Technische Universität München
Fakultät für Mathematik

Erweiterung der kontextuellen Kompressionsmethoden mit Hilfe von Delaunay Triangulierungen

Diplomarbeit von Marie Hinnenthal

Aufgabensteller : Prof. Dr. Rupert Lasser

Betreuer : Dr. Laurent Demaret

Abgabetermin : 11. November 2010

Ich erkläre hiermit, dass ich die Diplomarbeit selbständig und nur mit den angegebenen Hilfsmitteln angefertigt habe.

_____

(Ort, Datum Unterschrift)

Erweiterung der kontextuellen Kompressionsmethoden mit Hilfe von Delaunay Triangulierungen

Marie Hinnenthal

9. November 2010

Inhaltsverzeichnis

1 Einführung und Motivation **4**

1.1 Triangulierungen: Definitionen und Theorie 6

1.1.1 Grundlagen der Triangulierungen 7

1.1.2 Delaunay Triangulierung 10

1.1.3 Isotrope und anisotrope Approximationsmethoden 11

1.1.4 Abstrakte Formulierung des Approximationsproblems . . 13

1.1.5 Beispiele der Approximationsmethoden 14

1.2 Arithmetische Kodierung 22

1.2.1 Grundlagen der Kodierungstheorie 22

1.2.2 Arithmetische Kodierung 24

1.2.3 Adaptive arithmetische Kodierung 28

1.2.4 Kontextuelle adaptive arithmetische Kodierung 29

1.3 Zusammenfassung 32

2 Kontextuelles Kompressionsverfahren **33**

2.1 Bildapproximation 33

2.1.1 Bilddarstellung 34

2.1.2 Adaptive Thinning Algorithmus 34

2.1.3 Lineare Splines über der Delaunay Triangulierung 36

2.1.4 Signifikanz-Maß 37

2.1.5 Äquivalentes Signifikanz-Maß 39

2.1.6 Minimierung des mittleren quadratischen Fehlers 39

2.2 Kontextuelle Bildkompression 40

2.2.1 Reduzierte Bilddarstellung 40

2.2.2 Kontexte der Pixel Positionen 41

2.2.3 Kontexte der quantisierten Grauwerte 44

2.3 Rekonstruktion des Bildes beim Dekodierer 47

2.4 Zusammenfassung 47

3 Untersuchungen neuer Ansätze **48**

3.1 Motivation und Eigenschaften 48

3.2 Kontexte der Pixel Positionen 51

3.2.1 Theoretische Kosten der Pixel Positionen 51
3.2.2 Neue Ansätze / Methodik / Statistische Experimente . . . 53
3.3 Kontexte der Grauwerte 65
3.3.1 Theoretische Kosten der Grauwerte 65
3.3.2 Neue Ansätze / Methodik / Statistische Experimente . . . 67
3.4 Statistische Auswertungen / Ergebnisse 72
3.4.1 Pixel Positionen 73
3.4.2 Grauwerte 78

4 Kompressionsverfahren CAT* 83
4.1 Beschreibung des Verfahrens 83
4.2 Numerische Ergebnisse 85

5 Schlussbemerkung 96

Abbildungsverzeichnis 97

Tabellenverzeichnis 99

Literaturverzeichnis 101

Kapitel 1

Einführung und Motivation

Seit der Einführung der ersten Personal Computer vor etwa 30 Jahren erleben wir etwas, das von Wissenschaftlern, Journalisten oder Politikern als „digitale Revolution“ bezeichnet wird. Kern dieser Entwicklung ist die Digitalisierung von Daten, die wir mit Hilfe der Computertechnologie speichern und dank des Internets weltweit verbreiten können. Die Begrenzung der Kapazitäten in Übertragungskanälen und Speichermedien erfordert dabei die Entwicklung geeigneter Kompressionsverfahren. Insbesondere für die Speicherung digitaler Bilder und Videos ist eine hohe Speicherkapazität erforderlich. Es besteht somit ein großer Bedarf an Kompressionstechniken, die den Besonderheiten von Bildern angepasst sind.
In dieser Arbeit befassen wir uns mit der Erweiterung eines in den letzten Jahren von Demaret et al. [4] entwickelten, verlustbehafteten Bildkompressionsverfahrens, *Contextual Adaptive Thinning* (CAT). Dieses Verfahren kombiniert einen effizienten Algorithmus für die Identifizierung relevanter Bildinformationen, den Adaptive Thinning Algorithmus, mit einer kontextuellen adaptiven arithmetischen Kodierung der reduzierten Bilddaten. Bei diesem Kompressionsverfahren wird das Bild mit Hilfe einer linearen Splinefunktion über einer speziellen Triangulierung, der Delaunay Triangulierung einer adaptiv ausgewählten Pixelmenge, approximiert, sodass die Strukturen und die Geometrie des Bildes erfasst werden können [7]. Die Kontexte, die für die kontextuelle adaptive arithmetische Kodierung des bestehenden Verfahrens CAT verwendet werden, sind jedoch teilweise ad hoc gewählt. Der Schwerpunkt dieser Arbeit liegt in der Analyse neuer Kontexte und der Entwicklung effektiverer Kompressionsmethoden für die kontextuelle Kompression der relevanten Bildinformationen, die durch den Adaptive Thinning Algorithmus identifiziert wurden.

Die Anwendung geeigneter Kompressionsverfahren sind, neben den Anwendungen in digitalen Kameras, Mobiltelefonen und Navigationsgeräten, auch in der Medizin von besonderer Wichtigkeit. Insbesondere vor dem Hintergrund zunehmender Vernetzung. Das moderne Internet-/Intranet-orientierte RIS (Radiologie-

Informations-System) ist heute in fast jedem Krankenhaus eine Ausrüstung für die Übertragung und Bearbeitung medizinischer Bilder [2]. Diese stammen aus physikalisch-technischen Messprozessen und bilden gewaltige Datenmengen. Bei einer Computertomographie oder Magnetresonanztomographie, kurz CT oder MRT, benötigt man für ein Bild $512 \times 512 \times 16 = 0.5$ MByte Speicherplatz [3]. Bei einer Standard-Serie von 32 Bildern ergibt dies einen Speicherplatzbedarf von 16 MByte. Beim Röntgen, etwa einer Mammographie, werden für eine Aufnahme sogar 40 MByte benötigt. Bei einer Untersuchung, bei der 4 Aufnahmen gemacht werden, ergibt das 160 MByte. Dies verursacht eine unzumutbare lange Dauer bei der Übertragung unkomprimierter Bilder. Die Übertragungszeit beträgt dabei für 160 MByte, bei einer ISDN-Leitung, zwischen zwei und drei Stunden. Des Weiteren bestehen in Krankenhäusern gesetzliche Vorschriften zur Archivierung der Bilddaten, beispielsweise im *Picture Archiving and Communication System* (PACS). Komprimierte Datensätze können diesen Problemen entgegenwirken. Sie verbrauchen weniger Speicherplatz im PACS und werden darüber hinaus, bei gleicher Bandbreite, schneller übertragen als die Originaldaten.
Physikalisch-technische Messprozesse erzeugen allerdings oft fehlerbehaftete Daten. Sie enthalten Störungen, Messfehler oder Rauschen, deren Anteil irregulär ist. Daher sind diese Daten durch verlustlose Verfahren, bei denen keinerlei Information verloren geht, eher schlecht zu komprimieren. Es wird entschieden, einen nicht signifikanten Teil der vorliegenden Informationen zu vernachlässigen, um die verbleibenden relevanten Bildinformationen umso effizienter komprimieren zu können. Die Erzielung einer möglichst großen Kompressionsrate, das Vermeiden eventueller Verluste der (Bild-)Informationen und die Erhaltung der Befundungsqualität sind hierbei die Anforderung in der Medizin [2]. Öffentliche, hochwertige Verfahren im DICOM Standard (Digital Imaging and Communications in Medicine, ein offener Standard zum Austausch von Informationen in der Medizin) für die verlustbehaftete Bildkompression sind z.B. JPEG und JPEG2000. Sie genügen den ISO-Standards und werden intensiv evaluiert [3].

Viele verwendete Bildkompressionsverfahren, wie auch JPEG2000 oder EBCOT, basieren auf der diskreten Wavelet-Transformation und ähnlichen Techniken [13]. Die Nichtlinearität der Approximation eines Bildes mit Wavelets ergibt sich aus der Schwellenwertbildung der Waveletkoeffizienten. So lassen sich diejenigen Koeffizienten für die Übermittlung herausfiltern, die einen gewissen Schwellenwert überschreiten. Dieser Approximationstyp wird nichtlineare Waveletapproximation genannt. Für Bilder mit vielen Kanteninformationen erzeugt diese Art der Approximation jedoch unteroptimale Approximationsraten. Hierbei stellen die anisotropen Delaunay Triangulierungen, die in dem bestehenden nichtlinearen Kompressionsverfahren CAT verwendet werden, eine geeignete Alternative dar, um dieses Problem zu beheben.

Im ersten Kapitel dieser Arbeit beschäftigen wir uns mit den Grundlagen der Triangulierungen und der Bildapproximation mit Hilfe von Triangulierungen. Dabei werden wir auf die spezielle anisotrope Delaunay Triangulierung eingehen, die in dem bestehenden Kompressionsverfahren CAT Anwendung findet. Daraufhin werden wir einige Beispiele angegeben, um die Vorteile der adaptiven anisotropen Triangulierungen herauszuarbeiten. Des Weiteren werden wir eine kurze Einführung in die arithmetische Kodierung geben, die die Grundlage der kontextuellen Kompression unseres Verfahrens bildet. Im zweiten Kapitel werden wir die kontextuelle Kompressionsmethode CAT, die auf der anisotropen Delaunay Triangulierung basiert, beschreiben und auf die Methoden der kontextuellen adaptiven arithmetischen Kodierung dieses Verfahrens eingehen. Im dritten Kapitel folgt die Untersuchung neuer Ansätze und Methoden für die Bildung von neuen Kontexten, die das Verfahren CAT erweitern. Im vierten Kapitel werden wir letztlich das erweiterte Kompressionsverfahren CAT* vorstellen und numerischen Ergebnisse im Vergleich zu JPEG2000 und CAT angeben.

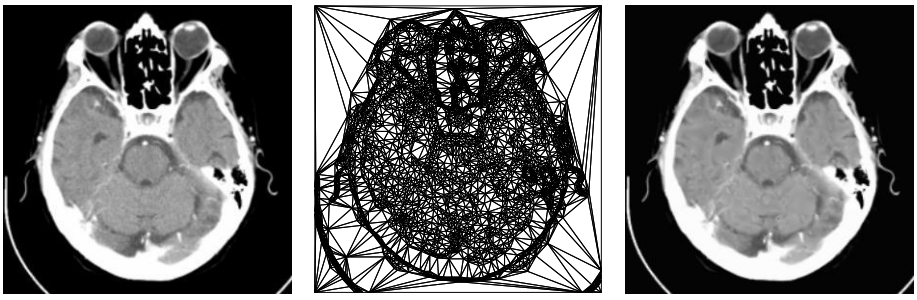


Abbildung 1.1: Schädel CT; Links: Originalbild ($300x300Pixel$, $267KB$), Mitte: Triangulierung, Rechts: Rekonstruktion nach der neuen Kompressionsmethode CAT*, Speicherplatzbedarf für das komprimierte Bild: 9,77 KB.

1.1 Triangulierungen: Definitionen und Theorie

In diesem Abschnitt führen wir wichtige Begriffe für die Definition der Triangulierungen ein. Dabei beschränken wir uns auf die Begriffe, die wir im Rahmen der Bildkompression benötigen. Daraufhin folgt die Definition einer speziellen Triangulierung, der Delaunay Triangulierung, die in unserer Kompressionsmethode Anwendung findet. Des Weiteren werden wir näher auf die grundlegenden Approximationsmethoden von digitalen Bildern bzw. Funktionen mit Hilfe von isotropen und anisotropen Triangulierungen eingehen und Beispiele für univariate und bivariate Funktionen angeben.

1.1.1 Grundlagen der Triangulierungen

Als erstes erfolgt nun die Festlegung einiger Notationen und die Einführung grundlegender Definitionen bezüglich der Triangulierungen und der zugehörigen Funktionenräume [6].

Sei $\Omega \subset \mathbb{R}^2$ eine kompakte ebene Menge mit einem polygonalen Rand. Obwohl wir versuchen diese Einführung allgemein zu halten, setzen die Meisten der folgenden Diskussionen $\Omega = [0,1]^2$ für den (stetigen) Bildbereich voraus.

Definition 1 *Ein* stetiges Bild *ist eine begrenzte und messbare Funktion*

$$f : \Omega \to [0, \infty),$$

sodass f in L^∞ liegt, dem Funktionenraum aller messbaren, im wesentlichen beschränkten Funktionen, d.h. $f \in L^\infty(\Omega)$.

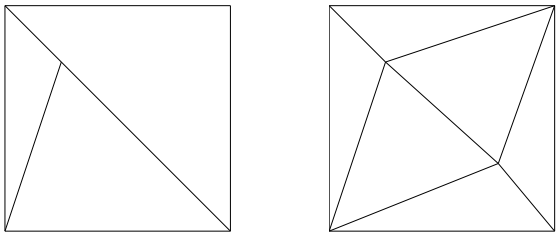
Obwohl Bilder begrenzt sind (d.h. in $L^\infty(\Omega) \subset L^p(\Omega)$ liegen), unterscheiden wir zwischen den verschiedenen Funktionenräumen $L^p(\Omega)$ bezüglich der unterschiedlichen Normen $\|\cdot\|_{L^p(\Omega)}$, um den Rekonstruktionsfehler der Menge Ω zu messen. Es ist wichtig zu erwähnen, dass ein Bild immer durch eine begrenzte Funktion dargestellt werden kann. Der Umkehrschluss ist (trivialerweise) nicht erfüllt: für jedes feste $p \in [1, \infty]$ entsprechen die meisten Funktionen in $L^p(\Omega)$ keinem natürlichen Bild. Eine der Hauptaufgaben der Funktionalanalysis ist es nun, so wenig Funktionenklassen wie möglich für die Beschreibung von Methoden in der Bildverarbeitung zu definieren. Diese Funktionenklassen werden durch Regularitätsbedingungen beschrieben und beinhalten die Darstellung relevanter Bilder. Bezüglich der Triangulierungen führt dies sofort zu einer zentralen Frage: Welche Klassen von Bildern können durch Approximationsmethoden, die auf Dreieicksnetzen basieren, gut wiedergegeben werden? Wir werden später in diesem Abschnitt auf diese Frage eingehen. Vorher widmen wir uns noch den grundlegenden Definitionen.

Definition 2 *Eine* Triangulierung $\mathcal{T}$ *eines Definitionsbereichs Ω ist eine endliche Menge $\{T\}_{T \in \mathcal{T}}$ geschlossener Dreiecke $T \in \mathbb{R}^2$, die folgende Bedingungen erfüllen.*

(a) *Die Vereinigung der Dreiecke in $\mathcal{T}$ bildet den Bereich Ω, d.h.*

$$\Omega = \bigcup_{T \in \mathcal{T}} T.$$

(b) *Für jedes Paar $T, T' \in \mathcal{T}$ zweier verschiedener Dreiecke, $T \neq T'$, ist der Schnitt*



(a) **(b)**

Abbildung 1.2: **(a)** eine nicht-konforme Triangulierung mit drei Dreiecken und fünf Ecken, wobei eine Ecke eine hängende Ecke ist. **(b)** eine konforme Triangulierung mit sechs Dreiecken und sechs Ecken.

über deren Inneres leer, d.h.

$$\overset{\circ}{T} \cap \overset{\circ}{T'} = \emptyset \qquad \text{für } T \neq T'.$$

Wir bezeichnen die Menge der Triangulierungen von Ω *mit* $\mathscr{T}(\Omega)$.

Die Bedingung (b) in Definition 2 erlaubt keine Überlappungen verschiedener Dreiecke in $\mathcal{T}$. Jedoch kann eine Triangulierung $\mathcal{T}$, entsprechend der oben angegebenen Definition, *hängende Ecken* besitzen. Dabei handelt es sich um eine Ecke eines Dreiecks in $\mathcal{T}$, die auf der Innenkante eines benachbarten Dreiecks liegt. Triangulierungen ohne hängende Ecken werden als *konform* bezeichnet. Dies führt uns zu der folgenden Definition.

Definition 3 *Eine Triangulierung* $\mathcal{T}$ *von* Ω *ist eine* konforme Triangulierung *von* Ω*, wenn sich jedes Paar zweier verschiedener Dreiecke aus* $\mathcal{T}$ *höchstens in einer gemeinsamen Ecke oder entlang einer gemeinsamen Kante schneiden. Wir bezeichnen die Menge aller konformen Triangulierungen von* Ω *mit* $\mathscr{T}_c(\Omega)$.

Abbildung 1.2 zeigt eine nicht-konforme und eine konforme Triangulierung.
Für einen ebenen Bereich Ω existieren verschiedene (konforme) Triangulierungen. Viele relevante Anwendungen basieren auf Triangulierungen, die lange und dünne Dreiecke besitzen, um numerische Stabilität zu erreichen. Ein Beispiel für eine solche Anwendung ist die Finite Elemente Methode. Hier ist die Delaunay Triangulierung eine beliebte Wahl. Diese spezielle Triangulierung wird im nächsten Abschnitt näher erklärt.

Wir wenden uns nun den Funktionen über Triangulierungen zu, z.B. den bivariaten Funktionen $f : \Omega \to \mathbb{R}$, die über $\mathcal{T}$ definiert sind. Eine Möglichkeit besteht in der Verwendung von stückweise definierten Polynomfunktionen. In diesem Fall

bilden die Restriktionen $f\big|_T$, bezüglich aller Dreiecke $T \in \mathcal{T}$, ein bivariates Polynom eines gewissen Grades. Um solche Polynomfunktionen über Triangulierungen zu definieren, wird im Vorhinein der maximale Grad des Polynoms festgelegt und zusätzliche Grenzen oder globale Glattheitsbedingungen gefordert.
Wir bevorzugen für unsere Bildapproximationen die Arbeit mit stückweise linearen Polynomfunktionen f. Darüber hinaus benötigen wir durch die Verwendung von linearen Polynomfunktionen, außer der globalen Stetigkeit für f, keine weiteren globalen Glattheitsbedingungen. Da ein natürliches Bild typischerweise unstetig ist, würde es Sinn machen, von der Forderung nach globaler Stetigkeit für f abzusehen. Dies führt uns zu den folgenden Definitionen zweier Funktionenräume von stückweise linearen Polynomen, eine mit der geforderten allgemeinen Stetigkeit und eine ohne. In diesen Definitionen bezeichnet $\mathcal{P}_1$ den linearen Raum bivariater linearer Polynome.

Definition 4 *Es sei $\mathcal{T} \in \mathscr{T}(\Omega)$ eine Triangulierung von Ω. Die Menge der* stückweise linearen Funktionen *über $\mathcal{T}$,*

$$\mathcal{S}_{\mathcal{T}} := \left\{ f : \Omega \to \mathbb{R} \ : f\big|_{\overset{\circ}{T}} \in \mathcal{P}_1 \right\},$$

wird von allen Funktionen beschrieben, dessen Restriktion auf die Innenkanten $\overset{\circ}{T}$ eines jeden Dreiecks $T \in \mathcal{T}$ ein lineares Polynom ist.

Dabei müssen wir beachten, dass in dieser Definition die Restriktion $f\big|_{\overset{\circ}{T}}$ von f auf jedes beliebige Dreieck $T \in \mathcal{T}$, von $\overset{\circ}{T}$ nach T erweitert werden könnte. In diesem Fall wäre f über Ω nicht wohldefiniert. Für eine konforme Triangulierung $\mathcal{T}(\Omega)$ jedoch wäre f über Ω wohldefiniert, wenn wir globale Stetigkeit voraussetzen.

Definition 5 *Es sei $\mathcal{T} \in \mathscr{T}_c(\Omega)$ eine konforme Triangulierung von Ω. Die Menge der* stetigen, stückweise linearen Funktionen *über $\mathcal{T}$,*

$$\mathcal{S}^0_{\mathcal{T}} := \left\{ f \in \mathscr{C}(\Omega) : f\big|_T \in \mathcal{P}_1 \right\},$$

wird von allen Funktionen über Ω beschrieben, dessen Restriktion auf jedes Dreieck $T \in \mathcal{T}$ ein lineares Polynom ist.

$\mathcal{S}^0_{\mathcal{T}}$ ist ein linearer Unterraum von $\mathcal{S}_{\mathcal{T}}$, d.h. $\mathcal{S}^0_{\mathcal{T}} \subset \mathcal{S}_{\mathcal{T}}$. Darüber hinaus ist $\mathcal{S}^0_{\mathcal{T}}$ endlich-dimensional. Die (endliche) Menge der *Courant Elemente* $\varphi_v \in \mathcal{S}^0_{\mathcal{T}}$, mit der Ecke v aus $\mathcal{T}$, die allgemein durch

$$\varphi_v(x) = \begin{cases} 1 & \text{for } x = v; \\ 0 & \text{for } x \neq v; \end{cases} \qquad \text{für jede Ecke } x \text{ in } \mathcal{T}$$

definiert ist, ist eine Basis von $\mathcal{S}^0_{\mathcal{T}}$. Daher ist die Dimension von $\mathcal{S}^0_{\mathcal{T}}$ gleich der Anzahl der Ecken in $\mathcal{T}$. Dementsprechend hat der lineare Funktionenraum $\mathcal{S}_{\mathcal{T}}$ die Dimension $3|\mathcal{T}|$, wobei $|\mathcal{T}|$ der Anzahl der Dreiecke in $\mathcal{T}$ entspricht.
Ein wichtiger Unterschied zwischen den zwei Approximationsräumen $\mathcal{S}^0_{\mathcal{T}}$ und $\mathcal{S}_{\mathcal{T}}$ ist, dass $\mathcal{S}^0_{\mathcal{T}}$ eine konforme Triangulierung voraussetzt, hingegen kann für $\mathcal{S}_{\mathcal{T}}$ die Triangulierung $\mathcal{T}$ hängende Ecken besitzen. Hierbei muss beachtet werden, dass dies zu unterschiedlichen Approximationsschemata führt.

Wir beenden diesen Abschnitt mit der Definition des *aspect ratios*, einer Qualitäts-Messung von Dreiecken.

Definition 6 *Es sei T ein Dreieck. Wir bezeichnen die maximale Kantenlänge von T mit h_T und den Radius des Inkreises mit r_T. Der* aspect ratio *von T ist definiert als der Quotient $\rho_T = h_T/r_T$.*

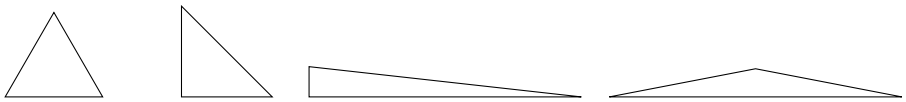


Abbildung 1.3: Vier Dreiecke T_1, T_2, T_3, T_4 gleicher Fläche, jedoch besitzen sie verschiedene aspect ratios $\rho_1 = 1$, $\rho_2 = 1.4$, $\rho_3 = 4.2$ und $\rho_4 = 6.1$ (von Links nach Rechts).

