (2004) bezogen. ATP ist ein volladaptives Verfahren. Lediglich die Kompressionsgüte von 0 bis 100 (verlustlos) muss angegeben werden.

bzip2 kann von der Internetseite `http://www.bzip.org/` bezogen werden. Die eingesetzte Version bezieht sich auf die Nummer 1.0.4 vom 20 Dezember 2006. Die Parameter, mit denen gearbeitet wurde, sind `bzip2.exe -z -f -c -9`.

S+P in der Version 4.01 vom 2. Dezember 1994 kann unter `http://www.cipr.rpi.edu/research/SPIHT/spiht3.html` heruntergeladen werden.

Die Analyse der Leistungsfähigkeit von SFALIC-HW bezüglich der erreichbaren Kompressionsrate gliedert sich in drei Teile. Jeder Teil befasst sich mit einem anderen Schwerpunkt. Die Hauptaufgabe dieser Arbeit besteht in der verlustlosen Kompression von DLR-spezifischen Sensordaten. Die erste Klasse dieser Sensordaten entstammt dem MFC-Projekt. Sie ist durch einen hohen Dynamikumfang von bis zu 14 bit gekennzeichnet. Das heißt, dass im ersten Teil der Analyse die Kompressionsraten für 16-bit-Grauwertbilder der MFC-Kamera betrachtet werden. Der zweite Teil analysiert die selben Sensordaten mit maximal 8 bit Dynamikumfang, sowie Daten des ASTER-Satelliten. Der dritte

Name	bzip2	ATP	S+P	GZIP	LS	HW	SFALIC
1 A.4(f)	9,9011	12,245	11,572	14,715	11,854	11,911	11,987
2 A.4(g)	10,026	12,2	11,653	14,769	16,47	11,957	12,035
3 A.4(h)	10,415	12,594	12,095	14,804	16,498	12,446	12,516
4 A.4(i)	8,8924	9,6386	11,789	11,981	16,356	11,863	11,95
5 A.4(j)	10,236	12,179	11,76	14,545	16,423	11,971	12,049
Durch-schnitt	9.8941	11.7714	11.7736	14.1631	15.5202	12.0297	12.1074

Tabelle 6.1.: Bitraten (bpp) für DLR-16-bit A.4
(LS = JPEG-LS, HW = SFALIC-HW)

und letzte Teil vergleicht die Leistung von SFALIC-HW auf Basis der Waterloo-Bildersammlung und dient einem globalen Überblick, da die meisten Kompressionsverfahren für eine breite Anwendungspalette entwickelt wurden. Die Gruppe

Name	bzip2	ATP	S+P	Gzip	LS	HW	SFALIC
1 A.4(a)	6,1844	5,997	6,028	7,0163	5,691	5.8988	6,0293
2 A.4(b)	7,0311	6,8262	6,603	7,2412	6,56	6.8088	6,948
3 A.4(c)	6,1711	5,6988	5,4723	7,2024	5,312	5.5306	5,6463
4 A.4(d)	7,2324	7,0836	6,7679	7,3474	6,655	6.9208	7,0449
5 A.4(e)	7,2937	7,1023	6,8083	7,3973	6,808	7.0223	7,1493
6 A.4(f)	4,2673	3,9105	4,1713	5,9112	3,693	3.9043	3,9817
7 A.4(g)	4,3932	3,8737	4,1945	6,155	3,677	3.9591	4,0387
8 A.4(h)	4,7465	4,2283	4,462	6,3484	4,039	4.4108	4,499
9 A.4(i)	4,3219	3,7843	4,1802	5,7678	3,597	3.8467	3,9294
10 A.4(j)	4,3403	3,8841	3,9464	5,9537	3,628	4.004	4,0817
Durch-schnitt	5.5982	5.2389	5.2634	6.6341	4.9660	5.2306	5.3348

Tabelle 6.2.: Bitraten (bpp) für DLR-8-bit A.4
(LS = JPEG-LS, HW = SFALIC-HW)

mit den relevanten und gleichzeitig interessantesten Ergebnissen sind die 16-bit-Grauwertbilder aus der Menge A.4. Das Programm bzip2 liefert erstaunlich gute Kompressionsergebnisse. Die durchschnittliche Kompressionsrate von bzip2 ist um bis zu $36,1$ % besser als die von der am schlechtesten abschneidenden JPEG-LS-Implementierung. Gegenüber SFALIC-HW ist JPEG-LS um $22,5$ % schlechter. Die schlechte durchschnittliche Kompressionsleistung von JPEG-LS ist zum Teil darauf zurückzuführen, dass für die MFC-Bilder A.4(g),

A.4(h), A.4(i) und A.4(j) eine unerwünschte Datenexpansion ausgeführt wird. Gegenüber von Gzip/deflate ist SFALIC-HW im Durchschnitt um 15,1 % besser. Gute Ergebnisse werden auch von ATP und S+P geliefert, die beide um drei Prozent besser sind als SFALIC-HW. In der zweiten Kategorie der 8-bit-

Name	bzip2	ATP	S+P	GZIP	LS	HW	SFALIC
A.5(a)	2,5263	3,2597	4,3649	2,6884	4,129	4,4065	4,5472
A.5(b)	5,1303	4,9823	4,9318	5,9196	4,314	4,6556	4,7646
A.5(c)	5,1328	6,7306	6,9156	5,2763	6,422	6,6378	6,7427
A.5(d)	5,5989	5,0616	4,9589	6,6826	4,712	4,9474	5,0561
A.5(e)	6,6077	6,395	6,1774	7,2533	6,036	6,1783	6,2716
A.5(f)	4,476	5,2662	6,382	4,7673	5,101	5,7379	5,821
A.5(g)	5,3388	4,7457	4,7115	6,9888	4,489	4,8129	4,9298
A.5(h)	3,4588	5,7115	3,9937	3,8326	6,049	6,3005	6,4625
A.5(i)	6,125	5,2801	4,9669	7,0547	4,733	5,3206	5,3596
A.5(j)	5,3368	4,7787	4,6248	6,1809	4,25	4,7442	4,8344
A.5(k)	5,1235	4,1676	4,0271	6,6813	4,005	4,2853	4,4351
A.5(l)	5,3073	4,5047	4,3872	6,7941	4,244	4,5835	4,6691
Durch-schnitt	5.0135	5.0737	5.0368	5.8433	4.8737	5.2176	5.3245

Tabelle 6.3.: Messwerte für Waterloo-8-bit A.5

Grauwertbilder A.4 arbeitet JPEG-LS bedeutend besser und liefert im Mittel die besten Resultate. Das standardisierte Verfahren ist um fünf Prozent besser als SFALIC-HW. Im Vergleich der Verfahren mit den geringsten Anforderungen an die Ressourcen kann sich SFALIC-HW gegenüber dem auf einem Wörterbuch basierenden Programm Gzip/deflate durchsetzten. SFALIC-HW ist im Durchschnitt um 21 % besser als Gzip/deflate. Gegenüber der ursprünglichen Version von SFALIC ist die in dieser Arbeit vorgestellte Hardwareimplementierung SFALIC-HW im Durchschnitt knapp zwei Prozent besser. Diese Beobachtung kann auch für die 16-bit-Grauwertbilder A.4(f), A.4(g), A.4(h), A.4(i) und A.4(j) und der Waterloo-Testserie A.5 gemacht werden. SFALIC-HW, angewendet auf die Grauwertbilder der Waterloo-Testserie, liefert ebenfalls gute Kompressionsergebnisse. Erneut ist JPEG-LS allen Vergleichsprogrammen überlegen und liefert ein um 6,5 % besseres Ergebnis als SFALIC-HW. Am geringsten ist der Abstand von SFALIC-HW und Gzip/deflate in der Kategorie Waterloo. Nichtsdestotrotz ist SFALIC-HW um zehn Prozent besser als das Verfahren Gzip/deflate.

7. Zusammenfassung und Ausblick

In dieser Arbeit wurden für Sensordaten von Matrix- und Zeilenkameras verlustfreie Kompressionsverfahren bezüglich ihrer Kompressionsrate und einer möglichen Umsetzung in rekonfigurierbarer Hardware untersucht. Dazu wurden geeignete Verfahren zur Dekorrelation vorgestellt und bezüglich ihrer Komplexität analysiert. Es hat sich gezeigt, dass eine eindimensionale Verarbeitung zweidimensionaler Sensordaten mittels einfacher linearer Prädiktion gute Voraussetzungen für eine statistische Codierung bietet. Anschließend wurden die Eigenschaften verschiedener Codierungsmöglichkeiten miteinander verglichen. Softwaresimulationen wurden durchgeführt, um einen geeigneten Codierer mit minimalem Ressourcenverbrauch zu ermitteln, mit dem Ergebnis, das ein dynamischer Golomb-Rice-Codierer die besten Voraussetzungen für eine ressourceneffiziente Hardwareumsetzung bietet.

Das zur Umsetzung gewählte kontextadaptive Verfahren SFALIC-HW ist in der Lage, die Sensordaten mit einer Eingangsdatenrate von einem Pixel pro Takt zu verarbeiten. Der Entwurf wurde auf einem FPGA mit einem Takt von 125 Mhz erfolgreich getestet. Es konnte gezeigt werden, dass der Ressourcenverbrauch von SFALIC-HW sehr gering ist, sodass mehrere voneinander unabhängige Instanzen auf handelsüblichen FPGAs eingesetzt werden können. Dennoch liefert SFALIC-HW gute Kompressionsergebnisse, die im wesentlichen nur um sechs bis zehn Prozent schlechter sind als der Standard JPEG-LS zur verlustfreien Kompression von 8-bit-Grauwertbildern. Desweiteren konnte dargestellt werden, dass im besonderen Fall von Fernerkundungsdaten mit einer hohen Auflösung und 16-bit-Datenwortbreite SFALIC-HW dem Standard JPEG-LS sowohl beim Ressourcenverbrauch als auch bei der Kompressionsleistung um bis zu 22 % überlegen ist.