1.1.2 Delaunay Triangulierung

Die Delaunay Triangulierung, von Delaunay 1934 eingeführt, ist in den letzten Jahren zu einer sehr populären Methode zur Generierung von Dreiecksnetzen herangewachsen [12]. Neben dem bestehenden Bildkompressionsverfahren findet die Delaunay Triangulierung Anwendung in der Kartographie bzw. in der Geodäsie oder auch bei der Finite-Elemente-Methode. Die Delaunay Triangulierung wird einerseits für die Erstellung von zweidimensionalen Dreiecksnetzen, als auch für die Generierung dreidimensionaler Tetraedernetze verwendet [12]. In Bezug auf unser Bildkompressionsverfahren CAT interessieren wir uns für den zweidimensionalen Fall.

Definition 7 *Eine konforme Triangulierung einer endlichen Menge $X \subset \Omega$ von Punkten wird* Delaunay Triangulierung *$\mathcal{D}$ von Ω genannt, wenn das Innere des Umkreises eines jeden Dreiecks der Triangulierung, keinen Knoten von $\mathcal{D}$ beinhaltet.*

Die Delaunay Triangulierung $\mathcal{D}$ von Ω, mit den Knoten $X \subset \Omega$, maximiert den minimalen Winkel entlang allen möglichen Triangulierungen von Ω, mit Knoten in X. Daher ist die Delaunay Triangulierung eine *optimale* Triangulierungsmethode.

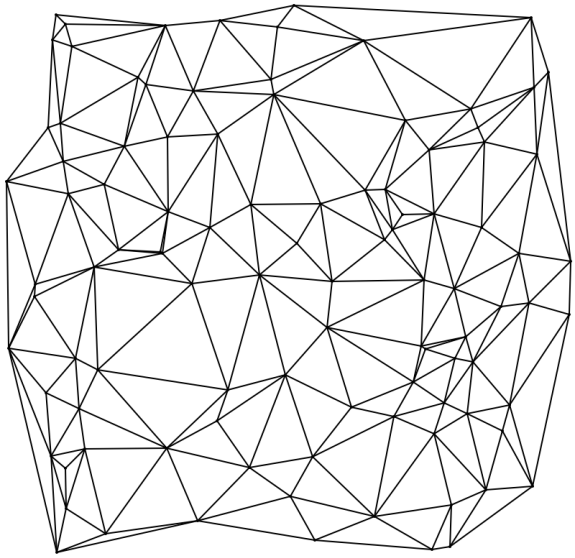


Abbildung 1.4: Die abgebildete Triangulierung ist eine Delaunay Triangulierung der Menge der Knoten

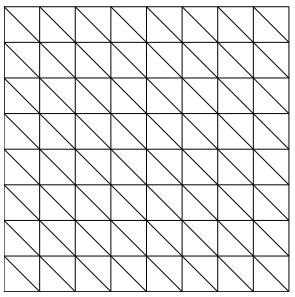
Darüber hinaus ist die Delaunay Triangulierung $\mathcal{D} \equiv \mathcal{D}(X)$ über allen Triangulierungen eindeutig bestimmt, unter der Voraussetzung, dass keine vier Knoten aus X cocircular sind, sprich wenn keine vier Knoten auf einem Umkreis liegen. Diese Eigenschaft wird *Delaunay Eigenschaft* genannt.

1.1.3 Isotrope und anisotrope Approximationsmethoden

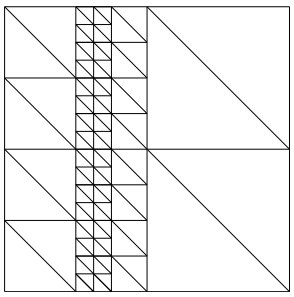
Bei der Approximation einer Funktion f, durch Funktionen der Räume $\mathcal{S}_{\mathcal{T}}^0$ oder $\mathcal{S}_{\mathcal{T}}$, hängt die Qualität der resultierenden besten Approximation von der Qualität und damit insbesondere von der Form der Dreiecke der Triangulierung $\mathcal{T}$ ab. Mehrere alternative Qualitätsmaße für Triangulierungen werden in [15] vorgestellt.

Die Approximationsmethoden mit Hilfe von Triangulierungen können in zwei große Klassen eingeteilt werden, die gleichmäßigen und die adaptiven Methoden. Des Weiteren unterscheidet man zwischen isotropen und anisotropen Methoden. Abbildung 1.5 zeigt Beispiele für gleichmäßige und adaptive und für isotrope und anisotrope Triangulierungen:

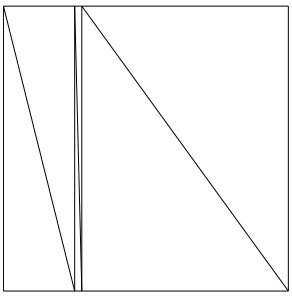
(a) Dreiecke einer *gleichmäßigen* Triangulierung haben vergleichbare Größen und Formen. Das bedeutet, dass jede Dreiecksform mit der eines gleichseitigen Dreiecks verglichen werden kann. Diese Art der Triangulierung kann als direkte Erweiterung der gleichmäßigen Einteilung im eindimensionalen Fall angesehen werden.



(a) gleichmäßige Triangulierung



(b) hierarchische, nichtgleichmäßige aber isotrope Triangulierung



(c) anisotrope Delaunay Triangulierung

Abbildung 1.5: Eine gleichmäßige, isotrope und anisotrope Triangulierung

(b) *Adaptive isotrope* Methoden erlauben uns, die Größe der Dreiecke zu verändern. Die äquivalenten Formen der Dreiecke bleiben jedoch erhalten. Grob gesprochen, kann diese Art des Zoomens durch homothetische Transformationen eines Referenz-Elements erzielt werden. Daher können diese Methoden mit den klassischen bidimensionalen Wavelet Methoden verglichen werden, die wir über das Tensorprodukt einer eindimensionalen Wavelet-Basis erhalten. Diese Triangulierungen sind besonders effizient, wenn sie auf Bilder angewendet werden, die Singularitäten an isolierten Punkten besitzen.

(c) Bei *anisotropen* Triangulierungen passen sich die Form und die Größe der Dreiecke der Struktur des Bildes an.

Wir werden dies in einer Definition formulieren. Es sollte beachtet werden, dass diese Definition nicht die einzige Möglichkeit ist, diesen Sachverhalt wiederzugeben. Ein Triangulierungs-Schema kann für eine Indexmenge $I \subset \mathbb{N}$ als eine Abbildung $L^2(\Omega) \times I \to \mathscr{T}(\Omega)$ oder $L^2(\Omega) \times I \to \mathscr{T}_c(\Omega)$, $(f,n) \mapsto \mathcal{T}_n$ beschrieben werden, mit $|\mathcal{T}_n| = n$.

Definition 8 *Ein Triangulierungs-Schema wird* gleichmäßige Triangulierung *genannt, wenn die folgenden zwei Bedingungen erfüllt sind:*

i. Es existieren zwei Konstanten C_1, C_2, sodass

$$\forall n \in \mathbb{N}, \forall\, T \in \mathcal{T}_n, \frac{C_1}{n} \leq |T| \leq \frac{C_2}{n}$$

gilt, wobei $|T|$ die Fläche des Dreiecks T bezeichnet.

ii. Es existieren Konstanten $C_3, C_4 > 0$, sodass

$$\forall\, n \in \mathbb{N}, \forall\, T \in \mathcal{T}_n, C_3 \leq \frac{h_T}{\rho_T} \leq C_4$$

gilt.

Eine isotrope adaptive Triangulierung *erfüllt die Bedingung ii, jedoch nicht notwendigerweise Bedingung i. Eine* anisotrope Triangulierung *wird dadurch definiert, dass sie a priori keine der beiden Bedingungen erfüllt.*

Gleichmäßige Bildapproximationen, die auf gleichmäßigen Triangulierungen beruhen, sind bezüglich der Qualität der Approximation den adaptiven Methoden unterlegen. Jedoch muss bei den adaptiven Bildapproximationsmethoden ein besonderes Augenmerk auf die zugrundeliegende adaptive Triangulierungen gelegt werden. Adaptive isotrope Triangulierungen sind für die Situationen geeignet, in denen das Bild an isolierten Punkten Sigularitäten aufweist. Für die lokale Anpassung an Sigularitäten eines Bildes entlang Kurven oder anderen Gegebenheiten des Bildes, die durch die heterogene Verteilung der Richtungsableitungen der ersten oder zweiten Ableitung gegeben sind, bieten sich hingegen anisotrope Triangulierungen besonders an. Daher werden anisotrope Triangulierungen bevorzugt für die Konstruktion geeigneter Triangulierungen für adaptive Bildapproximationsmethoden verwendet.

1.1.4 Abstrakte Formulierung des Approximationsproblems

Eine der Haupteigenschaft, die für die Entwicklung einer guten Triangulierungsmethode benötigt wird, ist die Fähigkeit relevante Bildklassen mit einer hinreichend schnellen Konvergenzrate zu approximieren. Wir werden einige Notationen und Konzepte einführen, die nötig sind, um diesen abstrakten Sachverhalt zu formalisieren.

Der Fehler zwischen dem Originalbild und dessen Approximation kann in jeder $L^p(\Omega)$-Norm gemessen werden. Um jedoch zusätzliche technische Schwierigkeiten zu vermeiden, beschränken wir die folgende Diskussion auf den Fall $p = 2$.

Für eine gegebene Triangulierung $\mathcal{T}$ eines Bildbereiches Ω, bezeichnen wir mit $\widetilde{\Pi}_{\mathcal{T}} f$ die orthogonale Projektion von f auf $\mathcal{S}_{\mathcal{T}}$. Es gilt $\widetilde{\Pi}_{\mathcal{T}} f \in \mathcal{S}_{\mathcal{T}}$. Des Weiteren erfüllt $\widetilde{\Pi}_{\mathcal{T}} f$ die Ungleichung

$$\|f - \widetilde{\Pi}_{\mathcal{T}} f\|_{L^2(\Omega)} \leq \|f - g\|_{L^p(\Omega)} \text{ für jedes } g \in \mathcal{S}_{\mathcal{T}}.$$

Gleichermaßen führen wir die Notation $\Pi_{\mathcal{T}} f$ ein, für die orthogonale Projektion von f auf $\mathcal{S}^0_{\mathcal{T}}$, mit $\Pi_{\mathcal{T}} f \in \mathcal{S}^0_{\mathcal{T}}$ und

$$\|f - \Pi_{\mathcal{T}} f\|_{L^2(\Omega)} \leq \|f - g\|_{L^p(\Omega)} \text{ für jedes } g \in \mathcal{S}^0_{\mathcal{T}}.$$

Intuitiv werden Approximationsklassen als Klassen von Funktionen definiert, die die gleiche Konvergenzgeschwindigkeit bezüglich des Approximationsfehlers besitzen. Eine Möglichkeit diesen Sachverhalt zu formulieren ist die Folgende:

$$\widetilde{\mathcal{A}^\alpha} := \left\{ f \in L^2(\Omega) | \; \exists\, C > 0 : \inf_{|\mathcal{T}|=n, \mathcal{T} \in \mathcal{T}(\Omega)} \|\widetilde{\Pi}_{\mathcal{T}} f - f\|_{L^2(\Omega)} \leq \frac{C}{n^\alpha} \right\}$$

und entsprechend

$$\mathcal{A}^\alpha := \left\{ f \in L^2(\Omega) | \; \exists\, C > 0 : \inf_{|\mathcal{T}|=n, \mathcal{T} \in \mathcal{T}_c(\Omega)} \|\Pi_{\mathcal{T}} f - f\|_{L^2(\Omega)} \leq \frac{C}{n^\alpha} \right\}.$$

Das abstrakte Problem der Approximation ist nun, eine hinreichende und allgemeine Bedingung für die Funktion f zu finden, um der Klasse $\mathcal{A}^\alpha$ bzw. $\widetilde{\mathcal{A}^\alpha}$ zugeordnet zu werden.

1.1.5 Beispiele der Approximationsmethoden

Ziel dieses Abschnitts ist, eine anschauliche Einführung in die Thematik der Approximationen zu geben. Es handelt sich hierbei nicht um eine vollständige Darstellung der Approximationstheorie, sondern lediglich um ein paar ausgewählte illustrierende Beispiele, um den Zusammenhang zwischen der Approximationstheorie und den isotropen und anisotropen Triangulierungs-Methoden für digitale Bilder herzustellen.

Gleichmäßige Intervalleinteilung

Für die Analyse der gleichmäßigen Approximation betrachten wir zuerst ein einfaches Beispiel für ein quadratisches Polynom. Es sei

$$f : [0,1] \to \mathbb{R},\; f(x) := \frac{Mx^2}{2},\;\; \text{mit } M > 0.$$

Weiter sei $\hat{f}(x) = f(0) + f'(0)x$ die Linearisierung von f an der Stelle $x = 0$. Dies liefert uns die triviale Approximation $\hat{f} = 0$ und damit die folgende Abschätzung des Approximationsfehlers im Intervall $[0,h)$:

$$\|f - \hat{f}\|^2_{L^2([0,h])} = \int_0^h |f''(x)|^2 \left(\frac{x^2}{2}\right)^2 dx = M^2 \cdot \int_0^h \frac{x^4}{4} dx = M^2 \cdot \frac{h^5}{20},$$

oder äquivalent

$$\|f - \hat{f}\|_{L^2(I_h)} = C \cdot M\sqrt{h} \cdot h^2 \text{ mit } C := \frac{1}{\sqrt{20}}.$$

Offensichtlich gilt diese allgemeine Abschätzung für jedes Intervall $I_h = [x_0, x_0 + h] \subset [0,1]$ und jedes f, dessen zweite Ableitung konstant ist über I_h, d.h. $f''(x) = M$ für alle $x \in [0,1]$. Dies ergibt

$$\|f''\|_{L^2(I_h)} = \sqrt{\int_{I_h} |f''(x)|^2 dx} = \sqrt{h} \cdot M.$$

Zusammengefasst und für $h = |I_h|$ gilt für den Approximationsfehler

$$\|f - \hat{f}\|_{L^2(I_h)} = \frac{1}{\sqrt{20}} \cdot |I_h|^2 \cdot \|f''\|_{L^2(I_h)}.$$

Des Weiteren erhalten wir für die Beziehung $\frac{1}{\tau} = \frac{1}{2} + 2 \Longleftrightarrow \tau = \frac{2}{5} (< 1)$ ($\frac{1}{2}$ bezieht sich darauf, dass wir den Fehler in der L^2-Norm messen und 2 darauf, dass $f \in \mathscr{C}^2([0,1])$) nach einigen Rechenschritten die Gleichung

$$\|f - \hat{f}\|_{L^2(I_h)} = C \cdot \|f''\|_{L^\tau(I_h)}.$$

Wir betrachten nun den Fehler der stückweise linearen Approximation über der gleichmäßigen Einteilung des Intervalls $[0,1]$ in n Teilintervalle $I_i = [i/n, (i+1)/n[$, mit $h_i = |I_i| = 1/n$. Da f'' konstant ist, gilt

$$\|f''\|^\tau_{L^\tau(I_i)} = \frac{1}{n} \cdot \|f''\|^\tau_{L^\tau([0,1])}, \text{ für } i = 1, \ldots, n.$$

Es folgt

$$\|f - \hat{f}\|^2_{L^2([0,1])} = \sum_{i=0}^{n-1} \|f - \hat{f}\|^2_{L^2(I_i)} = \frac{Cn}{n^{2/\tau}} \cdot \|f''\|^2_{L^\tau([0,1])}.$$

Mit $2/\tau - 1 = 4$ erhalten wir die klassische Fehlerabschätzung, entsprechend dem Approximationsfehler der Finite Elemente Theorie,

$$\|f - \hat{f}\|_{L^2([0,1])} = \frac{C}{n^2} \|f''\|_{L^\tau([0,1])}, \tag{1.1}$$

da $f \in \mathscr{C}^2([0,1])$ und $\|f''\|_{L^2([0,1])} = \|f''\|_{L^\tau([0,1])}$. Die gleichmäßige Einteilung des Intervalls $[0,1]$ ist somit optimal für quadratische Polynome.

Adaptive Intervalleinteilung

Interessant wird die Analyse, wenn wir eine Funktion g betrachten, deren zweite Ableitung nicht konstant ist. Wir nehmen an, dass die Funktion g stückweise konstant ist, d.h. $g \in \mathscr{C}^{1,1}([0,1])$, wobei $\mathscr{C}^{1,1}([0,1])$ den Raum der Funktionen beschreibt, deren zweite Ableitung g'' in $L^\infty([0,1])$ liegt. Analysieren wir beispielsweise die Funktion $g : [0,1] \in \mathscr{C}^1([0,1])$, die folgendermaßen definiert ist:

$$g''(x) = \begin{cases} M_1 & \text{für } x \in [0,h_1[\\ M_2 & \text{für } x \in]h_1,1] \end{cases}.$$

Da g'' auf den Intervallen $I_1 = [0,h_1]$ und $I_2 = [h_1,1]$ konstant ist, können wir uns für jedes der Intervalle auf das Ergebnis der vorangegangenen Diskussion beziehen. Die Problemstellung ist nun, für eine gegebene Anzahl an Teilintervallen $n \in \mathbb{N}$, n_1 und n_2, mit $n_1 + n_2 = n$, zu finden, sodass der Fehler bei einer stückweise linearen Approximation auf den Intervallen

$$I_{1,i} := \left[\frac{ih_1}{n_1}, \frac{(i+1)h_1}{n_1}\right] \text{ und } I_{2,j} := \left[h_1 + \frac{jh_2}{n_2}, h_1 + \frac{(j+1)h_2}{n_2}\right]$$

minimiert wird, mit $h_2 = 1 - h_1$. Mit der Idee, den gleichen Fehler für jedes kleine Intervall zu erlauben, d.h.

$$\|g - \hat{g}\|^2_{L^2(I_{1,i})} := \|g - \hat{g}\|^2_{L^2(I_{2,j})} \text{ für } i = 1,\ldots,n_1 \text{ und } j = 1,\ldots,n_2,$$

ergibt sich das Verhältnis

$$M_1^2 \left(\frac{h_1}{n_1}\right)^5 = M_2^2 \left(\frac{h_2}{n_2}\right)^5 \Longleftrightarrow M_1^\tau \frac{h_1}{n_1} = M_2^\tau \frac{h_2}{n_2}.$$

Nach ein paar Rechenschritten erhalten wir daraus und mit der Verwendung der vorangegangenen Ergebnisse für die Anzahl der Teilintervalle n_1 und n_2,

$$n_1 = \frac{\|g''\|^\tau_{L^\tau(I_1)}}{\|g''\|^\tau_{L^\tau([0,1])}} \cdot n$$
$$n_2 = \frac{\|g''\|^\tau_{L^\tau(I_2)}}{\|g''\|^\tau_{L^\tau([0,1])}} \cdot n.$$

Sind die Werte für n_1 und n_2 keine Ganzzahlen, so wird die nächstgelegene Ganzzahl gewählt. Wir beziehen nun die Fehlerabschätzung der vorangegangenen Dis-

kussion auf jedes Teilintervall und erhalten für den Approximationsfehler

$$\|g-\hat{g}\|^2_{L^2([0,1])} = C^2 \cdot \left(\sum_{i=1}^{n_1} \frac{1}{n_1^5} \|g''\|^2_{L^\tau(I_1)} + \sum_{j=1}^{n_2} \frac{1}{n_2^5} \|g''\|^2_{L^\tau(I_2)} \right).$$

Mit den Werten für n_1 und n_2 und der Fehlerabschätzung (1.1) für jedes Teilintervall, erhalten wir letztlich die Abschätzung

$$\|g-\widehat{g_n}\|^2_{L^2([0,1])} = C^2 \left(\sum_{i=1}^{n_1} \frac{1}{n^5} \|g''\|^2_{L^\tau([0,1])} + \sum_{j=1}^{n_2} \frac{1}{n^5} \right) \text{ ,mit}$$

$$\|g''\|^2_{L^\tau([0,1])} = \frac{C^2}{n^4} \|g''\|^2_{L^\tau([0,1])}, \tag{1.2}$$

die den Approximationsfehler (1.1) verallgemeinert. Durch die Verwendung adaptiver Intervalle, konnten wir den konstanten Wert für die Abschätzung des Fehlers für Funktionen, die in oder fast in $\mathscr{C}^2([0,1])$ liegen, verbessern. Andere Funktionen können auf die gleiche Weise behandelt werden und erfüllen damit die gleiche Fehlerabschätzung.

Dazu betrachten wir für den eindimensionalen Fall zuletzt eine Funktionenschar, bestehend aus einer Sequenz von antisymmetrischen Funktionen f_ε, mit $\varepsilon > 0$, die folgendermaßen definiert sind:

$$f_\varepsilon(x) = \begin{cases} -\frac{1}{\varepsilon^2}(x-\varepsilon)^2 + 1 & \text{wenn } x \in [0,\varepsilon] \\ 1-\varepsilon(x-\varepsilon)^2 & \text{wenn } x \in [\varepsilon,1] \end{cases}.$$

Es gilt $f_\varepsilon \in \mathscr{C}^{1,1}([-1,1])$, mit der zweiten Ableitung

$$f''_\varepsilon(x) = \begin{cases} -\frac{2}{\varepsilon^2} & \text{wenn } x \in [0,\varepsilon] \\ -\varepsilon & \text{wenn } x \in [\varepsilon,1] \end{cases}.$$

Daraus erhalten wir für die Funktionen f_ε die Grenzwerte für die Sobolev Halbnorm und die Besov Halbnorm:

$$|f_\varepsilon|_{W^{2,2}([-1,1])} := \|f''_\varepsilon\|_{L^2([-1,1])} \geq \frac{4}{\varepsilon^3} \xrightarrow[\varepsilon\to 0]{} \infty$$
$$, |f_\varepsilon|_{B^2_{\tau,\tau}([-1,1])} := \|f''_\varepsilon\|_{L^\tau([-1,1])} \xrightarrow[\varepsilon\to 0]{} 2.$$

Das bedeutet, dass $f_\varepsilon \notin W^{2,2}([-1,1])$, jedoch $f_\varepsilon \in B^2_{\tau,\tau}([-1,1])$, dem Besov-Raum. Der Sobolev-Raum $W^{2,2}([-1,1])$ ist der Funktionenraum der reellwertigen Funktionen $g \in L^2(T)$, deren partiellen schwachen Ableitungen bis zur Ordnung 2 in

$L^2(T)$ liegen. Weiter bezeichnen wir mit f_δ,

$$f_\delta = \begin{cases} -1 & \text{wenn } x < 0 \\ 1 & \text{sonst} \end{cases},$$

die Sprungfunktion, gegen die die Sequenz f_ε konvergiert, $\|f_\varepsilon - f_\delta\|_{L^2([-1,1])} \underset{\varepsilon\to 0}{\longrightarrow} 0$. Mit der Verwendung der Fehlerabschätzung (1.2) bezüglich adaptiver Intervalle, erhalten wir die Abschätzung

$$\|f_\varepsilon - \widehat{f_{\delta,n}}\|_{L^2([-1,1])} \leq \frac{C'}{n^2}, \tag{1.3}$$

für eine konstante C'. Hätten wir eine gleichmäßige Einteilung des Intervalls $[-1,1]$ gewählt, würde der Fehler aufgrund der Sobolev Halbnorm explodieren. Eine notwendige und hinreichende Bedingung an eine Funktion f, um der Fehlerabschätzung (1.2) bei einer adaptiven Intervallbildung zu genügen, ist also, dem Besov-Raum $B^2_{\tau,\tau}([-1,1])$ anzugehören. Aus der eben erhaltenen Fehlerabschätzung (1.3) lässt sich schließen, dass für stückweise glatte Funktionen die Größe der Intervalle bei der adaptiven Methode auf dem Inversen der Halbnorm $|f|_{B^2_{\tau,\tau}(\Omega)}$ der Funktion basiert. Wäre $f \in W^{2,2}(\Omega)$, würden wir lediglich eine Verbesserung der Konstanten bezüglich der klassischen Fehlerabschätzung bei der gleichmäßigen Einteilung eines Intervalls erhalten. Für die aktuell betrachtete Sprungfunktion hingegen, erlangen wir eine Änderung der Approximationsrate.

Kommen wir nun zu der Fehleranalyse für bivariate Funktionen $f \in \Omega \subset \mathbb{R}^2 \to \mathbb{R}$ und betrachten eine Funktion $f \in \mathscr{C}^2(\Omega)$, mit dem Gradienten

$$Df(x_0,y_0) = \left(\frac{\partial f}{\partial x}(x_0,y_0)\ \frac{\partial f}{\partial y}(x_0,y_0)\right)$$

und der Hessematrix

$$D^2 f(x_0,y_0) := \begin{pmatrix} \frac{\partial^2 f}{\partial x^2}(x_0,y_0) & \frac{\partial^2 f}{\partial x \partial y}(x_0,y_0) \\ \frac{\partial^2 f}{\partial x \partial y}(x_0,y_0) & \frac{\partial^2 f}{\partial^2 y}(x_0,y_0) \end{pmatrix}.$$

Um eine Abschätzung des Approximationsfehlers einer Funktion bezüglich einer stückweisen Funktion über einer Triangulierung $\mathcal{T}$ zu erhalten, wird in der Finite Elemente Methode die Fehlerabschätzung für ein einzelnes Dreieck $T \in \mathcal{T}$ bereitgestellt [16]. Die Hauptabschätzung erhalten wir dabei vom Bramble Hilbert Lemma [17]. Für $f \in W^{2,2}(T)$ erhalten wir folgende Fehlerabschätzung:

$$\|f - \widetilde{\Pi}_T f\|_{L^2(T)} \leq |f|_{W^{2,2}(T)} := \|D^2 f\|_{L^2(T)},$$

wobei $\widetilde{\Pi}_T f$ die orthogonale L^2-Projektion von f auf $\mathcal{S}_T$ darstellt. Wir wollen dieses Ergebnis, wie im eindimensionalen Fall, bezüglich der Adaptivität und der Anisotropie untersuchen.

Gleichmäßige Triangulierung

Dazu analysieren wir zuerst das quadratische Polynom f_1, das wie folgt definiert ist:

$$f_1(x,y) := M\left(\frac{x^2+y^2}{2}\right),$$

mit der Hessematrix

$$D^2 f_1(x,y) = \begin{pmatrix} M & 0 \\ 0 & M \end{pmatrix} \text{ für alle } (x,y) \in \Omega.$$

Die Taylorentwicklung zweiter Ordnung für die Funktion f_1 an der Stelle $(x_0,y_0) \in \Omega$, für $h = (h_1,h_2) = (x-x_0, y-y_0)$, lässt sich wie folgt darstellen:

$$f_1(x,y) = f_1(x_0,y_0) + Df_1 h + \frac{1}{2} h^t D^2 f_1 h.$$

Für ein gleichseitiges rechtwinkliges Dreieck T der Fläche $h^2/2$, ergibt sich folgende Fehlerabschätzung:

$$\begin{aligned} \|f_1 - \widetilde{\Pi}_T f_1\|^2_{L^2(T)} &= \int_T |f_1(x,y) - \Pi_T f_1(x,y)|^2 \, dxdy \\ &\leq \int_T M^2 \left(h_1^2 + h_2^2\right)^2 dh_1 dh_2 \leq C \cdot M^2 h^6, \end{aligned}$$

mit der von T und f_1 unabhängigen Konstanten $C > 0$. Diese Abschätzung ist äquivalent zu

$$\|f_1 - \widetilde{\Pi}_T f_1\|_{L^2(T)} \leq M \cdot h^{6/2} = Mh^3 = M \cdot |T|^{3/2} = \|D^2 f_1\|_{L^{2/3}(T)}. \tag{1.4}$$

Die $L^2(T)$-Norm der zweiten Ableitung wurde hier ersetzt durch die schwächere L^τ-Norm, wobei τ, entsprechend dem eindimensionalen Fall, die Beziehung

$$\frac{1}{\tau} := \frac{1}{p} + \frac{2}{2} \Longleftrightarrow \tau = \frac{2}{3}$$

erfüllt. Wir erhalten nach ein paar Berechnungen, entsprechend dem eindimensionalen Fall, die globale Fehlerabschätzung für die Approximation von Funktionen $f \in W^{2,2}(\Omega)$ mit Hilfe gleichmäßiger Triangulierung:

$$\|f - \widetilde{\Pi}_{\mathcal{T}_n} f\|_{L^2(\Omega)} \leq \frac{1}{n} |f|_{W^{2,2}(\Omega)} \text{ für } f \in W^{2,2}(\Omega). \tag{1.5}$$

Für quadratische Polynome mit konstanter Ableitung ist die gleichmäßige Triangulierung optimal.

Adaptive isotrope Triangulierung

Um nun eine Idee für die optimale Größe eines Dreiecks bei einer adaptiven isotropen Triangulierung zu bekommen, betrachten wir die stückweise quadratische Funktion $f_2 : [-1,1] \times [0,1] \to \mathbb{R}$, mit $\Omega_- := \{(x,y) | \, x < 0\}$ und $\Omega_+ := \{(x,y) | \, x \geq 0\}$, die wie folgt definiert ist:

$$f_2(x,y) = \begin{cases} M_1\left(\frac{x^2+y^2}{2}\right) & \text{wenn } (x,y) \in \Omega_- \\ M_2\left(\frac{x^2+y^2}{2}\right) & \text{wenn } (x,y) \in \Omega_+ \end{cases} .$$

Unter der Verwendung von (1.4) und der Vereinbarung, dass auf jedem Teilintervall der gleiche Fehler zugelassen ist, ergibt sich die im Folgenden dargestellte adaptive isotrope Triangulierung. Wir bezeichnen mit n_1 bzw. n_2 die Anzahl der Dreiecke in den Bereichen $x < 0$ bzw. $x > 0$ und erhalten, entsprechend dem eindimensionalen Fall, folgendes Verhältnis

$$\frac{|T_{1,i}|}{|T_{2,j}|} = \frac{M_2^\tau}{M_1^\tau} \text{ für alle } 1 \leq i \leq n_1, 1 \leq j \leq n_2.$$

Durch die Verwendung adaptiver isotroper Triangulierungen gewinnen wir wiederum eine Verbesserung in der Konstante bezüglich der Fehlerabschätzung (1.5) im gleichmäßigen Fall:

$$\|f - \widetilde{\Pi}_{\mathcal{T}_n} f\|_{L^2(\Omega)} \leq \frac{1}{n} |f|_{B^2_{\tau,\tau}(\Omega)} \text{ für } f \in W^{2,2}(\Omega), \tag{1.6}$$

wobei $(\mathcal{T}_n)_{n\in\mathbb{N}}$ die adaptive Triangulierung aus Definition (8) wiedergibt. Des Weiteren ist diese Aussage ebenfalls für Funktionen aus $B^2_{\tau,\tau}$ gültig. Insbesondere für bivariate Funktionen, die überall glatt sind, bis auf eine endliche Menge von isolierten Sigularitäten in den *cusps*, wird die Verbesserung durch eine adaptive Triangulierung sichtbar.

Der isotrope Ansatz geht jedoch an seine Grenzen, wenn wir Funktionen betrachten, die eine stark gerichtete Struktur aufweisen. Die Richtungsabhängigkeit wird dabei durch eine Hessematrix mit stark variierenden Eigenwerten reflektiert.

Adaptive anisotrope Triangulierung

Studieren wir beispielswiese die quadratische Funktion

$$P(x,y) = \frac{\lambda_1 x^2 + \lambda_2 y^2}{2},$$

mit der Hessematrix

$$D^2P(x,y) = \begin{pmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{pmatrix}.$$

Gilt für die Eigenwerte $\lambda_1 >> \lambda_2 > 0$, so ist der isotrope Ansatz nicht optimal. Wir werden aber im Folgenden sehen, dass anistrope Triangulierungsmethoden die Approximation verbessern. Für ein rechtwinkliges Dreieck T, mit den Werten h_1 und h_2, ergibt sich die Fehlerabschätzung

$$\begin{aligned} \|P - \widetilde{\Pi}_T P\|^2_{L^2(T)} &\leq \int_T \left(\lambda_1 h_1 2 + \lambda h_2^2\right) 2dh_1 dh_2 \\ &\leq C_1\lambda_1 2h_1 5h_2 + C_2\lambda_1\lambda_2 h_1^3 h_2^3 + C_3\lambda_2 2h_2^5 h_1, \end{aligned}$$

mit den von P unabhängigen Konstanten C_1, C_2, C_3. Unter der Annahme, dass $|T|$ fest ist, führt das Verhältnis

$$\frac{h_2}{h_1} = \sqrt{\frac{\lambda_1}{\lambda_2}}$$

zu einem minimalen Fehler:

$$\begin{aligned} \|P - \widetilde{\Pi}_T P\|_{L^2(T)} &= C \cdot \frac{\sqrt{\lambda_1}}{\sqrt{\lambda_2}} \sqrt{\lambda_1\lambda_2} \frac{h_1}{h_2} |T|^{3/2} = C \cdot \sqrt{\det(D^2P)} \cdot |T|^{3/2} \\ &= C \cdot \|\det(D^2P)\|_{L^\tau(T)} \\ &= \frac{C}{n^{1/\tau}} \cdot \|\det(P)\|_{L^\tau(\Omega)}. \end{aligned}$$

Damit erhalten wir für den globalen Approximationsfehler über Ω:

$$\|P - \widetilde{\Pi}P\|^2_{L^2(\Omega)} = \frac{C^2 n}{n^{2/\tau}} \cdot \|\det(P)\|^2_{L^\tau(\Omega)} = \frac{C^2}{n^2} \|\det(P)\|^2_{L^\tau(\Omega)}.$$

Da $P \in W^{2,2}(\Omega)$ (und sogar in $\mathscr{C}^2(\Omega)$), haben wir bei dieser neuen Abschätzung lediglich eine Verbesserung in der Konstanten.

Bei der Betrachtung der Verallgemeinerung $F_\varepsilon : \Omega \to \mathbb{R}$ der univariaten Funktion f_ε, mit

$$F_\varepsilon(x,y) = f_\varepsilon(x) + \varepsilon y^2$$

und der Hessematrix

$$\left|D^2F_\varepsilon(x,y)\right| = \begin{cases} \begin{pmatrix} \frac{2}{\varepsilon^2} & 0 \\ 0 & 2\varepsilon \end{pmatrix} & \text{wenn } |x| < \varepsilon \\ \begin{pmatrix} 2\varepsilon & 0 \\ 0 & 2\varepsilon \end{pmatrix} & \text{wenn } |x| > \varepsilon \end{cases},$$

erhalten wir nicht nur eine quantitative sondern auch eine qualitative Verbesse-

rung gegenüber den adaptiven isotropen Verfahren. Der Grund dafür ist, dass die Sprungfunktion F_δ, gegen die F_ε konvergiert, für $\varepsilon \to 0$, im Gegensatz zum eindimensionalen Fall, nicht in $B^2_{\tau,\tau}(\Omega)$ liegt. Dies hat zur Folge, dass die Fehlerabschätzung (1.6) für die isotrope adaptive Triangulierung bezüglich der Konstanten explodiert und nicht gültig für F_δ ist. Im Allgemeinen ist in diesen Fällen die anisotrope adaptive Triangulierung die beste Wahl.

1.2 Arithmetische Kodierung

In diesem Abschnitt beschäftigen wir uns mit der Beschreibung der arithmetischen Kodierung. Sie ist eine Form der Entropiekodierung und erreicht eine hohe Effizienz, da sie die Beschränkung umgeht, zu kodierende Symbole durch eine ganzzahlige Anzahl von Bits darzustellen. Für jede Datenquelle kann nahezu die bestmögliche Kompression erreicht werden, welche durch die Entropie beschränkt wird. Bevor wir die arithmetische Kodierung im Detail erklären und mittels eines Beispiels das Verfahren darstellen, werden wir einige grundlegende Begriffe einführen, die wir benötigen, um das Verfahren zu beschreiben. Daraufhin werden wir einen Algorithmus für die adaptive arithmetische Kodierung angeben. In einem letzten Abschnitt diskutieren wir die kontextuelle adaptive arithmetische Kodierung, die in dem bestehenden Kompressionsverfahren CAT und dem im vierten Kapital neu entwickelten Verfahren CAT* Anwendung findet.

1.2.1 Grundlagen der Kodierungstheorie

Es existieren verschiedene Arten von Daten, wie beispielsweise Textdaten, binäre Dateien oder in unserem Fall Bilder, die in komprimierter Form im Computer gespeichert oder durch Leitungen gesendet werden. Dazu sind verschiedene Darstellungsarten üblich, z.B. über die Repräsentation durch 0 und 1, über Buchstaben oder Pixel. Um eine abstrakte Beschreibung dieser Daten zu erhalten, führen wir den Begriff des *Symbols* ein. Auf diese Weise muss nicht weiter auf die verschiedenen Arten von Daten eingegangen werden.

Definition 9 (Alphabet und Symbol) *Eine endliche nichtleere Menge A heißt* Alphabet. *Mit $|A|$ bezeichnen wir die* Länge *oder Kardinalität des Alphabets. Die Elemente $\{a_1, \ldots, a_m\}$ eines Alphabets werden* Symbole *genannt.*

Die Anordnung der Symbole in A sei außerdem festgelegt. Die Verarbeitung der Daten im Computer erfolgt in einer eindeutigen Reihenfolge und mit Hilfe eines endlich großen Alphabets. Für die Übernahme dieser Struktur führen wir den Begriff der *Sequenz* ein.

Definition 10 (Sequenz) *Eine Folge $S = (s_1, s_2, \ldots)$ von Symbolen a_i aus dem Alphabet A heißt* Sequenz. *Abkürzend schreiben wir im Folgenden $S = s_1 s_2 \ldots$.*

Sofern die Sequenz eine Länge besitzt, wird diese durch $|S|$ beschrieben. Damit die folgenden Aussagen über empirische Wahrscheinlichkeiten sinnvoll sind, nehmen wir an, dass die Sequenzen eine Länge besitzen.
Betrachten wir eine gegebene Sequenz, dann stellen wir fest, dass die einzelnen Symbole dieser Sequenz mit einer gewissen, oft stark variierenden, Häufigkeit auftreten. Ein Beispiel wäre der Buchstabe *e* in deutschen Texten, der wesentlich häufiger auftritt als das *y*. Da die arithmetische Kodierung unter anderem auf diesen verschiedenen Häufigkeiten beruht, definieren wir die *empirische Wahrscheinlichkeit* eines Symbols.

Definition 11 (Empirische Wahrscheinlichkeit) *Sei $S = s_1 \dots s_n$ eine endliche Sequenz der Länge $|S| = n$ mit Symbolen aus $A = \{a_1, \cdots, a_m\}$ und $n(a_i)$ die absolute Häufigkeit des Symbols a_i in S. Dann ist $P(a_i) := \frac{n(a_i)}{n}$ die* empirische Wahrscheinlichkeit *bzw. die* relative Häufigkeit *des Symbols a_i (in S).*

Somit liegt die empirische Wahrscheinlichkeit $P(a_i)$ eines Elements im Intervall $[0,1)$. Die Summe über die empirischen Wahrscheinlichkeiten ist $\sum_{i=1}^{n} P(a_i) = 1$. Da konstante Sequenzen mit einem Symbol der relativen Häufigkeit 1 nicht kodiert werden müssen, da ihr Inhalt schon vollständig bekannt ist, betrachten wir lediglich das halboffene Intervall. Wenn wir uns noch einmal das Beispiel (e, y) ansehen, erkennen wir, dass die Wahrscheinlichkeiten der Symbole stark vom Kontext abhängen. Es ist also nötig eine Abbildungsvorschrift zu definieren, die festlegt, wie die Wahrscheinlichkeiten einzelnen Symbolen oder Symbolfolgen, die wiederum als Symbole aufgefasst werden können, zugeordnet werden sollen. Dazu dient das *Modell*.

Definition 12 (Modell) *Sei A ein Alphabet. Ein* Modell *M ist eine Abbildung*

$$\Phi\colon\ A \to [0,1)\colon\ a_i \mapsto P_M(a_i),$$

die jedem Element $a_i \in A$ eine (ggf. geschätzte) Wahrscheinlichkeit $P_M(a_i)$ zuordnet.

Hierbei ist zu beachten, dass ein Alphabet nicht zwingend aus Symbolen der Länge 1 bestehen muss. Auch Symbole beliebiger Länge, beispielsweise Wörter oder Zeichenketten sind zulässig. Wird für die Bestimmung der Wahrscheinlichkeit eines Symbols die *n* vorherigen Symbole herangezogen, spricht man von einem *Order-n-Modell*. Die reale Häufigkeitsverteilung der Sequenz wird oft von der des Modells abweichen, daher wird die Bezeichnung $P_M(a_i)$ verwendet, um die Abhängigkeit vom Modell *M* zu betonen.
Entscheidend für die Kompression ist die Wahl eines guten Modells. Je genauer ein Modell der Häufigkeitsverteilung der Sequenz gleicht, desto besser wird die Kompression wirken.

Um ein Maß für die Kompression zu erhalten und um Vergleiche über die Effizienz treffen zu können, führen wir an dieser Stelle den Begriff der Entropie ein. Die Entropie kennzeichnet den Informationsgehalt einer Sequenz.

Definition 13 (Entropie) *Sei S eine Sequenz über dem Eingabealphabet A. Die* Entropie $H_M(S)$ *der Sequenz S unter dem Modell M definieren wir als*

$$H_M(S) = \sum_{a \in A} P(a_i) \log_2 \frac{1}{P_M(a_i)}.$$

Die Entropie wird in der Einheit $[Bits/Symbol]$ *gemessen, da in der Formel nur die relative Häufigkeit benutzt wird und nicht die absolute. Um die Entropie einer Sequenz zu bestimmen, wird diese daher noch mit der Länge der Sequenz multipliziert.*

Die Definition der Entropie ist abhängig vom gewählten Modell. Die minimale Länge eines binären Codes für das Symbol a_i wird dabei durch $\log_2\left(\frac{1}{P_M(a_i)}\right) = -\log_2(P_M(a_i))$ beschrieben. Der Vorfaktor $P(a_i)$, welcher der empirischen Wahrscheinlichkeit entspricht, gibt an, wie häufig wir dieses Symbol in der gegebenen Sequenz auf diese Weise binär kodieren müssen.
Um die eigentliche Datenkompression zu beschreiben, fehlt uns nun noch ein Algorithmus für die Kodierung der Daten und eine Algorithmus, der die Daten aus dem Code wieder herstellt.

Definition 14 (Kodierer&Dekodierer) *Ein Algorithmus, der eine Sequenz kodiert, heißt* Kodierer *oder* Encoder. *Den dazu passenden Algorithmus, der die Sequenz wieder herstellt, bezeichnen wir als* Dekodierer *bzw.* Decoder.

Mit diesen grundlegenden Definitionen haben wir uns ein Werkzeug zurechtgelegt, um die arithmetische Kodierung zu beschreiben.

1.2.2 Arithmetische Kodierung

Die arithmetische Kodierung verwendet, im Gegensatz zur Huffmann Kodierung, keinen Code-Baum zur Generierung des Binärcodes, sondern lediglich eine eindimensionale Tabelle von empirischen Wahrscheinlichkeiten [9]. Sie kodiert stets die komplette Nachricht mit einem einzigen langen Codewort, das den Nachkommastellen einer Zahl aus dem Intervall $[0,1)$ entspricht. Sie hat den Vorteil, dass bestimmten Symbolen auch Codes zugewiesen werden können, die Bruchstücke von Bits lang sind. Der Nachteil daran ist, dass der Code erst dekodiert werden kann, wenn die komplette Nachricht übermittelt wurde.
Die Idee hinter der arithmetischen Kodierung ist also, eine Nachricht zu kodieren, ohne jedem Symbol ein alleinstehendes Codewort zuweisen zu müssen. Die Wahrscheinlichkeiten bzw. die relativen Häufigkeiten der Symbole einer Sequenz

bilden die Grundlage dieser Kodierung. Die summierten Wahrscheinlichkeiten ergeben 1, während die einzelnen Wahrscheinlichkeiten im Intervall $[0,1)$ liegen. Wir betrachten ein Modell M, das uns für m verschiedene Symbole die Wahrscheinlichkeiten $P_M(a_1),\ldots,P_M(a_m)$ der Symbole in der Sequenz der Länge n zuweist. Da die Summe der Wahrscheinlichkeiten 1 ergibt, können wir nun das Intervall $[0,1)$ so einteilen, dass die Größe des i-ten Teilintervalls genau der Wahrscheinlichkeit des i-ten Symbols des Alphabets entspricht, nämlich $P_M(a_i)$.
Zur Illustrierung dieses Sachverhalts betrachten wir beispielsweise das Alphabet $A = \{a,b,c,d\}$ und die Sequenz

$$S = aabacbaada,$$

mit den empirischen Wahrscheinlichkeiten

$$P(a) = 0.6, \quad P(b) = 0.2, \quad P(c) = 0.1, \quad P(d) = 0.1.$$

Wir werden nun Schritt für Schritt das Vorgehen bei der arithmetischen Kodierung analysieren. Wir beginnen mit der Einteilung des Intervalls $[0,1)$ in vier Teilintervalle, wobei die verschiedenen Intervalllängen den empirischen Wahrscheinlichkeiten $P(a),\ldots,P(d)$ entsprechen. Es ist zu beachten, dass die Reihenfolge der Einteilung willkürlich gewählt ist. Wichtig ist, dass der Dekodierer, wenn er die Nachricht empfängt, die gleiche Zuordnung vornimmt.

$$a \to [0.0,0.6) \qquad b \to [0.6,0.8) \qquad c \to [0.8,0.9) \qquad d \to [0.9,1)$$

Das erste Symbol der Sequenz S, das kodiert werden soll, ist $s_1 = a$. Wir wählen also in einem zweiten Schritt das Teilintervall, das wir dem Symbol a zugeordnet haben. $[0.0,0.6)$ dient nun als Ausgangspunkt für eine weitere Intervallschachtelung. Anhand der empirischen Wahrscheinlichkeiten $P(a),\ldots,P(d)$ ergibt sich folgende Einteilung:

$$a \to [0.0,0.36) \quad b \to [0.36,0.48) \quad c \to [0.48,0.54) \quad d \to [0.54,0.6)$$

Das nächste zu kodierende Symbol ist wiederum $s_2 = a$. Wir wählen also als neues Intervall $[0.0,0.36)$ und teilen auch dies, entsprechend den empirischen Wahrscheinlichkeiten der Symbole, in vier Teilintervalle auf. Wir können uns Schreibarbeit sparen, indem wir unser Wissen nutzen, dass das nächste zu kodierende Symbol ein b ist. Wir interessieren uns also nur für das Intervall, das dem Symbol b neu zugeordnet wird. Dies ist in unserem Fall $[0.216,0.288)$. Auf diese Weise wird solange weiter verfahren, bis die komplette Sequenz kodiert wurde.