Hinsichtlich der Kompressionsrate von Aufnahmen mit einer Bittiefe von mehr als 8 bit erzielt die Burrows-Wheeler-Transformation hervorragende Werte, deren Potential für Hardwareimplementierungen noch nicht ausreichend erforscht zu sein scheint. Das schlechte Abschneiden von JPEG-LS im Falle der 16-bit-Grauwertbilder bietet ebenfalls einen Anreiz für weitere Nachforschungen. Insbesondere die Erkennung von homogenen Flächen und der daraus resultierende Eintritt in den Lauflängenmodus, der nach Ansicht des Autors für die Datenexpansion der 16-bit-Grauwertbilder A.4(g), A.4(h), A.4(i) und A.4(j) verantwortlich ist, bietet Raum für weitere Überlegungen.

Literaturverzeichnis

[1] Bedi, Satish ; Edirisinghe, Eran A. ; Grecos, Christos: Improvements to the JPEG-LS prediction scheme. In: *Image Vision Comput.* 22 (2004), Nr. 1, S. 9–14

[2] Burrows, M. ; Wheeler, D. J.: A block-sorting lossless data compression algorithm. Version: 1994. `citeseer.ist.psu.edu/76182.html`. 1994 (124). – Forschungsbericht

[3] `http://www.dlr.de`

[4] Fenwick, Peter M.: A New Data Structure for Cumulative Frequency Tables. In: *Software - Practice and Experience* 24 (1994), Nr. 3, 327-336,. `citeseer.ist.psu.edu/fenwick94new.html`

[5] Gallager, R.G. ; Voorhis, D.C. van: Optimal source codes for geometrically distributed integer alphabets. In: *IEEE Transactions on Information Theory* 21 (1975), Nr. 3, S. 228–230

[6] Golomb, S.W.: Run-Length Encodings. In: *IEEE Transactions on Information Theory* Bd. 12, 1966, S. 399–401

[7] Hoggar, S. G.: *Mathematics of Digital Images: Creation, Compression, Restoration, Recognition.* New York, NY, USA : Cambridge University Press, 2006. – ISBN 0521780292

[8] Howard, Paul G. ; Vitter, Jeffrey S.: Practical Implementations of Arithmetic Coding. Providence, RI, USA : Brown University, 1991. – Forschungsbericht

[9] Howard, Paul G. ; Vitter., Jeffrey S.: Fast and Efficient Lossless Image Compression. Version: 1994. `citeseer.ist.psu.edu/howard93fast.html`. 1994 (Technical report DUKE–TR–1994–10). – Forschungsbericht

[10] ISO/IEC: *ISO/IEC 15444-1:2004 information technology-JPEG2000-image coding system-part 1: core coding system.* 2004

[11] ITU: *ISO/IEC 10918-1 : 1993(E) CCIT Recommendation T.81.* `http://www.w3.org/Graphics/JPEG/itu-t81.pdf`. Version: 1993

[12] Jamro, Ernest ; Wielgosz, Maciej ; Wiatr, Kazimierz: FPGA Implementation of the Dynamic Huffman Encoder. In: *Programmable Devices and embedded Systems* (2006). `http://agh.edu.pl`

[13] Kiely, A.: Selecting the Golomb Parameter in Rice Coding. In: *IPN Progress Report* 42 (2004), Nr. 159, S. 1–8

[14] Knuth, Donald E.: Dynamic Huffman coding. In: *J. Algorithms* 6 (1985), Nr. 2, S. 163–180. `http://dx.doi.org/http://dx.doi.org/10.1016/0196-6774(85)90036-7`. – DOI http://dx.doi.org/10.1016/0196–6774(85)90036–7. – ISSN 0196–6774

[15] Kominek, John: *The Waterloo BragZone and Fractals Repository*. `http://links.uwaterloo.ca/bragzone.base.html`. – [Online: Stand 2007-08-08T07:10:13Z]

[16] Krutz, David: *Ein Betriebssystem für konfigurierbare Hardware*. `http://edoc.hu-berlin.de/docviews/abstract.php?id=27784`. – [Online: Stand 2007-08-08T07:10:13Z]

[17] Lee, Hunjoong ; Jeong, Hayoung ; Ham, Donghyeon ; Lee, Yong-Surk: A Novel Architecture for High Performance CABAC Encoder. In: *The 21st International Technical Conference on Circuits/Systems, Computers and Communications*, Processor Laboratory, Department of Electrical and Electronic Engineering, Yonsei University, 2006. – ISBN 974–94418–9–3, S. 753–758

[18] Marks, K. M.: A JBIG-ABIC compression engine for digital document processing. In: *IBM J. Res. Dev.* 42 (1998), Nr. 6, S. 753–758. – ISSN 0018–8646

[19] Meffert, Beate ; Hochmuth, Olaf: *Werkzeuge der Signalverarbeitung*. Pearson Studium, 2004