Wir können das Vorgehen nun verallgemeinern. Wir bezeichnen die obere bzw. die untere Grenze des gesamten Intervall mit *high* (obere Grenze) bzw. mit *low* (un-

tere Grenze). Wenn wir mit der Kompression beginnen, liegen die Anfangswerte bei $low = 0.0$ und $high = 1.0$. Des Weiteren beschreiben wir die Grenzen der Teilintervalle, die sich aus den kommulierten Wahrscheinlichkeiten ergeben, wie folgt:

$$K(a_k) = \sum_{i=1}^{k} P_M(a_i).$$

In Bezug auf unser Besipiel erhalten wir folgende Werte:

$$low = 0.0, \quad high = 1.0, \quad K(a) = 0.6, \quad K(b) = 0.8 \quad K(c) = 0.9 \quad K(d) = 1.0.$$

Im Laufe der Kompression müssen nach jedem Schritt die Werte *low* und *high* angepasst werden. Hingegen bleiben die Werte $K(a_i)$ $(i = 1, \ldots, m)$ konstant und werden verwendet, um die Grenzen *low* und *high* zu bestimmen. Ist das erste zu kodierende Symbol s_1 einer gegebenen Sequenz S gleich dem k-ten Symbol a_k des zugrundeliegenden Alphabets A, sprich $s_1 = a_k$, dann erhalten wir als untere Grenze

$$low := \sum_{i=1}^{k-1} P_M(a_i) = K(a_{k-1})$$

und analog als obere Grenze

$$high := \sum_{i=1}^{k} P_M(a_i) = K(a_k).$$

Da es sich nun bei dem neu berechneten Intervall $[low, high)$ um ein echtes Teilintervall von $[0, 1)$ handelt, müssen wir für die neue Berechnung des Intervalls $[low, high)$ die Grenzen mit einem Ausgleichsfaktor $(high - low)$ verrechnen. Des Weiteren müssen wir den Anfang des Intervall berücksichtigen, indem wir die untere Grenze *low* addieren. Ist das aktuell zu kodierende Symbol einer gegebenen Sequenz S gleich dem j-ten Symbol a_j des zugrunde liegenden Alphabets A, erhalten wir als allgemeine Formeln für die Berechnung von *low* und *high*:

$$low' := low + \sum_{i=1}^{j-1} P_M(a_i) \cdot (high - low) = low + K(a_{j-1}) \cdot (high - low)$$

$$high' := low + \sum_{i=1}^{j} P_M(a_i) \cdot (high - low) = low + K(a_j) \cdot (high - low).$$

In einem letzten Schritt setzen wir

$$low := low'$$
$$high := high',$$

da die Grenzen des alten Intervalls vernachlässigt werden können.
Für unser anfängliches Beispiel ergeben sich folgende Intervalle:

$$
\begin{aligned}
a &\rightarrow [0, 0.6) \\
a &\rightarrow [0, 0.36) \\
b &\rightarrow [0.216, 0.288) \\
a &\rightarrow [0.216, 0.2592) \\
c &\rightarrow [0.25056, 0.25488) \\
b &\rightarrow [0.253152, 0.254016) \\
a &\rightarrow [0.253152, 0.2536704) \\
a &\rightarrow [0.253152, 0.25346304) \\
d &\rightarrow [0.253431936, 0.25346304) \\
a &\rightarrow [0.253431936, 0.2534505984)
\end{aligned}
$$

Für den Code Z wird nun eine Zahl aus dem vom arithmetischen Kodierer ermittlten Intervall $[0.253431936, 0.2534505984)$ gewählt. Da die Kodierung aufgrund des Startintervalls $[0,1)$ eine Zahl kleiner 1 liefert, kann die führende Null vernachlässigt werden. Die binäre Darstellung der Nachkommastellen von $Z = 0.2534332275390625$ ist $Z_{bin} = 0100000011100001_{bin}$. Dies sind 16 Bits im Vergleich zu 20 Bits, die wir ohne arithmetische Kodierung benötigen würden. Die Entropie der Sequenz beträgt $H_M(S) = 1.57$. Multipliziert mit der Länge der Sequenz ergibt das 16 Bits. Wir haben mit der arithmetischen Kodierung unserer Beispielsequenz die minimale Codelänge erreicht.

Die Dekodierung verläuft nach einem ähnlichen Schema, indem wir unser Verfahren rückwärts anwenden. Der Dekodierer empfängt die Zahl $Z := Code(S)$ aus dem Intervall $[0,1)$ und kann mit dieser Information die Originalsequenz S rekonstruieren. Für die Rekonstruktion muss der Dekodierer die Länge der Sequenz und die Wahrscheinlichkeiten $P_M(a), \ldots, P_M(d)$ kennen. Die Einteilung in die Teilintervalle muss der Dekodierer dabei auf die gleiche Weise vornehmen wie der Kodierer. Gehen wir zurück zu unserem Beispiel. Wir erhalten folgende Einteilung:

$$a \rightarrow [0.0, 0.6) \qquad b \rightarrow [0.6, 0.8) \qquad c \rightarrow [0.8, 0.9) \qquad d \rightarrow [0.9, 1)$$

Der Dekodierer prüft nun, in welchem der Teilintervalle die übermittelte Zahl Z liegt, um daraus auf das erste kodierte Symbol zu schließen. Offensichtlich liegt Z in dem Intervall, das dem Symbol a zugeordnet ist. Da das erste rekonstruierte

Symbol ein a ist, basiert die weitere Einteilung auf dem Intervall $[0.0, 0.6)$:

$$a \to [0.0, 0.36) \quad b \to [0.36, 0.48) \quad c \to [0.48, 0.54) \quad d \to [0.54, 0.6).$$

Es wird nun solang auf diese Weise weiter verfahren, bis schließlich alle Symbole dekodiert sind.

Wir verallgemeinern das Verfahren, indem wir, wie bei der Kodierung, Formeln angeben, die die Grenzen *high* und *low* nach jedem Schritt anpassen:

$$\begin{aligned} low' &:= low + K(a_{j-1}) \cdot (high - low) \\ high' &:= low + K(a_j) \cdot (high - low) \\ low &:= low' \\ high &:= high', \end{aligned}$$

dabei ist i so zu wählen, dass

$$low \leq Z \leq high$$

gilt. Das nächste Symbol in der Sequenz ist dann a_i. Nach n Schritten erhalten wir die dekodierte Sequenz.
Für eine Beschreibung der Implementierung der arithmetischen Kodierung, verweisen wir auf [11].

1.2.3 Adaptive arithmetische Kodierung

Bei der adaptiven arithmetischen Kodierung gehen wir von einem deterministischen Modell aus. Wir werden hierbei die empirischen Wahrscheinlichkeiten der Symbole einer Sequenz S verwenden und nach jedem Schritt die absolute Häufigkeit der Symbole aktualisieren. Dies ist sinnvoll, da sich der arithmetische Kodierer an die Häufigkeit der Symbole automatisch anpassen kann. Es sei $S = s_1 s_2 \ldots s_n$ eine Sequenz der Länge n über dem Alphabet $A = \{a_1, \ldots, a_m\}$, $s_i \in A$. Basierend auf der absoluten Häufigkeit $n(a_j)$ der Symbole a_j $(j = 1, \ldots, m)$ in S bestimmen wir die relativen Häufigkeiten,

$$P(a_1), \ldots, P(a_m)$$

dieser Symbole. Nach der arithmetischen Kodierung des Symbols $s_i = a_j$ mit der absoluten Häufigkeit $n(a_j)$, aktualisieren wir diese Häufigkeit,

$$n(a_j) = n(a_j) - 1,$$

und erhalten die dadurch festgelegten aktualisierten relativen Häufigkeiten für die Kodierung des Symbols s_{i+1} in S. Das Verfahren kann allgemein wie folgt dargestellt werden:

Algorithmus 1
Eingabe: $S = s_1 \dots s_n$;
Für $j = 1, \dots, n-1$:

1. *Für* $i = 1, \dots, m$:
   initialisiere $P(a_i) = n(a_i)/(\sum_{i=1}^{m} n(a_i))$;
2. *Kodiere* $s_j = a_i$ *nach dem Prinzip des arithmetischen Kodierers;*
3. *Update:* $n(a_i) = n(a_i) - 1$;

Ende;
Ausgabe: $Z = Code(S)$;

1.2.4 Kontextuelle adaptive arithmetische Kodierung

Da meist keine vollkommene Unabhängigkeit zwischen den Symbolen einer Sequenz besteht, ist es sinnvoll Kontexte der Symbole in die Kodierung mit einzubeziehen. Bei der kontextuellen adaptiven arithmetischen Kodierung wird nicht nur eine Wahrscheinlichkeits- bzw. Häufigkeitstabelle verwendet, sondern Mehrere, zwischen denen, je nach Bedarf, hin- und hergeschaltet werden kann. Die Grundlage zur kontextuellen Kodierung bildet der kausale Kontext.

Definition 15 (Kausale Kontext) *Der* kausale Kontext *wird mit Hilfe von vorangegangenen Symbolen, die bereits kodiert wurden, festgelegt. Über den kausalen Kontext lässt sich eine genauere Aussage über das aktuell zu kodierende Symbol treffen und hat damit einen direkten Einfluss auf die empirische Wahrscheinlichkeit des Symbols.*

Welcher kausale Kontext, d.h. welche vorangegangenen Symbole dabei für die Kodierung von s_i der Sequenz $S = s_1 \dots s_n$ verwendet werden, muss im Vorhinein sorgsam analysiert werden, um ein gutes Kompressionsergebnis zu erzielen. Basierend auf den festgelegten Kontexten erfolgt die Einteilung der Symbole in einheitliche Klassen.

Definition 16 (Kontextklassen) *Basierend auf dem kausalen Kontext erfolgt eine eindeutige Festlegung von* Kontextklassen M_i *($i = 0, \dots, N$). Die Menge aller Kontextklassen M_i fassen wir zusammen zu $\mathcal{M}$.*

Wurden die Kontextklassen festgelegt, können die Symbole der Sequenz S, mit dem entsprechenden Kontext, einer der Klassen zugeordnet werden.

Definition 17 *Es sei $S = s_1 \dots s_n$ eine Sequenz endlicher Länge $|S| = n$ über dem Alphabet $A = \{a_1, \cdots, a_m\}$. Jedes Symbol s_i ($i = 1, \dots, n$) der Sequenz S wird mit Hilfe der kausalen Kontexte einer Kontextklasse $M \in \mathcal{M}$ zugeordnet. Es gilt folgende Abbildungsvorschrift:*

$$S \rightarrow \mathcal{M},\ s_i \mapsto M.$$

Wurde für jedes Symbol $s_i \in A$ ($i = 1, \dots, n$) die Kontextklasse ermittelt, kann schließlich die für die kontextuelle adaptive arithmetische Kodierung notwendige Häufigkeitstabelle bestimmt werden. Dazu benötigen wir die Begriffe der *bedingten absoluten* und der *bedingten relativen Häufigkeit*.

Definition 18 (Bedingte absolute Häufigkeit) *Es sei $S = s_1 \dots s_n$ eine endliche Sequenz der Länge $|S| = n$ über dem Alphabet $A = \{a_1, \cdots, a_m\}$. Die absolute Häufigkeit $n(a_j|M)$ des Symbols $s_i = a_j$ in S, das den Kontext $M \in \mathcal{M}$ besitzt, wird* bedingte absolute Häufigkeit *genannt.*

Aus den bedingten absoluten Häufigkeiten lassen sich die bedingten relativen Häufigkeiten bestimmen.

Definition 19 (Bedingte relative Häufigkeit) *Sei $S = s_1 \dots s_n$ eine endliche Sequenz der Länge $|S| = n$ über dem Alphabet $A = \{a_1, \cdots, a_m\}$ und $n(a_j|M)$ die bedingte absolute Häufigkeit des Symbols $s_i = a_j$ in S, das den Kontext $M \in \mathcal{M}$ besitzt. Dann ist*

$$P(a_j|M) := \frac{n(a_j|M)}{\sum_{k=1}^{m} n(a_k|M)}$$

die bedingte relative Häufigkeit *des Symbols $s_i = a_j$ (in S), mit dem Kontext M.*

Mit Hilfe der bedingten absoluten bzw. relativen Häufigkeiten wir dann die Häufigkeitstabelle erstellt, die der kontextuelle arithmetische Kodierer verwendet, um die Symbole der Sequenz nach dem Prinzip der arithmetischen Kodierung zu verschlüsseln.

Um diesen Sachverhalt besser zu verstehen, besprechen wir an dieser Stelle ein allgemeines Beispiel. Der Einfachheit halber wählen wir das binäre Alphabet $A = \{0, 1\}$. Des Weiteren existieren die zwei Kontextklassen, $\mathcal{M} = \{B, C\}$, die durch die entsprechende kausale Information, bzw. den Kontext, festgelegt werden. Betrachten wir eine Sequenz S, dann wird das Symbol $s_i = a_j \in A$ der Kontextklasse B zugeordnet, wenn das vorangegangene Symbol eine 0 ist. Analog wird ein Symbol der Kontextklasse C zugewiesen, wenn das vorangegangene Symbol eine 1 ist. Mittels dieser Zuweisungen werden nun die bedingten absoluten Häufigkeiten bestimmt, wie oft den Elementen 0 bzw. 1 in S die Klasse B bzw. C zugewiesen wurden. Die bedingten Häufigkeiten bekommen die Bezeichnungen $n(0|B)$,

	B	C	
0	$n_{0,A} = 3$	$n_{0,B} = 3$	6
1	$n_{1,A} = 4$	$n_{1,B} = 3$	7
	7	6	13

Tabelle 1.1: Häufigkeitstabelle

$n(0|C)$, $n(1|B)$ und $n(1|C)$. Beispielsweise gibt $n(0|B)$ an, wie oft das Symbol 0 in S der Klasse B zugeordnet wurde. Wie schon im vorangehenden Abschnitt besprochen, gibt die *Entropie* die minimale Länge des Codes an, die durch die nichtadaptive arithmetische Kodierung produziert wird.

$$\log_2\left(\frac{n(0|B)}{n(0|B)+n(1|B)}\right)+\log_2\left(\frac{n(0|C)}{n(0|C)+n(1|C)}\right)<\log_2\left(\frac{n(0)}{n(0)+n(1)}\right)$$

Die linke Seite dieser Ungleichung beschreibt die *bedingte Entropie* unseres Beispiels. Die obere Schranke auf der rechten Seite stellt die allgemeine Entropie der Daten dar, wenn keine Kontexte in die Kodierung mit einbezogen werden. Die Länge des Codes bei der kontextuellen arithmetischen Kodierung verringert sich erheblich, insbesondere wenn die bedingten relativen Häufigkeiten $n(0|B)/(n(0|B)+n(1|B))$ und $n(0|C)/(n(0|C)+n(1|C))$ stark variieren.

Mit diesen Grundlagen besprechen wir nun ein konkrete Beispiel und analysieren die Sequenz

$$S = 00101110001011,$$

über dem Alphabet $A = \{0,1\}$. Wir bestimmen die bedingten Häufigkeiten von 0 und 1 bezüglich der Kontextklassen B und C und erhalten die in Tabelle 1.1 dargestellte Häufigkeitsverteilung. Insgesamt werden die 13 Zeichen der Sequenz einer der Klassen B oder C zugeordnet. Das erste Symbol $s_1 = 0$, wird, da es keinen Kontext besitzt, nicht in die Zählung mit einbezogen, sondern muss separat kodiert werden. Die bedingte relative Häufigkeit des Symobols $s_2 = 0$, mit dem Kontext B, beträgt beispielsweise $P(0|B) = n(0|B)/(n(0|B)+n(1|B)) = \frac{3}{7}$.

Im allgemeinen Fall sei $S = s_1 s_2, \ldots s_n$, mit $|S| = n$, eine endliche Sequenz über dem Alphabet $A = \{a_1, \ldots, a_m\}$ und $\mathcal{M}$ die Menge der verschiedenen Kontextklassen. In Algorithmus 2 wird das kontextuelle adaptive arithmetische Kodierungverfahren zusammengefasst.

Algorithmus 2 (Kontextuelle adaptive Arithmetische Kodierung)
Eingabe: $S = s_1 \ldots s_n$;

1. *Berechne die Häufigkeitstabelle* $n(a|M)$ *für alle* $a \in A$ *und alle* $M \in \mathcal{M}$*; übertrage dann* $n(a|M)$*;*

2. *Füge die Tabelle dem Bitstream hinzu;*
3. *Für* $i = 1, \ldots, n-1$*:*
   (a) *Ermittle für* $s_i = a_j \in A$ *die Kontextklasse* $M \in \mathcal{M}$*;*
   (b) *Für* $j = 1, \ldots, m$*:*
   initialisiere $P(a_j|M) = n(a_j|M)/(\sum_{j=1}^{m} n(a_j|M))$*;*
   (c) *Kodiere das Symbol* $s_i = a_j$*, entsprechend dem Prinzip der arithmetischen Kodierung;*
   (d) *Update:* $n(a_j|M) = n(a_j|M) - 1$*;*

Ende;
Ausgabe: $Z = Code(S)$*;*

Der adaptive Ansatz hierbei ist, im Gegensatz zur reinen kontextuellen Methode, dass die bedingte absolute Häufigkeit der Symbole $s_i = a_j \in A$ in S, mit dem Kontext $M \in \mathcal{M}$, anfangs übermittelt und nach der Kodierung eines jeden Symbols aktualisiert wird.
Lange Nachrichten, wie bei der Kodierung der Bildinformationen, und zusätzlich kleine relative Häufigkeiten der Symbole erzeugen bei der arithmetischen Kodierung ein sehr kleines Schlussintervall, das den Code $Z = Code(S)$ festlegt. Das bedeutet, dass mehr Nachkommastellen für die Festlegung des Codes benötigt und dadurch die Kosten (in Bits) in die Höhe getrieben werden. Die relativen Häufigkeiten haben somit direkten Einfluss auf die Anzahl der benötigten Bits für die Kompression.

1.3 Zusammenfassung

In diesem einführenden Kapitel konnte gezeigt werden, dass anisotrope Triangulierungen für breite Klassen von Funktionen gegenüber isotropen Triangulierungen eine höhere Approximationsrate erzielen können. Des Weiteren konnte bewiesen werden, dass die somit erzeugten Approximationsraten optimal sind. Waveletapproximationen erzeugen hingegen Approximationsraten, die denen der isotropen Methoden entsprechen.

Mit diesen Ergebnissen und dem Algorithmus für die kontextuelle adaptive arithmetische Kodierung verfügen wir nun über die erforderlichen Werkzeuge, um eine spezielle kontextuelle Bildkompressionsmethode mit Hilfe von Triangulierungen zu beschreiben.

Kapitel 2

Kontextuelles Kompressionsverfahren

Die bestehende Bildkompressionsmethode CAT, die auf der anisotropen Delaunay Triangulierung basiert, wurde im Jahr 2002 in der Arbeit [21] eingeführt und seither immer weiter entwickelt. Für die Generierung der Triangulierung verwendet dieses Verfahren den Adaptive Thinning Algorithmus, der rekursiv nicht-signifikante Pixel aus den Bilddaten entfernt. In den Arbeiten [4] und [8] wurden neue effektivere Bewertungskriterien für die Signifikanz der Pixel eines Bildes analysiert, die letztlich zu dem in [8] und [7] vorgestellten Verfahren führten. In [5] wurde schließlich die kontextuelle Kompression für das Verfahren CAT präsentiert. CAT kann grob in zwei aufeinanderfolgende Arbeitsschritte eingeteilt werden:

- Bildapproximation mit Hilfe der Delaunay Triangulierung
- Kontextuelle Kompression der reduzierten Bilddarstellung

Wir werden diese Schritte in den nächsten Abschnitten beschreiben und im vierten Kapitel auf die numerischen Ergebnisse des Verfahrens in Bezug auf JPEG2000 und das erweiterte Verfahren CAT*, das im vierten Kapitel vorgestellt wird, eingehen.

2.1 Bildapproximation

In Abschnitt 1.1.5 haben wir gesehen, dass die theoretischen Approximationsraten durch anisotrope Triangulierungen dafür sprechen, dass Bilder mit viel Kanteninformation besser durch Triangulierungen als durch Wavelets approximiert werden können. Dieser Abschnitt behandelt eine detaillierte Beschreibung der adaptiven Bildapproximation mit Hilfe der anisotropen Delaunay Triangulierung, die in der bestehenden Bildkompressionsmethode CAT Verwendung findet.

Wir beginnen mit der Vorstellung des Adaptive Thinning Algorithmus, der eine Menge ausgewählter Pixel eines Bildes bereitstellt. Zur Identifizierung dieser Menge, werden wir die Bewertungskriterien der Signifikanz eines Pixels und die linearen Splines über der Delaunay Triangulierung näher analysieren. Darüber hinaus werden wir den finalen Schritt des Bildapproximationsschemas beschreiben, welcher die beste Bildapproximation bereitstellt.