[20] Merhav, N. ; Seroussi, G. ; Weinberger, M.: *Optimal prefix codes for sources with two-sided geometric distributions*. `citeseer.ist.psu.edu/article/merhav97optimal.html`. Version: 2000

[21] Moffat, Alistair ; Neal, Radford M. ; Witten, Ian H.: Arithmetic coding revisited. In: *ACM Trans. Inf. Syst.* 16 (1998), Nr. 3, S. 256–294. `http://dx.doi.org/http://doi.acm.org/10.1145/290159.290162`. – DOI http://doi.acm.org/10.1145/290159.290162. – ISSN 1046–8188

[22] Mrak, Marta ; Marpe, Detlev ; Wiegand, Thomas: A context modeling algorithm and its application in video compression. In: *ICIP (3)*, 2003, S. 845–848

[23] Pennebaker, William B. ; Mitchell, Joan L. ; Jr., Glen G. L. ; Arps, Ronald: An Overview of the Basic Principles of the Q-Coder Adaptive Binary Arithmetic Coder. In: *IBM Journal of Research and Development* 32 (1988), Nr. 6, S. 717–726

[24] Rice, R. F.: Lossless Coding Standards for Space Data Systems. In: *JPL TRS 1992+*, 1996

[25] Ritter, Jörg: *Wavelet based image compression using FPGAs*. 2002. – Dissertation

[26] Said, Amir: Comparative Analysis of Arithmetic Coding Computational Complexity. In: *DCC '04: Proceedings of the Conference on Data Compression*. Washington, DC, USA : IEEE Computer Society, 2004. – ISBN 0–7695–2082–0, S. 562

[27] Said, Amir: Introduction to Arithmetic Coding - Theory and Practice. Version: 2004. `http://www.hpl.hp.com/techreports/2004/HPL-2004-76.pdf`. Media Technologies Laboratory : HP Laboratories Palo Alto, 2004. – Forschungsbericht

[28] Said, Amir: On the Determination of Optimal Parameterized Prefix Codes for Adaptive Entropy Coding. Version: 2006. `http://www.hpl.hp.com/techreports/2006/HPL-2006-74.pdf`. Media Technologies Laboratory : HP Laboratories Palo Alto, 2006. – Forschungsbericht

[29] Said, Amir ; Pearlman, William A.: A New Fast and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. In: *IEEE Transactions on Circuits and Systems for Video Technology* 6 (1996), 243–250. `citeseer.ist.psu.edu/said96new.html`

[30] Salomon, David: *Data Compression: The Complete Reference, 3rd Edition*. Springer, 2004 `http://www.davidsalomon.name/DC3advertis/DComp3Ad.html`. – ISBN 0–387–40697–2

[31] Sayood, Khalid: *Lossless Compression Handbook (Communications, Networking and Multimedia) (Communications, Networking and Multimedia)*. Academic Press, 2003. – ISBN 0126208611

[32] Sayood, Khalid: *Introduction to Data Compression*. Third. San Francisco, CA, USA : Morgan Kaufmann Publishers, 2006

[33] Shannon, Claude E.: A mathematical theory of communication. In: *Mobile Computing and Communications Review (Reprint)* 5 (2001), Nr. 1, S. 3–55

[34] Shapiro, Jerome M.: Embedded image coding using zerotrees of wavelet coefficients. (2001), S. 124–141. ISBN 1–55860–651–3

[35] Shojania, Hassan ; Sudharsanan, Subramania: A VLSI Architecture for High Performance CABAC Encoding. Kingston, ON K7L 3N6, Canada : Department of Electrical and Computer Engineering Queen's University„ 2005. – Forschungsbericht

[36] Space Data Systems, Consultative C.: CCSDS 122.0-B-1, Issue 1. In: *RECOMMENDED STANDARD FOR IMAGE DATA COMPRESSION* (2005)

[37] Starosolski, Roman: Simple fast and adaptive lossless image compression algorithm. In: *Softw. Pract. Exper.* 37 (2007), Nr. 1, S. 65–91. `http://dx.doi.org/http://dx.doi.org/10.1002/spe.v37:1`. – DOI http://dx.doi.org/10.1002/spe.v37:1. – ISSN 0038–0644

[38] Starosolski, Roman ; Skarbek, Wladyslaw: Modified Golomb-Rice Codes for Lossless Compression of Medical Images. In: *Proc. of International Conference on E-health in Common Europe*, 2003. – ISSN 0038–0644, S. 423–437

[39] Stefo, Riad ; Nunez, Jose L. ; Feregrino, Claudia ; Mahapatra, Sudipta ; Jones, Simon: FPGA-Based Modelling Unit for High Speed Lossless Arithmetic Coding. In: *FPL '01: Proceedings of the 11th International Conference on Field-Programmable Logic and Applications*. London, UK : Springer-Verlag, 2001. – ISBN 3–540–42499–7, S. 643–647

[40] Strutz, Tilo: *Bilddatenkompression : Grundlagen, Codierung, Wavelets, JPEG, MPEG, H.264*. 3., aktualisierte und erweiterte Auflage. Wiesbaden : Vieweg, 2005