2.1.1 Bilddarstellung

Um die grundlegende Idee des Adaptive Thinning zu erklären, sei X die Menge der äquidistant verteilten Pixel eines digitalen Bildes, deren Positionen ein zweidimensionales rechteckiges Gitter bilden. Daraus lässt sich, in Kombination mit den zugehörigen Grauwerten, eine bivariate diskrete Funktion $f\colon X \to \mathbb{R}$ beschreiben. Wir beschränken uns in dieser Arbeit auf die Kompression von digitalen Bildern, die ausschließlich aus Grauwerten bestehen. Das digitale Bild kann somit als Element $I \in \{0,1,...,2^r-1\}^X$ aufgefasst werden, wobei r die Anzahl der Bits für die Darstellung der Grauwerte der Pixel in X beschreibt.
Wir werden das Bild als eine Funktion über der konvexen Hülle $[X]$ von X betrachten, sodass $[X]$ den rechteckigen Bildbereich bildet. Jedem Pixel in X entspricht demnach ein ebener Gitterpunkt mit ganzzahligen Koordinaten, die in $[X]$ liegen. Das Ziel des Adaptive Thinning Algorithmus ist, eine kleine Teilmenge $Y \subset X$ der signifikanten Pixel des Bildes zu wählen, dessen Delaunay Triangulierungen $\mathcal{D} \equiv \mathcal{D}(Y)$ eine abzählbare endlich-dimensionale Menge $\mathcal{S}^0_{\mathcal{D}}$ an global stetigen, stückweise linearen Funktionen bereitstellt [6].

2.1.2 Adaptive Thinning Algorithmus

Um eine Menge X_n mit n ausgewählten Pixeln zu erhalten, konstruiert der Adaptive Thinning Algorithmus eine Folge von ineinander geschachtelten Teilmengen

$$X_n \subset X_{n+1} \subset ... \subset X_{N-1} \subset X_N = X, \tag{2.1}$$

wobei die Mächtigkeit $|X_p|$ jeder Teilmenge X_p in (2.1) gleich p ist und $N = |X|$ der Anzahl der Pixel in X entspricht [4].
Der Algorithmus entfernt rekursiv ein Pixel der aktuellen Menge auf eine *greedy* Art und Weise. Das bedeutet, dass schrittweise der Folgezustand gewählt wird, der zum Zeitpunkt der Wahl das beste Ergebnis liefert. Die Entscheidung, ob ein Pixel entfernt wird, hängt von dem Grauwert ab, den das Pixel besitzt. In jedem Schritt entspricht das entfernte Pixel einem *weniger signifikantem Pixel*. Der Adaptive Thinning Algorithmus verläuft nach folgendem Schema.

Algorithmus 3
Eingabe: Menge X der Pixel und deren zugehörige Grauwerte $\{f(x)\}_{x\in X}$.

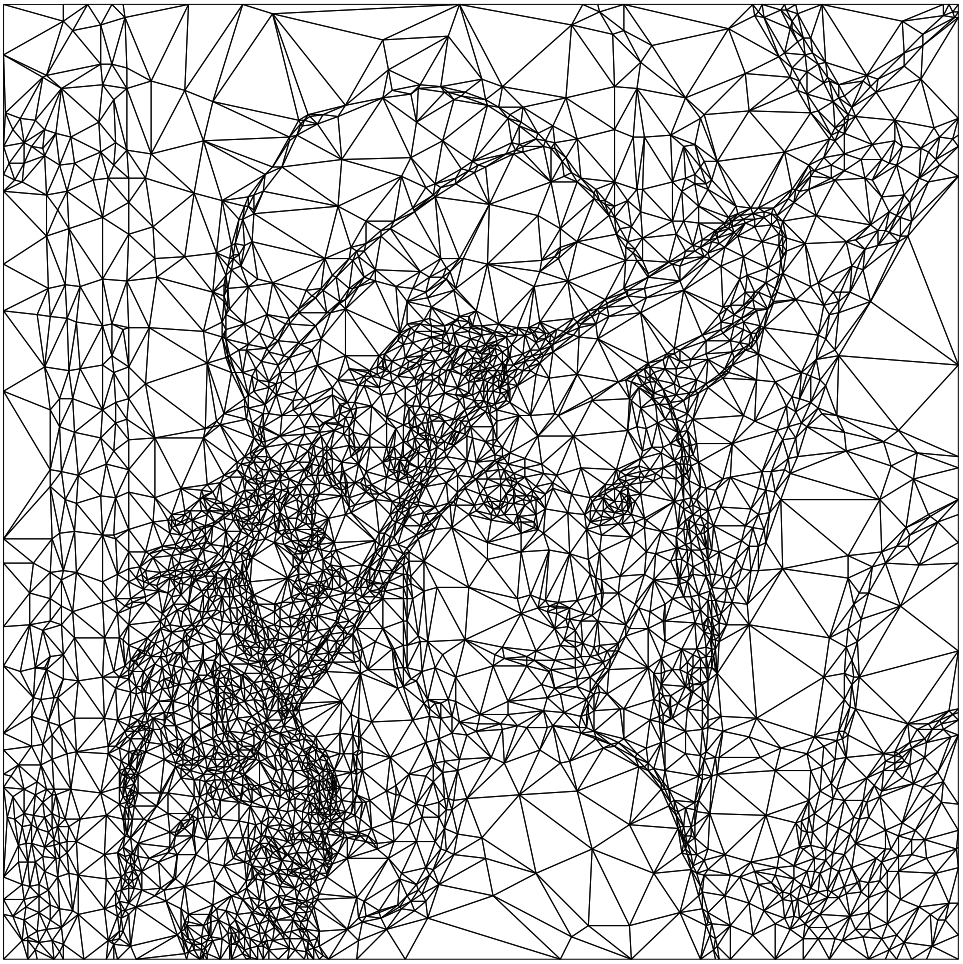


Abbildung 2.1: Lena, eine Delaunay Triangulation

1. *Sei* $X_N = X$;
2. *Für* $k = 1,...,N-n$
   (a) *Finde einen* weniger signifikantes Pixel $x \in X_{N-k+1}$;
   (b) *Setze* $X_{N-k} = X_{N-k+1} \setminus x$;

Ende;
Ausgabe: Teilmenge $Y = X_n \subset X$ *der signifikanten Pixel.*

Um die Adaptive Thinning Strategie zu beschreiben, fehlt nun noch eine Definition, wann ein Pixel ein *weniger signifikantes Pixel* ist, sprich welche Kriterien zur Entfernung nicht-relevanter Pixel herangezogen werden. Dies erfordert eine genauere Diskussion über das Bildapproximationsschema, das in Algorithmus 3

verwendet wird. Für eine Teilmenge $Y = X_p$ in (2.1), wird für die Approximation des Bildes ein linearer Spline über der Delaunay Triangulierung $\mathcal{D}(Y)$ von Y herangezogen, welcher an den Punkten y die Funktionswerte $f(y)$ annimmt. Die Beschreibung der linearen Splines über $\mathcal{D}(Y)$ erfolgt in Abschnitt 2.1.3, als Grundlage für die notwendigen Bewertungskriterien der Signifikanz eines Pixels.
Es ist zu beachten, dass für jedes entfernte Pixel $y \in Y$ die Delaunay Triangulierung $\mathcal{D}(Y \setminus y)$ von $Y \setminus y$ durch ein *lokales Update* berechnet werden kann. Dies folgt aus der Delaunay Eigenschaft, welche impliziert, dass nur die Zelle $C(y)$ von y in $\mathcal{D}(Y)$ bei diesem Update retrianguliert werden muss. Dabei bezeichnet die Zelle $C(y)$ von y die Menge, die von allen Dreiecken aus $\mathcal{D}(Y)$ gebildet wird, die y als Knoten besitzen. Abbildung 2.2 zeigt einen Knoten $y \in \mathcal{D}(Y)$ und die Zelle $C(y)$. Mit $\mathcal{D}(Y)$ von Y erhalten wir die Grundlage für die Konstruktion der linearen Splines.

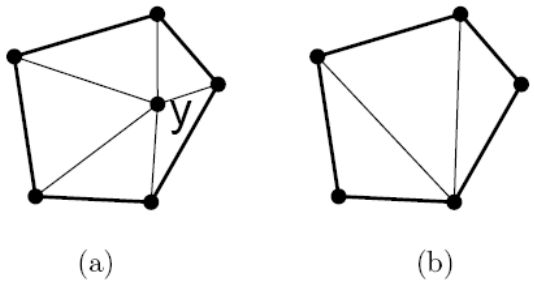


Abbildung 2.2: Entfernung der Ecke $y \in \mathcal{D}(Y)$ und die Delaunay Triangulierung der Zelle $C(y)$. Die fünf Dreiecke der Zelle $C(y)$ in (a) werden bei einem lokalen Update durch die drei Dreiecke in (b) ersetzt.

2.1.3 Lineare Splines über der Delaunay Triangulierung

Es sei $\mathcal{P}_1$ die Menge aller linearen bivariaten Polynome. Für die Delaunay Triangulierung $\mathcal{D} \equiv \mathcal{D}(Y)$ von $Y \subset X$ definieren wir die Menge der stetigen stückweise linearen Splines

$$\mathcal{S}_{\mathcal{D}}^0 := \left\{ f \in \mathcal{C}(Y) : f\big|_T \in \mathcal{P}_1 \right\},$$

die sich aus allen stetigen Funktionen über Y zusammensetzt. Die Restriktion der Funktionen aus $\mathcal{S}_{\mathcal{D}}^0$ auf ein beliebiges Dreieck $T \in \mathcal{D}(Y)$ liegt in $\mathcal{P}_1$. Jedes Element aus $\mathcal{S}_{\mathcal{D}}^0$ bezieht sich auf einen linearen Spline über $\mathcal{D}(Y)$. Für die gegebenen Grauwerte der Pixel aus Y, $\{f(y) : y \in Y\}$, existiert ein eindeutig bestimmter Spline Interpolant $s \in \mathcal{S}_{\mathcal{D}}^0$, der die Gleichung

$$s(y) = f(y)$$

für alle $y \in Y$ erfüllt.
Unter der Voraussetzung, dass die konvexe Hülle $[Y]$ von Y mit der konvexen Hülle $[X]$ von X übereinstimmt, bildet $\mathcal{S}^0_{\mathcal{D}}$ für eine fixe Teilmenge $Y \subset X$ eine Menge von Approximationen für das Bild I. Hierzu ist es notwendig, dass die vier Eck-Pixel von X ebenfalls Elemente der Menge Y sind.
Um nun beurteilen zu können, welches Pixel entfernt werden kann, sodass die Qualität des Bildes nicht beeinträchtigt wird, benötigen wir ein Entscheidungsmaß, das so genannte Signifikanz-Maß.

2.1.4 Signifikanz-Maß

Die Qualität des Bildkompressionsverfahrens wird von dem *Peak Signal to Noise Ratio*,

$$PSNR = 10 * log_{10}\left(\frac{2^r \times 2^r}{\bar{\eta}(Y,X)}\right),$$

in der Einheit dB (Dezibel) gemessen. Für jedes $Y \subset X$ mit der Anzahl $|X|$ der Pixel in X, beschreibt

$$\bar{\eta}(Y,X) = \frac{\eta(Y,X)}{|X|} \tag{2.2}$$

den mittleren quadratischen Fehlers (MSE, *mean square error*) und

$$\eta(Y,X) = \sum_{x \in X} |s(x) - f(x)|^2$$

den quadratischen Fehler der Approximation s bezüglich f [7]. Dabei bezieht sich der lineare Spline Interpolant s auf die Menge Y der ausgewählten Pixel. Gute Bildkompressionen haben typischerweise einen PSNR Wert von 30 dB oder mehr für das rekonstruierte Bild [8]. Wir versuchen eine Teilmenge $Y \subset X$ zu konstruieren, sodass der *Approximationsfehler* $\eta(Y,X)$ möglichst klein ist.
Die Konstruktion einer solchen Menge $Y \subset X$ wird durch Algorithmus 3 mit einer geeigneten Definition eines weniger signifikanten Pixels sichergestellt. Eine intuitive Erklärung gibt die folgende Definition.

Definition 20 *Es sei $Y \subset X$. Ein Pixel $y^* \in Y$ wird als* wenig signifikant *in Y bezeichnet, wenn*

$$\eta(y^*) = \min_{y \in Y} \eta(y),$$

wobei für alle $y \in Y$,

$$\eta(y) = \eta(Y \setminus y, X)$$

die Signifikanz *des Pixels y in Y angibt.*

In dieser Arbeit interessieren wir uns nicht nur für die Signifikanz von einzelnen Pixeln, sondern auch für die Signifikanz von Pixelpaaren. Die nächste Defini-

tion wird diesen Begriff näher beleuchten.

Definition 21 *Es sei $Y \subset X$. Ein Paar $\{y_1^*, y_2^*\} \subset Y$ zweier Pixel in Y heißt* weniger signifikant *in Y, wenn*

$$\eta(y_1^*, y_2^*) = \min_{\{y_1, y_2\} \subset Y} \eta(y_1, y_2),$$

wobei für jedes Pixelpaar $\{y_1, y_2\} \subset Y$,

$$\eta(y_1, y_2) = \eta(Y \setminus \{y_1, y_2\}, X)$$

die Signifikanz *in Y angibt.*

Aus diesen zwei Definitionen ergibt sich nun das Signifikanz-Maß.

Definition 22 *Es sei $Y \subset X$. Ein Pixel $y^* \in Y$ heißt* weniger signifikant *in Y, wenn er einem weniger signifikanten Pixelpaar in Y, $\{y^*, y\} \subset Y$, angehört und die Ungleichung $\eta(y^*) \leq \eta(y)$ erfüllt.*

Um die Qualität der Verteilung der ausgewählten Pixel weiter zu optimieren und damit eine bessere Bildapproximation zu erhalten, wird der in [7] vorgestellte lokale Optimierungsprozess verwendet. Diese lokale Optimierung beruht auf einem iterativen Austausch von Pixelpaaren. Dazu definieren wir, wann Pixel austauschbar sind [14].

Definition 23 *Für jede Menge $Y \subset X$, sei $Z = X \setminus Y$. Ein Pixelpaar $(y, z) \in Y \times Z$, das der Ungleichung*

$$\eta((Y \cup z) \setminus y, X) < \eta(Y, X)$$

genügt, wird als austauschbar *bezeichnet. Eine Teilmenge $Y \subset X$ heißt* lokal optimal *in X, wenn kein austauschbares Pixelpaar $(y, z) \in Y \times Z$ existiert.*

Bei einem Austausch eines beliebigen Pixelpaares $(y, z) \in Y \times Z$, wird der Approximationsfehler $\eta(Y, X)$ zweifelsohne reduziert. Daraus ergibt sich der folgende Algorithmus.

Algorithmus 4 (Austausch)
Eingabe: $Y \subset X$;

1. *Sei $Z = X \setminus Y$;*
2. *Solange Y nicht lokal optimal in X ist:*
   (a) *Lokalisiere ein austauschbares Pixelpaar $(y, z) \in Y \times Z$;*
   (b) *Setze $Y = (Y \setminus y) \cup z$ und $Z = (Z \setminus z) \cup y$;*

Ende;
Ausgabe: $Y \subset X$, lokal optimal in X.

2.1.5 Äquivalentes Signifikanz-Maß

Um die Rechenkosten von Algorithmus 3 zu reduzieren, führen wir das zum Signifikanz-Maß η äquivalente Maß e_δ ein. Dazu bilden wir eine brauchbare Beziehung zwischen dem Signifikanz-Maß für ein Pixel $y \in Y$,

$$e_\delta(y) = \eta(y) - \eta(Y,X),$$

und dem Signifikanz-Maß für ein Pixelpaar $\{y_1,y_2\} \subset Y$,

$$e_\delta(y_1,y_2) = \eta(y_1,y_2) - \eta(Y,X).$$

Jedes Pixelpaar ist dabei entweder eine Kante der Delaunay Triangulierung $\mathcal{D}(Y)$, $[y_1,y_2] \in \mathcal{D}(Y)$, oder die zwei Pixel y_1,y_2 sind in $\mathcal{D}(Y)$ nicht miteinander verbunden. Im letzteren Fall haben die zwei Zellen $C(y_1)$ und $C(y_2)$ kein gemeinsames Dreieck. Es gilt

$$\eta(y_2) - \eta(Y,X) = \eta(y_1,y_2) - \eta(y_1).$$

Daraus folgt

$$e_\delta(y_1,y_2) = \eta(y_1,y_2) - \eta(Y,X).$$

Dies zeigt, dass für jedes Pixelpaar $\{y_1,y_2\} \subset Y$, mit $[y_1,y_2] \notin \mathcal{D}(Y)$, folgendes gilt

$$e_\delta(y_1,y_2) = e_\delta(y_1) + e_\delta(y_2). \tag{2.3}$$

Das Maß der Signifikanzen $\{e_\delta(y_1,y_2) : \{y_1,y_2\} \subset Y\}$ kann, aufgrund der einfachen Darstellung in (2.3), auf die Signifikanz-Maße von $\{e_\delta(y_1,y_2) : [y_1,y_2] \subset Y\}$ und $\{e_\delta(y) : y \in Y\}$ reduziert werden. Der dadurch resultierende Algorithmus hat die Komplexität $O(NlogN)$, wobei $N = |X|$ die Anzahl der Pixel darstellt.

2.1.6 Minimierung des mittleren quadratischen Fehlers

Durch die *Approximation der kleinsten Quadrate* reduzieren wir in einer Nachbearbeitung des Adaptive Thinning weiter den mittleren Fehler. Genauer gesagt, berechnen wir die allgemein *beste Approximation* $s^* \in \mathcal{S}^0_\mathcal{D}$, die der Gleichung

$$\sum_{x\in X} |s^*(x) - f(x)|^2 = \min_{x\in X} \sum_{x\in X} |s(x) - f(x)|^2$$

genügt, für die vom Adaptive Thinning Algorithmus ausgegebene Menge $Y \subset X$ der signifikanten Pixel und die Grauwerte $f(x)$ der Pixel in X. Ist $\mathcal{S}^0_\mathcal{D}$ ein endlichdimensionaler Raum und $Y \subset X$, dann existiert eine solche beste Approximation und ist eindeutig.
Die von der Menge Y und den entsprechenden *optimalen* Grauwerten $f^*(y) =$

$s^*(y)$, $y \in Y$, erzeugte finale Approximation des Bildes I, wird durch den linearen Spline $s^* \in \mathcal{S}^0_{\mathcal{D}}$ wiedergegeben. Die Darstellung des Bildes durch die bestmögliche Approximation kann auch wie folgt gewählt werden:

$$s^*(x) = \sum_{y \in Y} s^*(y)\varphi_y(x) \text{ für } x \in X,$$

wobei die Menge der *Courant Elemente* $\varphi_y \in \mathcal{S}^0_{\mathcal{D}}$, mit dem Knoten $y \in Y$, die allgemeine Lagrange-Basis darstellt.

2.2 Kontextuelle Bildkompression

Bevor wir auf die Kompressionsmethoden in den Abschnitten 2.2.2 und 2.2.3 eingehen, geben wir in Abschnitt 2.2.1 eine zusammenfassende, allgemeine Darstellung der reduzierten Bilddaten an. Darauf erfolgt die genaue Analyse des zweiten Teilschritts des bestehenden Kompressionsverfahrens CAT: die kontextuelle Kompression der reduzierten Bilddarstellung. Diese wird in zwei aufeinanderfolgenden Schritten durchgeführt:

- Kompression der Pixel Positionen
- Kompression der quantisierten Grauwerte

Wir werden dabei speziell auf die Bildung der verwendeten kausalen Kontexte eingehen, die die Grundlage der kontextuellen adaptiven arithmetischen Kodierung bilden und einen großen Einfluss auf die Effizienz der Kodierung und die Qualität des rekonstruierten Bildes haben.

2.2.1 Reduzierte Bilddarstellung

Das mit Hilfe der anisotropen Delaunay Triangulierung approximierte Bild, das durch die beste Approximation s^* gegeben ist, besitzt die reduzierte Bildinformation $\{(y, f^*(y)) : y \in Y\}$. Um eine gute Kompressionsrate zu erzielen, führen wir in einem ersten Schritt eine allgemeine Quantisierung der optimalen Grauwerte durch und erhalten die quantisierten Symbole.

Definition 24 *Es sei $\{(y, f^*(y)) : y \in Y\}$ die reduzierte Bildinformation und $\{f^*(y) : y \in Y\}$ die Menge der optimalen Grauwerte. Die* quantisierten Grauwerte *werden als die Menge*

$$Q_Y := \{Q(f^*(y)) : y \in Y\}$$

definiert, wobei wir für ein $s < r$ den Wertebereich $\{0, 1, \cdots, 2^s - 1\}$ nutzen.

Da die Information über die Topologie der Dreiecke [13], aufgrund der Eindeutigkeit der Delaunay Triangulierung $\mathcal{D}(Y)$, für die Rekonstruktion des Bildes

nicht benötigt wird, beschränkt sich die Kompression der Bilddarstellung auf die Menge

$$\{y, Q(f^*(y)) : y \in Y\} \in I_n^s := \{\{0,1,\cdots,2^s-1\}^Z : Z \subset X \text{ und } |Z| = n\},$$

mit $n = |Y|$. Die Mächtigkeit der Menge I_n^s ist $\binom{|X|}{n} \times 2^{s \times n}$. Unter der Annahme, dass jedes Element in I_n^s die gleiche Auftrittswahrscheinlichkeit besitzt, betragen die theoretischen Kosten der Kodierung der unkomprimierten Daten

$$\log_2(\binom{|X|}{n}) + s \times n \text{ Bits.}$$

Wenn wir den Vorteil, dass mit Hilfe des linearen Slines über der Delaunay Triangulierung die Kanteninformationen eines Bildes erfasst werden, für die Kodierung der reduzierten Bildinformationen $\{y, Q(f^*(y)) : y \in Y\}$ nutzen, lassen sich die Kosten deutlich reduzieren.

2.2.2 Kontexte der Pixel Positionen

Bevor wir auf die in dem bestehenden Verfahren CAT verwendete Methodik der kontextuellen Kompression der Pixel Positionen $\{y : y \in Y\}$ näher eingehen, werden wir wichtige Definitionen einführen, die wir für die Beschreibung des Verfahrens benötigen.

Da wir für die Kodierung der Pixel Positionen lediglich die Information benötigen, ob es sich um einen ausgewählten Knoten handelt oder nicht, wählen wir für die Darstellung der Bilddaten die Repräsentanten 0 und 1. Diese Symbole bilden das der Kodierung zugrundeliegende Alphabet.

Definition 25 *Das Alphabet, das der Kodierung der Pixel Positionen zugrundeliegt, ist definiert durch*

$$\mathcal{A} := \{0,1\}.$$

Dabei stellt 0 *ein entferntes Pixel und* 1 *ein signifikantes Pixel, gemäß des Adaptive Thinning Algorithmus, dar.*

Basierend auf dieser Definition, können wir jedem Pixel $p \in X$ die Eigenschaft zuweisen, ob es sich um ein signifikantes Pixel $p \in Y$ oder um ein entferntes Pixel $p \in X \setminus Y$ handelt. Dies lässt sich wie folgt darstellen.