[41] Sweldens, W.: The lifting scheme: A construction of second generation wavelets. In: *SIAM J. Math. Anal.* 29 (1997), Nr. 2, S. 511–546

[42] Vitter, J. S.: Dynamic Huffman Coding. In: *ACM Trans. Math. Softw.* 15 (1989), Nr. 2, 158–167. `citeseer.ist.psu.edu/vitter89dynamic.html`

[43] Weinberger, M. J. ; Seroussi, G. ; Sapiro, G.: LOCO-I: a low complexity, context-based, lossless image compression algorithm. In: *DCC '96: Proceedings of the Conference on Data Compression*. Washington, DC, USA : IEEE Computer Society, 1996. – ISBN 0–8186–7358–3, S. 140

[44] Witten, Ian H. ; Moffat, Alistair ; Bell, Timothy C.: *Managing gigabytes (2nd ed.): compressing and indexing documents and images*. San Francisco, CA, USA : Morgan Kaufmann Publishers Inc., 1999. – ISBN 1–55860–570–3

A. Anhang

A.1. Schnittstellen des Hardwarebetriebssystems

In Codeausschnitt A.1 ist die vollständige Schnittstellenbeschreibung einer VHDL-Komponente mit Hardwarebetriebssystemunterstützung dargestellt. Das Taktsignal `input_clock` wird durch das Modul nach außen über `output_clock` geleitet. Die Kommunikation mit dem Betriebssystem beginnt, indem das Signal `input_request` den Wert `valid` annimmt und damit der Hardware-Komponente der Wunsch nach Kommunikation im Master-Slave-Verfahren angezeigt wird, wobei exemplarisch die Komponente die passive Position einnimmt. Die Bereitschaft des Slaves zum Empfang von Daten wird dem Master mittels einem `valid`-Wert des `input_confirm`-Signals angezeigt.

Erst mit dem Zurücksetzen des `input_confirm`- oder des `input_request`-Signals wird die Übertragung beendet. `Input_busy` dient einer kurzfristigen Unterbrechung. `Input_data` und `Input_data_valid` sind die üblichen Signale zum Transfer der Daten von Master zu Slave. Alle Signale mit dem Präfix `output` können dazu verwendet werden, die Hardwarekomponente in eine aktive Rolle zur Datenübertragung zu versetzen und sind in der Funktionalität identisch zu den `input`-Signalen.

```
1  entity sfalic_predictor is
2    generic (
3      MAX_SIZEX : integer;
4      WIDTH     : integer
5      );
6    port (
7      sizeX        : in  std_logic_vector(15 downto 0);
8      sizeY        : in  std_logic_vector(15 downto 0);
9
10     input_clock         : in  std_logic;
11     input_request       : in  valid_type;
12     input_confirm       : out valid_type;
13     input_frame_number  : in  PIPE_FRAMENUMBER_TYPE;
14     input_data_valid    : in  valid_type;
15     input_busy          : out valid_type;
16     input_data          : in  std_logic_vector(WIDTH-1
          downto 0);
17
```

```
18      output_clock          : out std_logic;
19      output_request        : out valid_type;
20      output_confirm        : in  valid_type;
21      output_frame_number   : out PIPE_FRAMENUMBER_TYPE;
22      output_data_valid     : out valid_type;
23      output_busy           : in  valid_type;
24      output_data           : out std_logic_vector(
           OUTPUT_WIDTH -1 downto 0)
25      );
26  end;
```

Listing A.1: Schnittstellen des Hardwarebetriebssystems

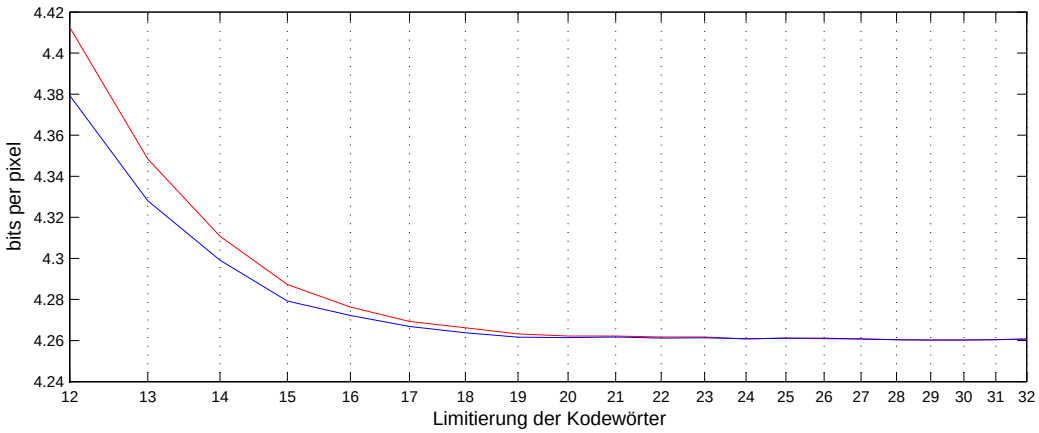


Abbildung A.1.: Vergrößerter Ausschnitt der Kompressionsraten eines 8-bit-Grauwertbildes (A.5(k)) mit vollständiger und unvollständiger modifizierten Golomb-Rice-Codetabelle als Funktion des Codeparameters k

A.2. Prädiktoren

Modified-Median-Edge-Detektor

In Erweiterung des JPEG-LS Median-Edge-Detektors werden zusätzlich diagonale Kanten erkannt. (siehe hierzu [1])

Nummer	Wert
1	v=0
2	v=a
3	v=b
4	v=c
5	v = a+b-c
6	v = $\frac{a+(b-c)}{2}$
7	v = $\frac{b+(a-c)}{2}$
8	v = $\frac{(a+b)}{2}$

Tabelle A.1.: Lossless-JPEG-Prädiktoren

Name	Dimension	MED	linear
(1) washsat	[512 512]	4.5863	4.4065
(2) kameramann	[256 256]	4.5916	4.6556
(3) mountain	[640 480]	6.7779	6.6378
(4) goldhill	[512 512]	4.8698	4.9474
(5) mandrill	[464 352]	6.2247	6.1783
(6) library	[512 512]	5.5538	5.7379
(7) peppers	[621 498]	4.8872	4.8129
(8) frog	[512 512]	6.5308	6.3005
(9) barb	[512 512]	5.1516	5.3206
(10) boat	[512 512]	4.5735	4.7442
(11) zelda	[512 512]	4.2765	4.2853
(12) lena	[512 512]	4.5147	4.5835
Durchschnitt		5.2115	5.2176

Tabelle A.2.: JPEG-LS-Prädiktor im Vergleich zum Prädiktor $v = \frac{(A+B)}{2}$

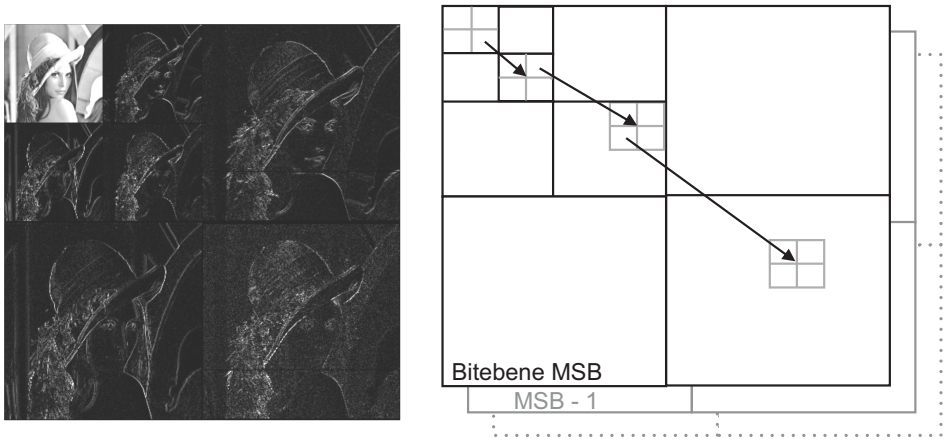
Lossless-JPEG-Prädiktoren

A.3. Testserien

A.3.1. Bildinformation

A.4(a) ASTER-Aufnahme nordöstlich von Berlin, Liebnitzsee, Bundesautobahn A11; `http://photojournal.jpl.nasa.gov/catalog/PIA01914`

A.4(b) ASTER-Aufnahmen vom Terra NASA Satelliten vom 15.010.2005; Size: 12.1 by 15.9 kilometers; Location: 52.5 degrees North latitude, 13.3 degrees East longitude `http://photojournal.jpl.nasa.gov/catalog/PIA01914`



(a) Lena (A.5(l)) mit dem Haar-Wavelet transformiert (8 bit)

(b) zero tree Kodierung (Schema)

Abbildung A.2.: Wavelettransformation und Codierung in einer Baumstruktur

A.4(c) ASTER-Aufnahmen vom Terra NASA Satelliten vom 27.06.2007; Size: 15 by 15 kilometers; Location: 38.9 degrees North latitude, 120 degrees West longitude `http://photojournal.jpl.nasa.gov/catalog/PIA09698`

A.4(d), A.4(e) ASTER-Aufnahme von Berlin, Flughafen Tempelhof, Südkreuz `http://photojournal.jpl.nasa.gov/catalog/PIA01914`

A.4(e), A.4(e) ASTER-Aufnahme von Berlin, Flughafen Tempelhof, Südkreuz in einer vergrößerten Auflösung `http://photojournal.jpl.nasa.gov/catalog/PIA01914`.