Definition 26 *Es sei X die Pixelmenge, $Y \subset X$ die Menge der signifikanten Pixel und $\mathcal{A} = \{0,1\}$ das zugrundeliegende Alphabet für die Kodierung der Pixel Posi-*

tionen. Es gilt

$$X \to \mathcal{A}$$
$$p \mapsto a_p = \begin{cases} 1 & ,wenn\ p \in Y, \\ 0 & ,wenn\ p \in X \setminus Y. \end{cases}$$

a_p bezeichnet das Pixel $p \in X$ mit der Eigenschaft $a \in \mathcal{A}$.

Es ist offensichtlich, dass die ausgewählten Pixel nicht vollkommen unabhängig voneinander im Bild liegen. Bei der Kodierung der einzelnen Positionen spielt die Anzahl der ausgewählten bzw. signifikanten Pixel in der Umgebung eine große Rolle. Liegen beispielsweise um dem aktuell zu kodierenden Pixel viele signifikante Pixel, so ist die Wahrscheinlichkeit sehr groß, dass es sich bei dem Aktuellen ebenfalls um ein signifikantes Pixel handelt. Auf diese Eigenschaft werden wir im dritten Kapitel noch genauer eingehen. Damit lassen sich nun die Kontexte der Pixel Positionen festlegen.

Definition 27 (Kontextbox) *Es sei $p \in X$. Die Menge $B(p)$ der kausalen Pixel $q = (q_x, q_y)$, die der Bedingung*

$$\max(|p_x - q_x|, |p_y - q_y|) \leq b,$$

für $b \in \mathbb{N}$, genügen, wird als Kontextbox *des Pixels $p = (p_x, p_y) \in X$ bezeichnet. Für jedes Pixel $p \in X$ sei n_p die Anzahl der signifikanten Pixel in $B(p)$.*

Dabei ist die Anzahl n_p der kausale Kontext, der für die Festlegung der Kontextklassen verwendet wird. Abbildung 2.3 illustriert das genaue Vorgehen für die Bildung der Kontexte.
Wie in Algorithmus 2 beschrieben, werden die bedingten absoluten Häufigkeiten der Symbole 0 und 1 bezüglich der Kontextklassen dem Bitstream hinzugefügt. Dies ist dadurch begründet, dass diese Informationen bei der Dekodierung der Pixel Positionen zur Verfügung stehen müssen. Um damit die Gesamtkosten der Kodierung nicht unnötig in die Höhe zu treiben, wird die Anzahl der Kontextklassen beschränkt.

Definition 28 *Es sei $\mathcal{M} = \{M_0, \ldots, M_{\max-1}\}$ die Menge der Kontextklassen. Dabei bezeichnet*

$$\max := |\mathcal{M}|$$

die maximale Anzahl der Kontextklassen *in $\mathcal{M}$.*

Es ist zu beachten, dass b und max frei wählbare Parameter sind, die zu Beginn der Applikation im Header festgelegt werden müssen. Wir verfügen nun über die nötigen Definitionen, um die Methode der kontextuellen Kompression der Pixel

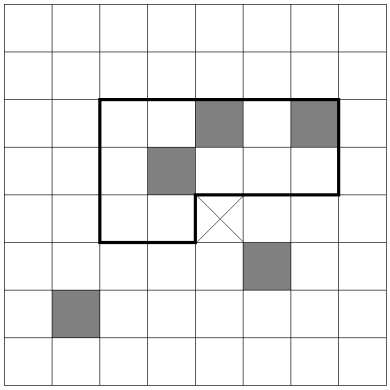


Abbildung 2.3: Kontextbox. Jedes Viereck des Gitters entspricht einem Pixel p; signifikante Pixel sind in grau dargestellt. Das Pixel mit dem Kreuz ist das aktuelle Pixel, das kodiert werden soll. Die Kontextbox für $b = 2$ ist der Bereich, der von der dicken Linie umrandet ist; die Kontextbox beinhaltet $n_p = 3$ signifikante Pixel.

Positionen des Kompressionsverfahrens CAT zu beschreiben.

Wir betrachten ein digitales Bild und legen mit Hilfe der Sequenz S, über dem Alphabet A, die Reihenfolge für die Kodierung der Pixel Positionen fest. Die Sequenz wird spaltenweise gelesen und startet in der linken oberen Bildecke und endet in der rechten unteren Bildecke. Die Größe der Kontextbox und die maximale Anzahl der Kontextklassen werden dem Header des Programms übergeben. Für jedes Symbol der Sequenz, mit dem entsprechenden Pixel $p \in X$, wird die Anzahl der signifikanten Pixel in der Kontextbox notiert. Basierend auf dem kausalen Kontext n_p, erfolgt die Einteilung der Pixel a_p in die Kontextklassen. Die Kompression der Pixel Positionen kann wie folgt dargestellt werden.

Kompression der Pixel Positionen 1
Eingabe: $S = s_1 \dots s_n$, *mit* $n = |X|$, $s_i \in \mathcal{A} = \{0,1\}$ *und* $\mathcal{M} = \{M_0, \cdots, M_{\max-1}\}$;
Für $i = 1, \dots, n$:

1. *Für* $s_i = a_p$ *bestimme* n_p *in* $B(p)$;

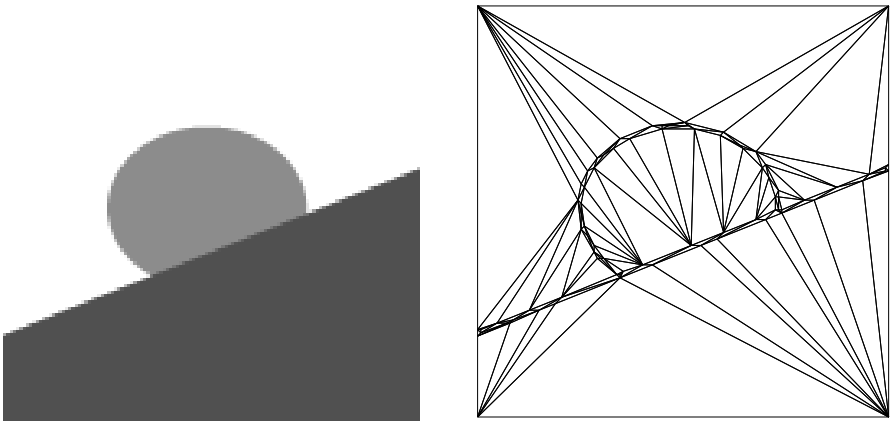
   (a) *Ist* $n_p \geq \max - 1$ *setze* $n_{a_p} := \max - 1$;

   (b) *Ansonsten setze* $n_{a_p} := n_p$;

2. *Ordne* $s_i = a_p \in \mathcal{A}$ *einer Kontextklasse in* $\mathcal{M}$ *zu:*

$$a_p \mapsto M_{n_{a_p}};$$

Ende;
Ausgabe: Häufigkeitstabelle $n(a_p|M)$, *für alle* $a_p \in \mathcal{A}$ *und* $M \in \mathcal{M}$;



(a) Drunken Sunset, Original

(b) Drunken Sunset, Delaunay Triangulierung $\mathcal{D}(Y)$

Abbildung 2.4: Drunken Sunset

Daraufhin kann Algorithmus 2 für die Kodierung der Pixel Positionen angewendet werden.

2.2.3 Kontexte der quantisierten Grauwerte

Bevor wir nun zu der Methode der kontextuellen Kompression der quantisierten Grauwerte $Q_Y := \{Q(f^*(y)) : y \in Y\}$ kommen, die das bestehende Verfahren CAT verwendet, werden wir einige Definitionen einführen, um das Prinzip der Kontextbildung für die Grauwerte beschreiben zu können.

Da die Delaunay Triangulierung $\mathcal{D}(Y)$ der ausgewählten Pixel $Y \subset X$ eindeutig bestimmt ist, können die Kanteninformationen der Dreiecke für die Kontextbildung der quantisierten Grauwerte herangezogen werden. Da die Pixel Positionen zuerst dekodiert werden, stehen diese Kanteninformationen auch dem Dekodierer zur Verfügung. Für stückweise glatte Funktionen liegen die großen, flachen Dreiecke in den glatten Bereichen des Bildes. Die langen, dünnen Dreiecke liegen hingegen dort, wo das Bild Konturen ausweist. Die Lage dieser speziellen Dreiecke wird aufgrund der Konstruktion der Delaunay Triangulierung $\mathcal{D}(Y)$, nach dem Adaptive Thinning, auf diese Weise festgelegt. Aufgrund der Bildstruktur nehmen wir an, dass die gegenüberliegenden Pixel von verhältnismäßig langen Kanten eine geringe Grauwertdifferenz aufweisen, die von eher kurzen Kanten eine Große, wie in Abbildung 2.4 ersichtlich. Die Kanten, die eine große Grauwertvariation aufweisen, liegen entlang der scharfen Kanten des Bildes und sind im Vergleich zu den Kanten in den flachen Bereichen kurz. Unser iteratives Verfahren zur Kodierung der quantisierten Grauwerte $\{Q(f^*(y)) : y \in Y\}$ ist also durch die Annahme motiviert, dass eine überwiegend fallende Funktion bezüglich der Länge der Kanten

vorliegt. Dies lässt sich wie folgt formalisieren.

Definition 29 *Es sei $Y \subset X$ und $y \in Y$. Mit $Y_y \subset Y$ wird die Menge der kausalen Pixel bezüglich $y \in Y$ bezeichnet, die nach einer vordefinierten Reihenfolge bereits kodiert wurden. Weiter sei $\mathcal{D}(Y_y) \subset \mathcal{D}(Y)$ eine Teilmenge der Delaunay Triangulierung, die die kausalen Dreiecke bezüglich y beinhaltet.*

Definition 30 *Für jedes signifikante Pixel $y \in Y \subset X$ wird die Menge*

$$B(y) := \{[y,p], p \in Y_y : [y,p] \in \mathcal{D}(Y_y)\}$$

der kausalen Kanten *der Delaunay Triangulierung $\mathcal{D}(Y)$ definiert, die y als Knotenpunkt besitzen.*

Hiermit lassen sich nun geeignete Definitionen für die Beschreibung der Kantenlängen angeben.

Definition 31 *Für jedes signifikante Pixel $y \in Y \subset X$ wird die Menge*

$$L(y) := \{\|y-p\|_2, [y,p] \in B(y)\}$$

der Längen der kausalen Kanten *der Delaunay Triangulierung $\mathcal{D}(Y)$ bezüglich der Euklidischen Norm definiert, die y als Knotenpunkt besitzen. Dabei bezeichnet*

$$l_y := \max_{[y,p] \in B(y)} \|y-p\|_2 \in L(y)$$

die Länge der längsten kausalen Kante *bezüglich y.*

Für die Kodierung der quantisierten Grauwerte benötigen wir nun noch eine Relation zwischen den Knotenpunkten, die sich entlang der Kanten der Delaunay Triangulierung gegenüberliegen.

Definition 32 *Für jedes signifikante Pixel $y \in Y \subset X$ sei*

$$V(y) := \{Q(f^*(y)) - Q(f^*(p)), [y,p] \in B(y)\}$$

die Menge der Grauwertvariationen *zwischen dem quantisierten Grauwert $Q(f^*(y))$ und den Grauwerten der kausalen Pixel, die y entlang einer gemeinsamen Kante gegenüberliegen.*

Definition 33 *Für alle $y \in Y$ definiert $v_y \in V(y)$ die* Grauwertvariation der längsten kausalen Kante *bezüglich y. Es gilt die Abbildungsvorschrift*

$$Y \to \mathcal{V}, y \mapsto v_y,$$

wobei die Menge aller möglichen Grauwertvariationen v_y, für alle $y \in Y$, zu der geordneten Menge

$$\mathcal{V} := \{v : v \in [\min_{y \in Y} v_y, \max_{y \in Y} v_y]\}$$

zusammengefasst wird. $\mathcal{V}$ bildet das zugrundeliegende Alphabet für die Kodierung der Grauwerte.

Uns liegen nun die nötigen Definitionen vor, um die Methode der kontextuellen Kompression der quantisierten Grauwerte des Kompressionsverfahrens CAT anzugeben.

Wir betrachten eine Sequenz S, die die Reihenfolge der Kodierung festlegt. Die Sequenz wird spaltenweise gelesen. Sie startet in der linken oberen Bildecke und endet in der rechten unteren Bildecke. Jedem Symbol der Sequenz entspricht ein signifikantes Pixel $y \in Y$ und damit ein Symbol des Alphabets $\mathcal{V}$, dessen Elemente im Vorhinein gemäß Definition 33 ermittelt werden. Für jedes Symbol der Sequenz, mit entsprechendem signifikanten Pixel y, wird die längste kausale Kante ermittelt. Die Einteilung der Grauwertvariation des Pixels y in die Kontextklassen, wird durch den kausalen Kontext l_y, die Länge der längsten Kante, festgelegt.
Die Kompression der quantisierten Grauwerte erfolgt nach der im Folgenden dargestellten Methode.

Kompression der quantisierten Grauwerte 1
Eingabe: $S = s_1 \dots s_N$, mit $N = |Y|$, $s_i \in \mathcal{V}$ und $\mathcal{M} = \{M_0, M_1, M_2\}$;

1. *Bestimme das Alphabet $\mathcal{V}$ gemäß Definition 33;*
2. *Für $i = 1, \dots, N$:*
   (a) *Für $s_i = v_y$ bestimme die längste kausale Kante bezüglich y, mit der Länge l_y;*
   (b) *Ordne $s_i = v_y \in \mathcal{V}$ einer Kontextklasse in $\mathcal{M}$ zu:*

$$\mathcal{V} \to \mathcal{M}$$
$$v_y \mapsto \begin{cases} M_0 & ,\text{wenn } l_y < 5 \\ M_1 & ,\text{wenn } 5 \leq l_y < 8 \\ M_2 & ,\text{sonst} \end{cases};$$

Ausgabe: Häufigkeitstabelle $n(v_y|M_j)$, für alle $v_y \in \mathcal{V}$ und $M_j \in \mathcal{M}$;

Daraufhin kann Algorithmus 2 für die Kodierung der quantisierten Grauwerte angewendet werden.

2.3 Rekonstruktion des Bildes beim Dekodierer

Nachdem das Bild den Adaptive Thinning Algorithmus durchlaufen hat und die Informationsmenge $\{(y, Q(f^*(y))) : y \in Y\}$ gemäß der kontextuellen adaptiven arithmetischen Kodierung komprimiert wurde, müssen diese Informationen wieder dekodiert werden, um das verschlüsselte Bild beim Dekodierer rekonstruieren zu können. Erreichen die verschlüsselten Informationen der Menge $\{(y, Q(f^*(y))) : y \in Y\}$ den Dekodierer, wird das komprimierte Bild in vier Schritten rekonstruiert.

- Die kodierten Informationen der Menge $\{y : y \in Y\}$ werden mit Hilfe der kontextuellen adaptiven arithmetischen Kodierung dekodiert. Daraus ergibt sich die Menge Y der Pixel Positionen. Die *eindeutige* Delaunay Triangulierung $\mathcal{D}(Y)$ der Menge Y wird generiert.

- Die verschlüsselte Information der Menge Q_Y wird mit Hilfe der kontextuellen adaptiven arithmetischen Kodierung dekodiert. Daraus ergibt sich die Menge $\tilde{Q}_Y$. Diese Menge wird requantisiert und wir erhalten die Menge $\{\tilde{f}(y) : y \in Y\}$.

- Der eindeutige lineare Spline-Interpolant $s \in \mathcal{S}^0_{\mathcal{D}}$, der die Gleichung $s(y) = \tilde{f}(y)$ für alle $y \in Y$ erfüllt, wird konstruiert.

- Das rekonstruierte Bild wird mittels der Informationsmenge $\tilde{I} = \{(x, s(x)) : x \in X\}$ wiedergegeben.

2.4 Zusammenfassung

Die lineare Splineapproximation $s* \in \mathcal{S}^0_{\mathcal{D}}$, mit der zugehörigen Menge Y, repräsentiert das Bild. Das Ziel der Konstruktion von Y, durch den Adaptive Thinning Algorithmus, und somit von $\mathcal{S}^0_{\mathcal{D}}$ ist, den resultierenden mittleren quadratischen Fehler zu minimieren. Dieses Ziel wird durch die bestmögliche Approximation s^* des Bildes I, entlang aller Spline Funktionen aus $\mathcal{S}^0_{\mathcal{D}}$, erreicht. Da es sich bei der Wahl der optimalen Teilmenge $Y^* \subset X$ (aus allen Teilmengen $Y \subset X$ der gleichen Mächtigkeit) um ein NP-hartes Optimierungsproblem handelt, kann der Adaptive Thinning Algorithmus als eine sehr effiziente Methode zum Auffinden guter Bildapproximationen aus einem großen Angebot an Delaunay Triangulierungen betrachtet werden [5]. Die resultierenden reduzierten Bilddaten $\{y, f^*(y) : y \in Y\}$ werden gemäß den Methoden in den Abschnitten 2.2.2 und 2.2.3 komprimiert. Die Dekodierung und Rekonstruktion des Bildes erfolgt gemäß Abschnitt 2.3.
Die numerischen Ergebnisse und Auswertungen der resultierenden Kosten des Kompressionsverfahren CAT folgen in Kaptitel 4.2 in Bezug auf JPEG2000 und das erweiterte Verfahren CAT*, das wir im vierten Kapitel entwickeln werden.

Kapitel 3

Untersuchungen neuer Ansätze

Um den resultierenden Speicherplatzbedarf aus dem bestehenden Kompressionsverfahren CAT weiter zu minimieren und die Übertragungsgeschwindigkeit zu erhöhen, untersuchen wir Erweiterungen der bestehenden Kompressionsmethode und neue Ansätze der Kontextbildung. Dazu gehen wir im ersten Abschnitt dieses Kapitels auf die Motivation für die Bildung neuer Kontexte ein und geben einige Eigenschaften der adaptiven Delaunay Triangulierung an. Die Analyse der neuen Methoden und die statistischen Auswertungen der Kompression der Pixel Positionen und der zugehörigen Grauwerte erfolgt in den Abschnitten 3.2 und 3.3. Es handelt sich hierbei um komplett neu entwickelte Methoden, die in dieser Arbeit untersucht und implementiert wurden. Im letzten Abschnitt dieses Kapitels werden wir die Ergebnisse der erzielten Kostenreduzierung in Bezug auf verschiedene Testbilder und im Hinblick auf die Ergebnisse der Methoden in CAT betrachten und analysieren.

3.1 Motivation und Eigenschaften

Die theoretischen Untersuchungen in Abschnitt 1.1.5 haben gezeigt, dass Anisotropie optimale Approximationsraten liefert. Die mit Hilfe des Adaptive Thinning Algorithmus erzeugten Triangulierungen weisen dementsprechend eine starke Heterogenität bezüglich der dargestellten Dreiecke auf. Im Hinblick auf die Kompression eines Bildes sind solche Strukturen relevant. Wir werden die Kosten der Kodierung reduzieren, indem wir diese Heterogenität berücksichtigen, um in den nächsten Abschnitten sinnvolle Kontexte zum einen für die Pixel Positionen und zum anderen für die Grauwerte festzulegen.

Dazu betrachten wir das ausgedünnte Bild *Lena* (Abbildung 3.1) und erkennen, dass der Adaptive Thinning Algorithmus, zur Minimierung des mittleren quadratischen Fehlers $\eta(Y;X)$, bestimmte Cluster bildet, in denen es Anhäufungen der

Abbildung 3.1: Signifikante Pixel nach Adaptive Thinning

signifikanten Pixel gibt, beispielsweise entlang scharfer Kanten. Doch nicht nur entlang scharfer Kanten beobachten wir diese Anhäufungen. Auch in Bereichen, in denen das Bild viele Informationen aufweist, liegen die ausgewählten Pixel nah zusammen. In den Arealen, in denen wir keine großen Grauwertvariationen auffinden, sind die signifikanten Pixel eher spärlich gesät. Dies wirkt sich auf die Form und die Größe der Dreiecke der anisotropen Delaunay Triangulierung aus. Wie schon im vorangegangenen Kapitel erwähnt, liegen für stückweise glatte Funktionen die großen und flachen Dreiecke in den glatten Bereichen des Bildes. Hingegen liegen die langen und dünnen Dreiecke dort, wo das Bild Konturen ausweist. Gemäß den Regeln des Adaptive Thinning Algorithmus kann ein Bild grob in verschiedene Bereiche eingeteilt werden. Es existieren drei Kategorien, wie in Abbildung 3.2 dargestellt.

1. *Flache Bereiche*, in denen nicht viele verschiedene Farbinformationen vor-

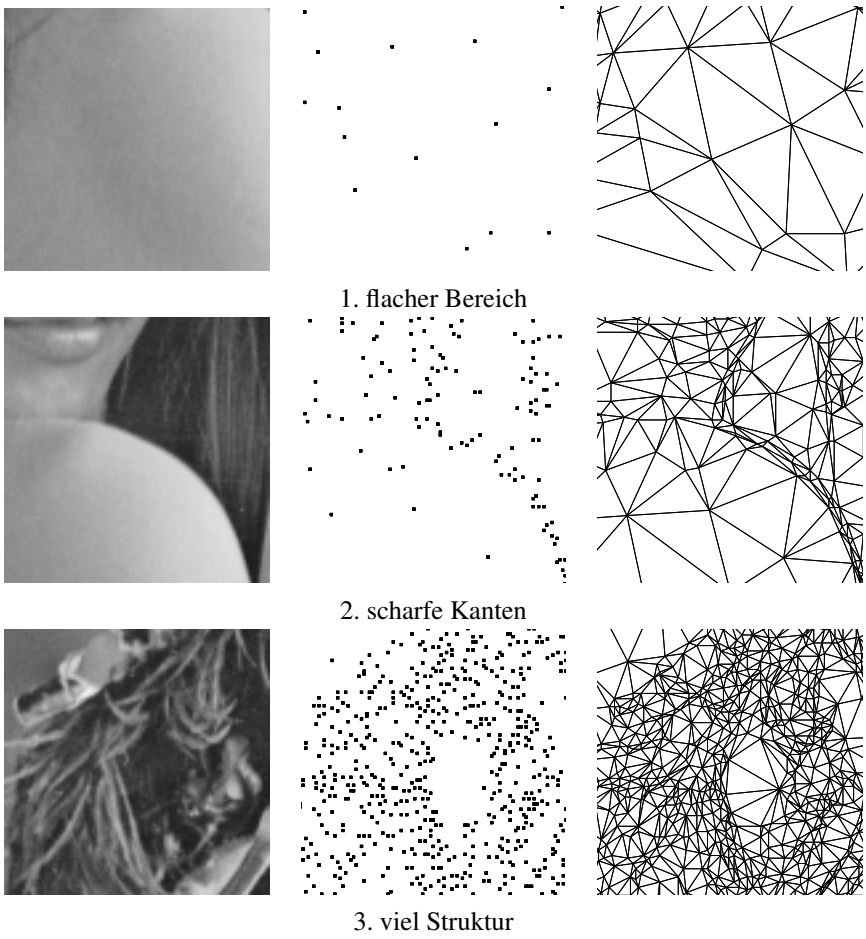


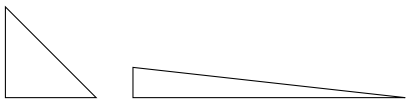
Abbildung 3.2: Kategorien der Bildbereiche mit den ausgewählten Pixeln und der zugehörigen Delaunay Triangulierung; 1. flacher Bereich: die Dreiecksform ist groß und flach, 2. scharfe Kanten: die Dreiecksform ist dünn und lang 3. viel Struktur: die Dreiecksform ist klein und flach aber auch dünn und lang

liegen. Hier wurden mit dem Adaptive Thinning Algorithmus viele Pixel entfernt. Die ausgewählten Pixel liegen eher weit auseinander und besitzen nur leicht variierende Grauwerte. Die Dreiecke haben eine große und flache Form.