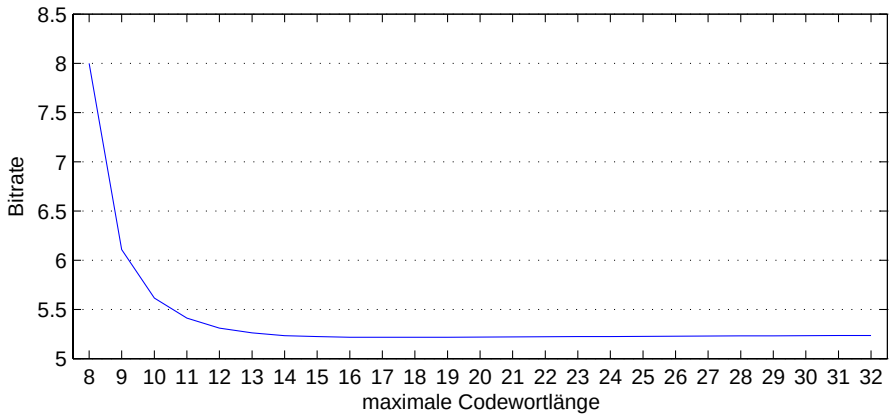
A.4(f) MFC-Aufnahme, von Berlin-Adlershof, S-Bahn, Rudower Chaussee mit unveränderter Auflösung.

A.4(g) MFC-Aufnahme, von Berlin-Adlershof, BESY, Landwehrkanal, Bundesautobahn A113 mit verringerter Auflösung.

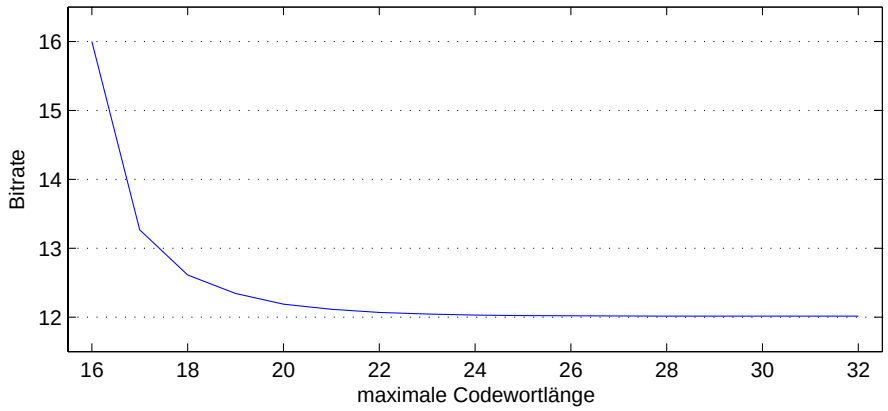
A.4(h) MFC-Aufnahme, von Berlin-Adlershof, Joachimsthal mit verringerter Auflösung.

A.4(i) MFC-Aufnahme, von Berlin-Adlershof, BESY, Landwehrkanal, mit starkem Schatten in der linken unteren Bildhälfte mit verringerter Auflösung.

A.4(j) MFC-Aufnahme, von Berlin-Adlershof, Dach des BESY-Gebäudes mit verringerter Auflösung.

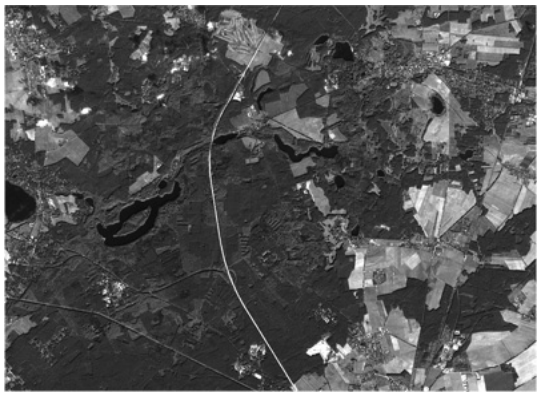


(a) durchschnittliche Bitraten für 8-bit-Grauwertbilder A.5 in Abhängigkeit zu der maximalen Codewortlänge im Fall einer Golomb-Rice-Codierung

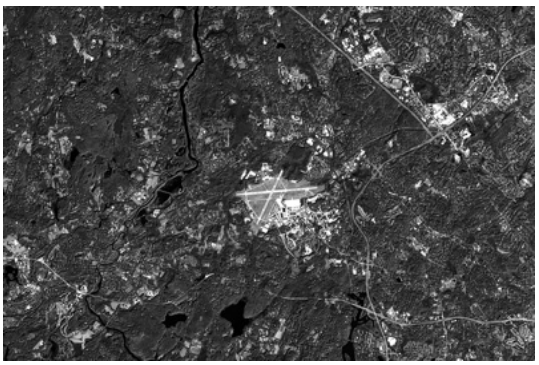


(b) durchschnittliche Bitraten für 16-bit-Grauwertbilder A.4 in Abhängigkeit zu der maximalen Codewortlänge im Fall einer Golomb-Rice-Codierung

Abbildung A.3.:



(a) ASTER Barnim (980 × 712)



(b) ASTER Lexington (1200 × 800)

(c) ASTER South Lake Tahoe(1000 × 1000)

(d) ASTER Berlin (256 × 256)

(e) ASTER Berlin (512 × 512)

(f) MFC Adlershof (8004 × 3616)

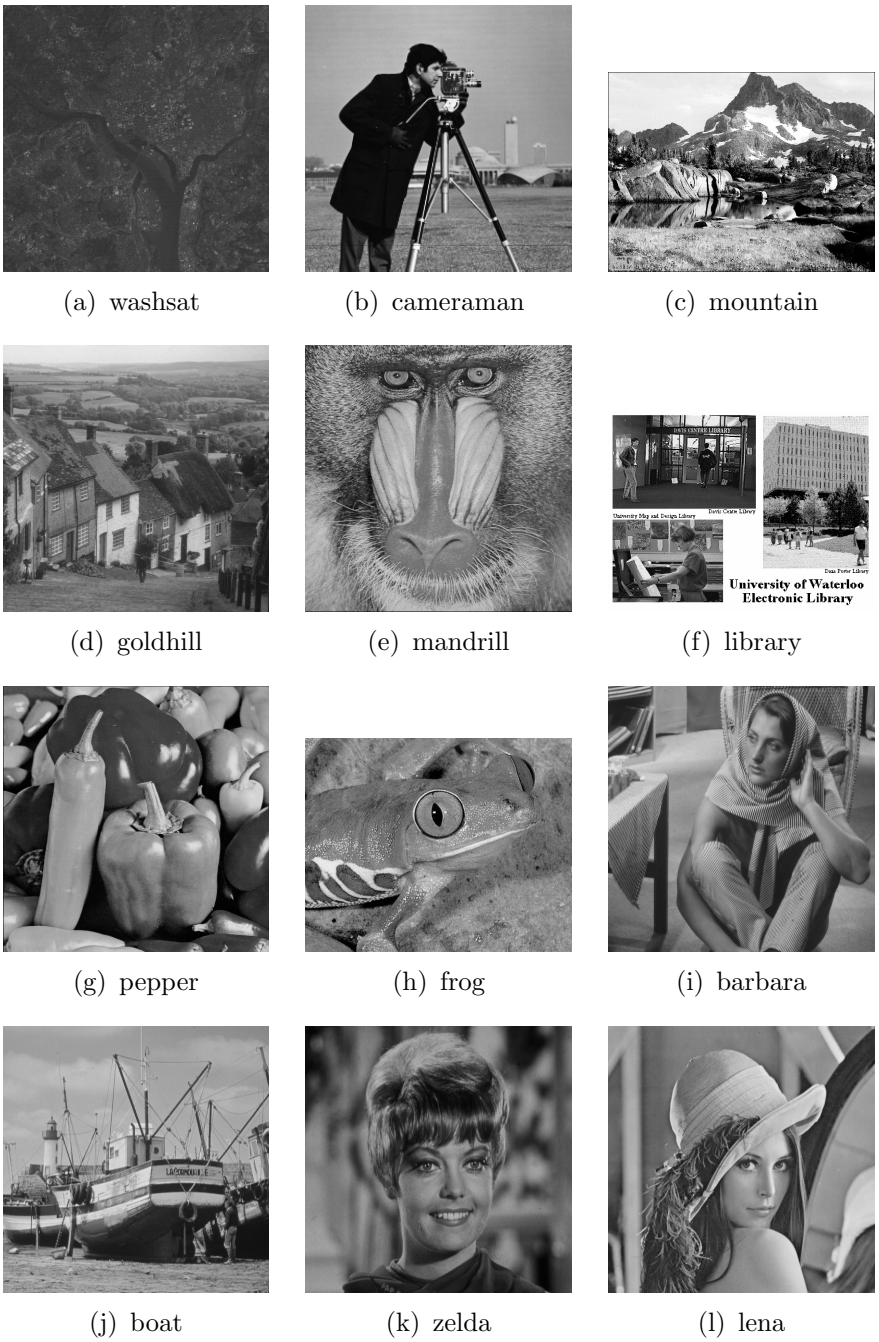
(g) MFC Landwehrkanal (5028 × 2464)

(h) MFC Siedlung (1844 × 1216)

(i) MFC Landwehrkanal (4364 × 2804)

(j) MFC Dach von BESY (1160 × 640)

Abbildung A.4.: DRL-Testserie



(a) washsat (b) cameraman (c) mountain

(d) goldhill (e) mandrill (f) library

(g) pepper (h) frog (i) barbara

(j) boat (k) zelda (l) lena

Abbildung A.5.: Waterloo-Testserie (8 bit)[15]

Selbständigkeitserklärung

Ich erkläre, diese Diplomarbeit selbständig und nur unter Verwendung der angegebenen Literatur und Hilfsmittel angefertigt zu haben.

Berlin, den 23. Oktober 2007

Ich bin damit einverstanden, dass ein Exemplar dieser Arbeit in der Bibliothek des Instituts für Informatik der Humboldt-Universität zu Berlin ausgestellt wird.

Berlin, den 23. Oktober 2007