2. *Bereiche entlang scharfer Kanten*, in denen viele verschiedene Farbinformationen existieren. Die ausgewählten Pixel liegen speziell in der Nähe der

Kanten nahe beieinander und besitzen eine hohe Grauwertvariation. Die Dreiecksform ist dünn und lang.

3. *Bereiche mit viel Struktur*, in denen viele verschiedene Farbinformationen vorliegen. In diesen Bereichen liegen viele signifikante Pixel sehr nahe zusammen und unterscheiden sich stark in ihren Grauwerten. Die Dreiecke sind eher klein und haben flache, aber auch lange und dünne Formen.



Diese herausstechenden Eigenschaften werden wir insbesondere für die Bildung der Kontexte für die Grauwertkodierung nutzen.

3.2 Kontexte der Pixel Positionen

Bevor wir uns im Folgenden mit der Bildung möglicher Kontexte bezüglich der Pixel Positionen beschäftigen und dazu verschiedene Ansätze analysieren, werden wir eine Methodik für die Berechnung des Speicherplatzbedarfs und somit für die Kosten der Kodierung der Pixel Positionen beschreiben.

3.2.1 Theoretische Kosten der Pixel Positionen

Um eine Aussage über die Qualität der neu entwickelten Methoden zu treffen, müssen wir eine Möglichkeit finden, die theoretische Anzahl der Bits, die für die Speicherung der Pixel Positionen benötigt werden, zu berechnen. Die *kombinatorischen Kosten* der Kodierung, wenn der Kontext außer Acht gelassen wird, werden wie folgt definiert.

Definition 34 *Es sei $n = |X|$ die Anzahl der Pixel in X und $p = |Y|$ die Anzahl der ausgewählten Pixel in Y.*

$$K := \log_2 \binom{n}{p} = \log_2 \left(\frac{n!}{p!(n!-p!)} \right)$$

bezeichnet die kombinatorischen Kosten *der Kodierung, wenn der Kontext außer Acht gelassen wird.*

Da n und p meist große Werte annehmen, kann bei der praktischen Umsetzung in unserer Applikation die Fakultät nicht mehr berechnet werden. Wir behelfen

uns daher einer numerischen Methode zur Approximation der kombinatorischen Kosten, bei der wir die Stirling Formel

$$n! \approx \sqrt{2\pi n}\left(\frac{n}{e}\right)$$

für die Näherung der Fakultät verwenden. Die *approximierten kombinatorischen Kosten* ergeben sich wie folgt.

Korollar 1 *Es sei $n = |X|$ die Anzahl der Pixel in X und $p = |Y|$ die Anzahl der ausgewählten Pixel in Y. Daraus ergeben sich die* approximierten kombinatorischen Kosten

$$\tilde{K} =: (n-p+0,5)\log\left(\frac{n}{n-p}\right) + (p+0,5)\log\left(\frac{n}{p}\right) - 0,5\frac{\log(2\pi n)}{\log(2.0)},$$

für die Kodierung der Pixel Positionen.

Wenn wir den Kontext in die Kodierung mit einbeziehen, werden die kombinatorischen Kosten für jede Kontextklasse einzeln bestimmt. Dazu benötigen wir die bedingte absolute Häufigkeit der signifikanten Pixel, die der Kontextklasse $M_i \in \mathcal{M}$ zugeordnet wurden und die absolute Häufigkeit der Pixel, die insgesamt diesen Kontext besitzen. Dies lässt sich wie folgt formalisieren.

Definition 35 *Es sei $\mathcal{M}$ die Menge der Kontextklassen. Weiter sei $n(1|M_i)$ die absolute Häufigkeit der signifikanten Pixel, die der Kontextklasse $M_i \in \mathcal{M}$ zugeordnet wurden und $n(M_i)$ die absolute Häufigkeit der Pixel die insgesamt den Kontext M_i besitzen. Für alle $M_i \in \mathcal{M}$ definiert*

$$K_{M_i} := \log_2 \binom{n(M_i)}{n(1|M_i)}$$

die kombinatorischen Kosten der Kontextklasse M_i. *Durch die Addition der Kosten für die einzelnen Kontextklassen, ergeben sich die theoretischen* kombinatorischen Gesamtkosten

$$K_{\mathcal{M}} := \sum_{M_i \in \mathcal{M}} K_{M_i},$$

wenn der Kontext in die Kodierung mit einbezogen wird.

Für die Abschätzung der Kosten erhalten wir, unter der Verwendung der in Korollar 1 angegebenen Approximation, folgendes Ergebnis.

Korollar 2 *Die* approximierten kombinatorischen Kosten $\tilde{K}_{M_i}$ *für die Klassen $M_i \in \mathcal{M}$ ergeben sich durch die Verwendung von Korollar 1, mit $n = n(M_i)$ und $p = n(1|M_i)$. Durch die Addition der einzelnen approximierten kombinatorischen Kos-*

ten der Kontextklassen lassen sich die approximierten kombinatorischen Gesamtkosten

$$\tilde{K}_{\mathcal{M}} = \sum_{M_i \in \mathcal{M}} \tilde{K}_{M_i}.$$

für die Kodierung der Pixel Positionen berechnen.

Wir haben uns hiermit ein nützliches Werkzeug für die Abschätzung der kombinatorischen Kosten der Kodierung der Pixel Positionen zurechtgelegt und können nun die neu entwickelten Kontexte bezüglich ihrer Effizienz untersuchen.

3.2.2 Neue Ansätze / Methodik / Statistische Experimente

Wir beginnen nun mit der Analyse der neuen Ansätze für die Bildung der Kontexte für die Pixel Positionen und werden im Folgenden neue Kompressionsmethoden entwickeln, um den Speicherplatzbedarf in Bezug auf die aktuelle Methode des Kompressionsverfahrens CAT aus Abschnitt 2.2.2 zu reduzieren. Wir benötigen wiederum eine vordefinierte Reihenfolge, in der wir das Bild scannen. Wir beginnen in den folgenden Kompressionsmethoden immer in der linken oberen Ecke des Pixel-Gitters und arbeiten uns zeilenweise in die rechte untere Ecke vor, siehe Abbildung 3.3. Die Pixel, die am Rand des Bildes liegen und keine vollständige Kontextbox besitzen, werden dabei vernachlässigt und separat kodiert.

Definition 36 *Es sei X die Menge der Pixel und $Y \subset X$ die Menge der ausgewählten Pixel.*

$$X' \subset X \text{ mit } X' = \{p \in X : \exists B(p)\}$$

bezeichnet die Menge der Pixel, die einen vollständigen Kontext besitzen. Weiter bezeichnet $Y' \subset X'$ die Menge der ausgewählten Pixel, für die eine vollständige Kontextbox konstruiert werden kann.

In unserem ersten Ansatz wählen wir die Methode aus Abschnitt 2.2.2 und ändern lediglich die Reihenfolge, in der die Bilddaten kodiert werden. Die Methode lässt sich wie folgt beschreiben.

Kompressionsmethode 1
Eingabe: $S = s_1 \dots s_n$, *mit* $n = |X|$, $s_i \in \mathcal{A} = \{0,1\}$ *und* $\mathcal{M} = \{M_0, \dots, M_{\max -1}\}$;
Für $i = 1, \dots, n$:

1. *Für* $s_i = a_p$ *bestimme* n_p *in* $B(p)$;
   (a) *Ist* $n_p \geq \max -1$ *setze* $n_{a_p} := \max -1$;
   (b) *Ansonsten setze* $n_{a_p} := n_p$;
2. *Ordne* $s_i = a_p \in \mathcal{A}$ *einer Kontextklasse in* $\mathcal{M}$ *zu:*

$$a_p \mapsto M_{n_{a_p}};$$

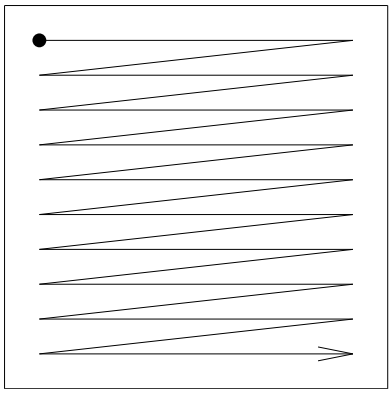


Abbildung 3.3: Orientierung der Sequenz S für die neu entwickelten Methoden

Ende;
Ausgabe: Häufigkeitstabelle $n(a_p|M)$, für alle $a_p \in \mathcal{A}$ und $M \in \mathcal{M}$;

Daraufhin kann Algorithmus 2 für die kontextuelle adaptive arithmetische Kodierung der Pixel Positionen angewendet und die verschlüsselten Daten als Bitstream gespeichert werden.
Tabelle 3.1 zeigt uns die zugehörige zweidimensionale Häufigkeitsverteilung der Pixel mit den bedingten absoluten und relativen Häufigkeiten bezüglich der qualitativen Variable $\mathcal{M}$ und den Ausprägungen M_i ($i = 0, \ldots, \max - 1$) für das Test-

M_i	$n(1\|M_i)$	$n(0\|M_i)$	$n(M_i)$	$P(1\|M_i)$	$\tilde{K}_{M_i}$
M_0	312	49839	50151	0.0062	2730
M_1	473	39997	40470	0.0117	3709
M_2	394	26884	27278	0.0144	2967
M_3	443	17922	18365	0.0241	3006
M_4	508	13766	14274	0.0356	3159
M_5	540	11517	12057	0.0448	3175
M_6	534	9709	10243	0.0521	3020
M_7	520	8946	9466	0.0549	2900
M_8	549	7745	8294	0.0662	2910
M_9	496	6404	6900	0.0719	2567
M_{10}	512	5499	6011	0.0852	2520
M_{11}	6654	40822	47476	0.1402	27749

Tabelle 3.1: Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$; Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 1

bild *Lena*, mit 512×512 Pixeln und $|Y| = 12144$ signifikantne Pixeln. Des Weiteren sind die approximierten kombinatorischen Kosten für die jeweiligen Kontextklassen dargestellt. Die geschätzten kombinatorischen Gesamtkosten betragen $\tilde{K}_{\mathcal{M}} = 60412$ Bits.
Es ist ersichtlich, dass ein Anstieg von n_{a_p} zu einer größeren bedingte relative Häufigkeit $P(1|M_i)$ führt. Des Weiteren kann beobachtet werden, dass mit einer steigenden bedingten relativen Häufigkeit $P(1|M_i)$ eine Senkung der kombinatorischen Kosten erzielt werden kann.

Die Wahl der neuen Kontexte in unserer nächsten Methode ist motiviert durch die Erkenntnis, dass in den Bereichen mit großen Pixel-Anhäufungen die relative Häufigkeit der signifikanten Pixel sehr groß ist. Dies gilt theoretisch auch an den Rändern dieser Areale. Es tritt jedoch oft die Situation auf, dass die Anzahl n_p der signifikanten Pixel in den Randbereichen dieser Anhäufungen eher klein ist und somit die bedingte relative Häufigkeit der signifikanten Pixel sinkt, siehe Abbildung 3.6. Die Herangehensweise aus dem ersten Ansatz kann nun erweitert werden, indem man unterscheidet, ob die relevanten Pixel der Kontextbox $B(p)$ in der unmittelbaren Umgebung des Pixels $p \in X'$ liegen. So könnte die Situation an den Randbereichen von Pixel-Anhäufungen abgefangen werden. Dies kann wie folgt formuliert werden.

Definition 37 *Es sei $p \in X'$. Die Menge $B_\beta(p)$ der kausalen Pixel $q = (q_x, q_y)$, die der Bedingung*

$$\max(|p_x - q_x|, |p_y - q_y|) \leq \beta, \text{ für } \beta < b$$

genügen, wird als Kontextbox der Größe β *des Pixels $p = (p_x, p_y) \in X'$ bezeichnet. Für jedes Pixel $p \in X'$ sei $n_{\beta,p}$ die Anzahl der signifikanten Pixel in $B_\beta(p)$.*

Mit dieser Definition lässt sich die neue Methode für die Kompression der Pixel Positionen wie folgt beschreiben:

Kompressionsmethode 2
Eingabe: $S = s_1 \dots s_n$, mit $n = |X|$, $s_i \in \mathcal{A} = \{0,1\}$ und $\mathcal{M} = \{M_0, \dots, M_{\max}\}$;
Für $i = 1, \dots, n$:

1. *Für $s_i = a_p$ bestimme n_p in $B(p)$ und $n_{\beta,p}$ in $B_\beta(p)$;*
   (a) *Ist $n_p \geq \max - 1$ setze $n_{a_p} := \max - 1$;*
   (b) *Ansonsten setze $n_{a_p} := n_p$;*
2. *Ordne $s_i = a_p \in \mathcal{A}$ einer Kontextklasse in $\mathcal{M}$ zu:*

$$a_p \mapsto \begin{cases} M_0 & \text{,wenn } n_{a_p} = 0, \\ M_{n_{a_p}} & \text{,wenn } n_{a_p} \geq 1 \text{ und } n_{\beta,p} < n_{a_p}, \\ M_{n_{a_p}+1} & \text{,wenn } n_{a_p} \geq 1 \text{ und } n_{\beta,p} \geq n_{a_p}; \end{cases}$$

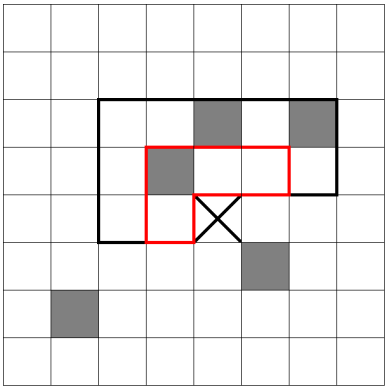


Abbildung 3.4: Kontextbox Methode 2. Jedes Viereck des Gitters entspricht einem Pixel p; signifikante Pixel sind in grau dargestellt. Das Pixel mit dem Kreuz ist das aktuelle Pixel, das kodiert werden soll. Die Kontextbox für $b = 2$ ist der Bereich, der von der dicken schwarzen Linie umrandet ist; die Kontextbox beinhaltet $n_p = 3$ signifikante Pixel. Die Kontextbox der unmittelbaren Umgebung für $\beta = 1$ ist der Bereich, der von der dicken roten Linie umrandet ist; die Kontextbox beinhaltet $n_{\beta,p} = 1$ signifikante Pixel.

Ende;
Ausgabe: Häufigkeitstabelle $n(a_p|M)$, *für alle* $a_p \in \mathcal{A}$ *und* $M \in \mathcal{M}$*;*

Daraufhin kann Algorithmus 2 für die kontextuelle adaptive arithmetische Kodierung der Pixel Positionen angewendet und die verschlüsselten Daten als Bit-

M_i	$n(1\|M_i)$	$n(0\|M_i)$	$n(M_i)$	$P(1\|M_i)$	$\tilde{K}_{M_i}$
M_0	312	49839	50151	0.0062	2730
M_1	397	31341	31738	0.0125	3073
M_2	452	34479	34931	0.0129	3477
M_3	454	18794	19248	0.0236	3096
M_4	515	13914	14429	0.0357	3200
M_5	539	11553	12092	0.0446	3173
M_6	535	9713	10248	0.0522	3024
M_7	520	8947	9467	0.0549	2900
M_8	549	7745	8294	0.0662	2910
M_9	496	6404	6900	0.0719	2567
M_{10}	512	5499	6011	0.0852	2520
M_{11}	6615	40669	47284	0.1399	27605
M_{12}	39	153	192	0.2031	136

Tabelle 3.2: Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$; Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 2

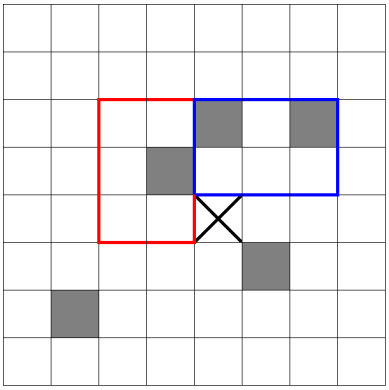


Abbildung 3.5: Kontextbox Methode 3. Jedes Viereck des Gitters entspricht einem Pixel p; signifikante Pixel sind in grau dargestellt. Das Pixel mit dem Kreuz ist das aktuelle Pixel, das kodiert werden soll. Die Kontextbox für $b = 2$ wird in zwei Hälften eingeteilt,$B_r(p)$ ist blau umrandet und $B_l(p)$ ist rot umrandet; die zwei Hälften beinhalten $n_{r,p} = 2$ und $n_{l,p} = 1$ signifikante Pixel.

stream gespeichert werden.
Tabelle 3.2 zeigt die Häufigkeitsverteilung der signifikanten und der entfernten Pixel für $\beta = 4$ und die approximierten Kosten für die jeweiligen Kontextklassen. Die approximierten Gesamtkosten liegen bei $\tilde{K}_{\mathcal{M}} = 60412$ Bits. Dies ist keine Verbesserung zu unserer ersten Methode. Wir werden jedoch in Abschnitt 3.4.1 sehen, dass mit dieser Methode gute Ergebnisse, im Vergleich zu der bestehenden Methode, erzielt werden.

Eine weitere Möglichkeit, eine genauere Aussage über die Pixel an den Rändern von Pixel-Anhäufungen zu treffen ist, die Kontextbox des aktuellen Pixels in zwei Hälften zu teilen. Liegen die ausgewählten Pixel in der linken Hälfte der Kontextbox, nehmen wir an, dass sich der aktuelle Pixel rechts von einer Pixel-Anhäufung befindet und umgekehrt.

Definition 38 *Es sei $B(p)$ die Kontextbox des Pixels $p \in X'$. Die Menge*

$$B_r(p) \subset B(p)$$

wird als die Menge der kausalen Pixel von p bezeichnet, die in der rechten Hälfte von $B(p)$ liegen. Die Anzahl der signifikanten Pixel in $B_r(p)$ ist $n_{r,p}$. Entsprechend wird die Menge

$$B_l(p) = B(p) \setminus B_r(p)$$

als die Menge der kausalen Pixel von p definiert, die in der linken Hälfte von $B(p)$ liegen. Die Anzahl der signifikanten Pixel in $B_l(p)$ ist $n_{l,p}$.

Mit dieser Definition können wir nun eine weitere Kompressionsmethode für die Pixel Positionen darstellen.

Kompressionsmethode 3
Eingabe: $S = s_1 \dots s_n$, *mit* $n = |X|$, $s_i \in \mathcal{A} = \{0,1\}$ *und* $\mathcal{M} = \{M_0, \dots, M_{\max}\}$;
Für $i = 1, \dots, n$:

1. *Für* $s_i = a_p$ *bestimme* n_p *in* $B(p)$, $n_{l,p}$ *in* $B_l(p)$ *und* $n_{r,p}$ *in* $B_r(p)$;
   (a) *Ist* $n_p \geq \max - 1$ *setze* $n_{a_p} := \max - 1$;
   (b) *Ansonsten setze* $n_{a_p} := n_p$;
2. *Ordne* $s_i = a_p \in \mathcal{A}$ *einer Kontextklasse in* $\mathcal{M}$ *zu:*

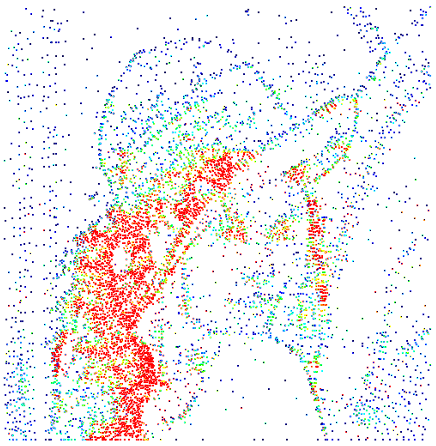
$$a_p \mapsto \begin{cases} M_{n_{a_p}} & ,\text{wenn } n_{a_p} \geq 2 \text{ und } n_{r,p} < n_p \text{ und } n_{l,p} < n_{a_p}, \\ M_{n_{a_p}+1} & ,\text{wenn } n_{a_p} \geq 2 \text{ und } n_{r,p} \geq n_p \text{ oder } n_{l,p} \geq n_{a_p}, \\ M_{n_{a_p}} & ,\text{sonst}; \end{cases}$$

Ende;
Ausgabe: Häufigkeitstabelle $n(a_p|M)$, *für alle* $a_p \in \mathcal{A}$ *und* $M \in \mathcal{M}$;

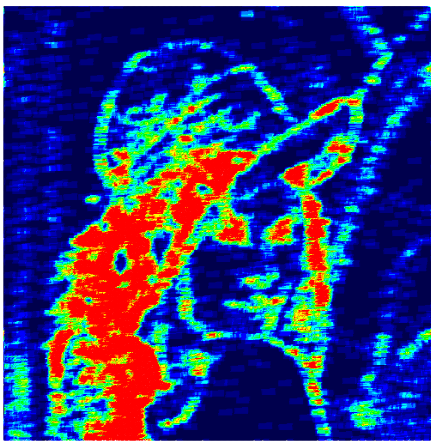
Daraufhin kann Algorithmus 2 für die kontextuelle adaptive arithmetische Kodierung der Pixel Positionen angewendet und die verschlüsselten Daten als Bitstream gespeichert werden.

M_i	$n(1\|M_i)$	$n(0\|M_i)$	$n(M_i)$	$P(1\|M_i)$	$\tilde{K}_{M_i}$
M_0	312	49839	50151	0.0062	2730
M_1	473	39997	40470	0.0117	3709
M_2	189	14379	14568	0.0130	1451
M_3	514	25599	26113	0.0197	3641
M_4	554	15878	16432	0.0337	3489
M_5	554	12565	13119	0.0422	3306
M_6	568	10303	10871	0.0522	3211
M_7	533	9367	9900	0.0538	2989
M_8	559	7982	8541	0.0654	2972
M_9	498	6507	7005	0.0711	2586
M_{10}	520	5576	6096	0.0853	2558
M_{11}	2864	22146	25010	0.1145	12833
M_{12}	3797	18912	22709	0.1672	14782

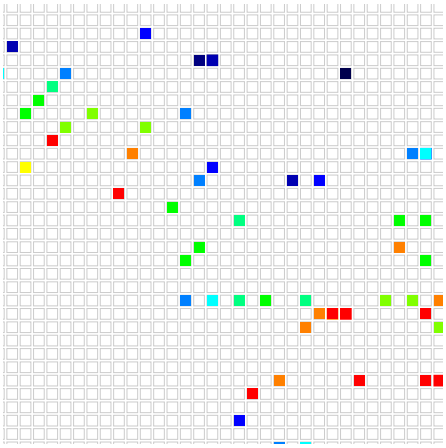
Tabelle 3.3: Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$, Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 3



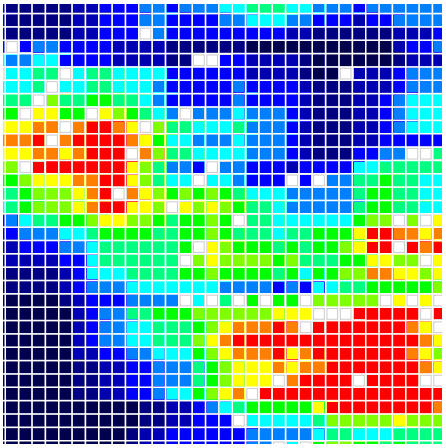
(a) Einteilung in die Kontextklassen; signifikante Pixel, Y



(b) Einteilung in die Kontextklassen; entfernte Pixel, $X \setminus Y$



(c) Einteilung in die Kontextklassen entlang einer Kante; signifikante Pixel, Y



(d) Einteilung in die Kontextklassen entlang einer Kante; entfernte Pixel, $X \setminus Y$

(e) Farbskala: $M_{\max-1} = \text{rot}, \ldots, M_0 = \text{schwarz}$

Abbildung 3.6: Lena 512x512, farbige Darstellung der Einteilung in die Kontextklassen für $\max = 12$ und $b = 8$.

Die Häufigkeitsverteilung ist in Tabelle 3.3 dargestellt mit den jeweiligen approximierten Kosten für die Kontextklassen. Die approximierten Geasmtkosten betragen 60256 Bits.

Ziehen wir nicht nur die Anzahl der signifikanten Pixel der Kontextbox für die Festlegung der Kontextklassen heran, sondern auch die Kontextklassen der signifikanten Pixel in der Kontextbox, erhalten wir für unseren Testfall eine deutliche Reduzierung der Kosten. Die Idee dahinter ist, dass wenn die signifikanten Pixel in

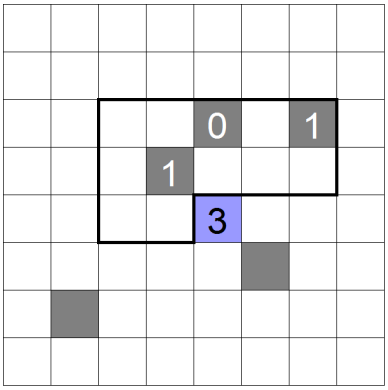


Abbildung 3.7: Kontextbox Methode 4. Jedes Viereck des Gitters entspricht einem Pixel p; signifikante Pixel sind in grau dargestellt. Das blaue Pixel ist das aktuelle Pixel, das kodiert werden soll. Die Kontextbox für $b = 2$ ist der Bereich, der von der dicken schwarzen Linie umrandet ist; die Kontextbox beinhaltet $n_p = 3$ signifikante Pixel; für jedes signifikante Pixel in der Box ist die Anzahl der signifikanten Pixel für dessen Kontextbox notiert.

der Kontextbox einer höheren Klasse zugeordnet wurden, der aktuelle Pixel in der Nähe einer größeren Anhäufung von Pixeln liegt. Diesen Fall finden wir beispielsweise entlang scharfer Kanten des Bildes vor, wie in Abbildung 3.6 dargestellt. Für die Beschreibung des Verfahrens benötigen wir folgende Definition.

Definition 39 *Es sei $B(p)$ die Menge der kausalen Pixel von $p \in X'$ und n_p die Anzahl der signifikanten Pixel in $B(p)$. Die Menge*

$$B^*(p) := Y \cap B(p).$$

wird definiert als die Menge der signifikanten Pixel in $B(p)$. *Es gilt $n_p = |B^*(p)|$.*

Die neue Methode der Kompression der Pixel Positionen lässt sich nun wie folgt formulieren.

Kompressionsmethode 4
Eingabe: $S = s_1 \dots s_n$, mit $n = |X|$, $s_i \in \mathcal{A} = \{0,1\}$ und $\mathcal{M} = \{M_0, \dots, M_{\max}\}$;
Für $i = 1, \dots, n$:

1. *Für $s_i = a_p$ bestimme n_p in $B(p)$ und n_q in $B(q)$, $\forall q \in B^*(p)$;*

   (a) *Ist $n_p \geq \max - 1$ setze $n_{a_p} := \max - 1$;*

   (b) *Ansonsten setze $n_{a_p} := n_p$;*

2. *Ordne $s_i = a_p \in \mathcal{A}$ einer Kontextklasse in $\mathcal{M}$ zu:*

$$a_p \mapsto \begin{cases} M_0 & ,\text{wenn } n_{a_p} = 0, \\ M_{n_{a_p}} & ,\text{wenn } n_{a_p} \geq 1 \text{ und } \exists q \in B^*(p) : n_q < n_{a_p}, \\ M_{n_{a_p}+1} & ,\text{wenn } n_{a_p} \geq 1 \text{ und } \forall q \in B^*(p) : n_q \geq n_{a_p}; \end{cases}$$

Ende;
Ausgabe: Häufigkeitstabelle $n(a_p|M)$, für alle $a_p \in \mathcal{A}$ und $M \in \mathcal{M}$;

Daraufhin kann Algorithmus 2 für die kontextuelle adaptive arithmetische Kodierung der Pixel Positionen angewendet und die verschlüsselten Daten als Bitstream gespeichert werden.
Dies führt zu der in Tabelle 3.4 dargestellten Häufigkeitsverteilung der Symbole in $\mathcal{A} = \{0,1\}$, mit den geschätzten Kosten für die Kontextklassen und den approximierten Gesamtkosten $\tilde{K}_{\mathcal{M}} = 60065$ Bits.

Beziehen wir nicht nur die signifikanten Pixel in den Kontext mit ein, sondern auch die Kontextklassen aller kausalen Pixel der Kontextbox, erhalten wir bezüglich Kompressionsmethode 1 eine Kostenreduzierung um ein paar Bits. Für die Beschreibung dieser Methode benötigen wir eine weiter Definition.

M_i	$n(1\|M_i)$	$n(0\|M_i)$	$n(M_i)$	$P(1\|M_i)$	$\tilde{K}_{M_i}$
M_0	312	49839	50151	0.0062	2730
M_1	135	19346	19481	0.0069	1158
M_2	554	39208	39762	0.0139	4203
M_3	463	21254	21717	0.0213	3225
M_4	518	15018	15536	0.0333	3270
M_5	562	12791	13353	0.0421	3356
M_6	560	10470	11030	0.0508	3189
M_7	542	9399	9941	0.0545	3029
M_8	550	8063	8613	0.0639	2945
M_9	529	6757	7286	0.0726	2731
M_{10}	492	5573	6065	0.0811	2457
M_{11}	1431	13476	14907	0.0960	6794
M_{12}	5287	27856	33143	0.1595	20978

Tabelle 3.4: Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$, Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 4

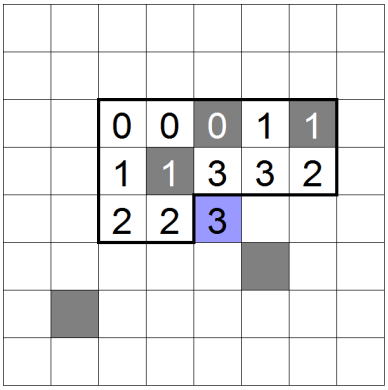


Abbildung 3.8: Kontextbox Methode 5. Jedes Viereck des Gitters entspricht einem Pixel p; signifikante Pixel sind in grau dargestellt. Das blaue Pixel ist das aktuelle Pixel, das kodiert werden soll. Die Kontextbox für $b = 2$ ist der Bereich, der von der dicken schwarzen Linie umrandet ist; die Kontextbox beinhaltet $n_p = 3$ signifikante Pixel; für jedes Pixel in $B(p)$ ist die Anzahl der signifikanten Pixel für dessen Kontextbox notiert.

Definition 40 *Es sei $B(p)$ die Menge der kausalen Pixel von $p \in X'$.*

$$B_\alpha(p) \subset B(p)$$

bezeichnet eine beliebige Teilmenge von $B(p)$, die der Bedingung

$$|B_\alpha(p)| \geq \alpha \cdot |B(p)|,$$

für ein festgelegtes $\alpha \in [0,1]$, genügt.

Wir legen einen Wert für α fest und verfügen nun über die nötigen Hilfsmittel, um die neue Methode für die Kompression der Pixel Positionen zu beschreiben.

Kompressionsmethode 5
Eingabe: $S = s_1 \ldots s_n$, mit $n = |X|$, $s_i \in \mathcal{A} = \{0,1\}$ und $\mathcal{M} = \{M_0, \ldots, M_{\max}\}$;
Für $i = 1, \ldots, n$:

1. *Für $s_i = a_p$ bestimme n_p in $B(p)$ und n_q in $B(q)$, $\forall q \in B(p)$;*

   (a) *Ist $n_p \geq \max - 1$ setze $n_{a_p} := \max - 1$;*

   (b) *Ansonsten setze $n_{a_p} := n_p$;*

2. *Ordne $s_i = a_p \in \mathcal{A}$ einer Kontextklasse in $\mathcal{M}$ zu:*

$$a_p \mapsto \begin{cases} M_{n_{a_p}} & ,\text{wenn } \nexists B_\alpha(p) \subseteq B(p) \text{ ,sodass } \forall q \in B_\alpha(p) : n_q > n_{a_p}, \\ M_{n_{a_p}+1} & ,\text{wenn } \exists B_\alpha(p) \subseteq B(p) \text{ ,sodass } \forall q \in B_\alpha(p) : n_q > n_{a_p}; \end{cases}$$

Ende;
Ausgabe: Häufigkeitstabelle $n(a_p|M)$, für alle $a_p \in \mathcal{A}$ und $M \in \mathcal{M}$;

Daraufhin kann Algorithmus 2 für die kontextuelle adaptive arithmetische Kodierung der Pixel Positionen angewendet und die verschlüsselten Daten als Bitstream gespeichert werden.
Für $\alpha = 0,3$ erhalten wir die in der Tabelle 3.5 dargestellte Häufigkeitsverteilung mit den geschätzten Kosten für die Kontextklassen. Die approximierten Gesamtkosten betragen für diese Methode $\tilde{K}_{\mathcal{M}} = 60352$ Bits.

In einem letzten Ansatz wollen wir versuchen, die Kompressionsmethoden 4 und 5 zu kombinieren, um eine weitere Ersparnis in den Kosten zu erzielen. Mit Hilfe der Definitionen für die genannten Methoden und der Festlegung von $\alpha \in [0,1]$ können wir die neue Methode für die Kompression der Pixel Positionen darstellen.

Kompressionsmethode 6
Eingabe: $S = s_1 \dots s_n$, mit $n = |X|$, $s_i \in \mathcal{A} = \{0,1\}$ und $\mathcal{M} = \{M_0, \dots, M_{\max}\}$;
Für $i = 1, \dots, n$:

1. *Für $s_i = a_p$ bestimme n_p in $B(p)$ und n_q in $B(q)$, $\forall q \in B(p)$;*
   (a) *Ist $n_p \geq \max - 1$ setze $n_{a_p} := \max - 1$;*
   (b) *Ansonsten setze $n_{a_p} := n_p$;*

M_i	$n(1\|M_i)$	$n(0\|M_i)$	$n(M_i)$	$P(1\|M_i)$	$\tilde{K}_{M_i}$
M_0	125	25713	25838	0.0048	1137
M_1	345	45797	46142	0.0075	2927
M_2	467	33185	33652	0.0139	3545
M_3	408	20973	21381	0.0191	2908
M_4	484	14780	15264	0.0317	3091
M_5	507	12926	13433	0.0377	3109
M_6	539	10711	11250	0.0479	3116
M_7	570	9705	10275	0.0555	3171
M_8	503	8528	9031	0.0557	2795
M_9	550	7003	7553	0.0728	2837
M_{10}	490	5933	6423	0.0763	2493
M_{11}	6947	43796	50743	0.1369	29224

Tabelle 3.5: Lena (512*x*512), $|Y| = 12144$, $b = 8$, max $= 12$, Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 5

2. *Ordne $s_i = a_p \in \mathcal{A}$ einer Kontextklasse in $\mathcal{M}$ zu:*

   (a) *Für $n_{a_p} = 0$ verwende die Vorschrift der* Kompressionsmethode 5*;*

   (b) *Ansonsten verwende für die Einteilung in die Kontextklassen* Kompressionsmethode 4*;*

Ende;
Ausgabe: Häufigkeitstabelle $n(a_p|M)$, für alle $a_p \in \mathcal{A}$ und $M \in \mathcal{M}$;

Daraufhin kann Algorithmus 2 für die kontextuelle adaptive arithmetische Kodierung der Pixel Positionen angewendet und die verschlüsselten Daten als Bitstream gespeichert werden.
Dadurch erhalten wir die in der Tabelle 3.6 dargestellte Häufigkeitsverteilung. Die approximierten Gesamtkosten betragen für diese Methode $\tilde{K}_{\mathcal{M}} = 60054$ Bits. Durch die Verteilung der Pixel mit $n_p = 0$ auf die zwei Kontextklassen M_0 und M_1 mit Hilfe von *Kompressionsmethode 5*, gewinnen wir bezüglich der *Kompressionsmethode 4* in diesem Testfall 11 Bits.

M_i	$n(1\|M_i)$	$n(0\|M_i)$	$n(M_i)$	$P(1\|M_i)$	$\tilde{K}_{M_i}$
M_0	125	25713	25838	0.0048	1137
M_1	322	43472	43794	0.0074	2740
M_2	554	39208	39762	0.0139	4203
M_3	463	21254	21717	0.0213	3225
M_4	518	15018	15536	0.0333	3270
M_5	562	12791	13353	0.0421	3356
M_6	560	10470	11030	0.0508	3189
M_7	542	9399	9941	0.0545	3029
M_8	550	8063	8613	0.0639	2945
M_9	529	6757	7286	0.0726	2731
M_{10}	492	5573	6065	0.0811	2457
M_{11}	1431	13476	14907	0.0960	6794
M_{12}	5287	27856	33143	0.1595	20978

Tabelle 3.6: Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$, Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 6

In Abschnitt 3.4.1 folgt die Auswertung und die Analyse der verschiedenen Methoden anhand von vier Testbildern und unterschiedlichen Werten für $|Y|$ bezüglich der bestehenden Methode des Bildkompressionsverfahrens CAT.

3.3 Kontexte der Grauwerte

In den folgenden Abschnitten beschäftigen wir uns mit der Analyse verschiedener Ansätze für die Bildung der Kontexte bezüglich der quantisierten Grauwerte Q_Y. Wir werden versuchen gewisse Regelmäßigkeiten in den ausgedünnten Bilddaten, insbesondere in den Triangulierungen zu identifizieren, um diese sinnvoll als Kontexte für die Grauwertkodierung zu nutzen. Um die Qualität der neuen Methoden zu beurteilen, werden wir auch für die Grauwerte ein Verfahren beschreiben, um die kombinatorischen Kosten für die Kodierung der Grauwerte zu berechnen.

3.3.1 Theoretische Kosten der Grauwerte

Wie auch schon bei den Pixel Positionen wird uns die im Folgenden eingeführte Approximation der kombinatorischen Kosten helfen, eine Abschätzung für die Kosten der Grauwertkodierung zu erhalten. Im Unterschied zu den Pixel Positionen besteht das zugrundeliegende Alphabet für die kontextuelle Kodierung der Grauwerte nicht nur aus den zwei Symbolen $\{0,1\}$, sondern aus allen quantisierten Grauwerten, bzw. Grauwertvariationen, $v \in \mathcal{V} = \{v_1, \ldots, v_m\}$. Lassen wir den Kontext außer Acht, erhalten wir für die *kombinatorischen Kosten* folgende Definition.

Definition 41 *Es sei $\mathcal{V}$ die Menge aller vorkommenden* Grauwerte *und $n(v_i)$ die Anzahl der Pixel, die den Grauwert $v_i \in \mathcal{V}$ besitzen. Daraus ergeben sich die* kombinatorischen Kosten

$$K = \log_2 \binom{n}{n(v_1), n(v_2), \ldots, n(v_m)} = \log_2 \left(\frac{n!}{n(v_1)! \cdot n(v_2)! \cdots n(v_m)!} \right)$$

Bits, mit $n = |Y|$ und $\sum_{i=1}^{m} v_i = n$.

Da n und $n(v_i)$ $(i = 1, \ldots, m)$ meist große Werte annehmen, behelfen wir uns wieder einer numerischen Methode zur Approximation der kombinatorischen Kosten, bei der wir erneut die Stirling Formel für die Näherung der Fakultät verwenden. Die *approximierten kombinatorischen Kosten* lassen sich mit Hilfe des folgenden Algorithmus berechnen.

Algorithmus 5
Eingabe: $n = |Y|$, $m = |\mathcal{V}|$, $\tilde{K} = 0$;
Für $i = 1, \ldots, m-1$:

1. *Setze* $p = n(v_i)$ *und berechne*
   $Kosten = (n - p + 0{,}5) \log \left(\frac{n}{n-p} \right) + (p + 0{,}5) \log \left(\frac{n}{p} \right) - 0{,}5 \frac{\log(2\pi n)}{\log(2.0)}$;
2. *Aktualisiere* $\tilde{K} = \tilde{K} + Kosten$;
3. *Aktualisiere* $n = n - p$;

Ende;
Ausgabe: approximierte kombinatorische Kosten $\tilde{K}$.

Wenn wir den Kontext in die Kodierung mit einbeziehen, werden die kombinatorischen Kosten für jede Kontextklasse einzeln bestimmt. Dazu benötigen wir die bedingten absoluten Häufigkeiten der *Grauwertvariation*, die der Kontextklasse $M_j \in \mathcal{M}$ zugeordnet wurden und die absoluten Häufigkeiten der Grauwertvariationen, die insgesamt diesen Kontext besitzen. Die kombinatorischen Kosten für die Kontextklassen ergeben sich wie folgt.

Definition 42 *Es sei* $\mathcal{M}$ *die Menge der Kontextklassen und* $\mathcal{V}$ *die Menge der Grauwertvariationen. Weiter sei* $n(v_i|M_j)$ *die bedingte absolute Häufigkeit der Grauwertvariation* $v_i \in \mathcal{V}$*, die der Kontextklasse* $M_j \in \mathcal{M}$ *zugeordnet wurde und*

$$n(M_j) = \sum_{i=1}^{m} n(v_i|M_j)$$

die bedingte absolute Häufigkeit der Grauwertvariationen, die diesen Kontext besitzen. Für alle $M_j \in \mathcal{M}$ *bezeichnet*

$$K_{M_j} := \log_2 \binom{n(M_j)}{n(v_1|M_j), \ldots, n(v_m|M_j)}$$

die kombinatorischen Kosten der Kontextklasse M_j*. Durch die Addition der Kosten für die einzelnen Kontextklassen, ergeben sich die theoretischen* kombinatorischen Gesamtkosten*:*

$$K_{\mathcal{M}} = \sum_{M_j \in \mathcal{V}}^{n-1} K_{M_j}.$$

Für die Abschätzung der Kosten erhalten wir, unter Verwendung der in Algorithmus 5 angegebenen Approximation, folgendes Korollar.

Korollar 3 *Die* approximierten kombinatorischen Kosten $\tilde{K}_{M_j}$ *für die Klassen* $M_j \in \mathcal{M}$ *ergeben sich unter Verwendung von Algorithmus 5, mit* $n = n(M_j)$ *und* $n(v_i) = n(v_i|M_j)$*. Durch die Addition der einzelnen approximierten kombinatorischen Kosten* $\tilde{K}_{M_j}$ *lassen sich die* approximierten kombinatorischen Gesamtkosten *für die Kodierung der Grauwerte berechnen:*

$$\tilde{K}_{\mathcal{M}} = \sum_{M_j \in \mathcal{M}} \tilde{K}_{M_j}.$$

Anhand der approximierten kombinatorischen Kosten können wir die Effizienz der im Folgenden entwickelten Kompressionsmethoden messen.

3.3.2 Neue Ansätze / Methodik / Statistische Experimente

Wir werden nun neue Ansätze für die Bildung der Kontexte der quantisierten Grauwerte Q_Y diskutieren und neue Kompressionsmethoden entwickeln, um den Speicherplatzbedarf in Bezug auf die bestehende Methode aus Abschnitt 2.2.3 zu reduzieren. Um sinnvolle Kontexte für die kontextuelle adaptive arithmetische Kodierung beschreiben zu können, überlegen wir uns, welche Informationen durch die Pixelmenge $\{y, y \in Y\}$ und speziell durch die dadurch festgelegte Delaunay Triangulierung $\mathcal{D}(Y)$ von Y zur Verfügung gestellt werden. Dazu nutzen wir die in Abschnitt 3.1 dargestellten Eigenschaften. Aus diesen Darstellungen lassen sich zwei weitere wichtige Beschaffenheiten der Delaunay Triangulierung ableiten:

1. Es besteht ein Zusammenhang zwischen der Grauwertvariation und der Länge der Kante zweier sich gegenüberliegenden Pixel. Wie wir bereits in Abschnitt 2.2.3 analysiert haben, liegt eine überwiegend fallende Funktion der Grauwertvariation bezüglich der Länge der Kanten vor.

2. Die Lage der Pixel zueinander liefert uns Informationen über die Beschaffenheit der Dreiecke (dünnes, langes, flaches Dreieck) und somit über den *aspect ratio*. Je größer der *aspect ratio*, desto dünner und länger ist die Form des Dreiecks. Das bedeutet, dass wir in den Bereichen mit viel Struktur oder entlang scharfer Kanten einen größeren *aspect ratio* erwarten als in den flachen Bereichen.

In einem ersten Versuch analysieren wir, wie bei der in Abschnitt 2.2.3 vorgestellten Methode, die Zusammenhänge zwischen den Grauwerten und den Abständen der Pixel zueinander. Die Sequenz wird zeilenweise gelesen. Sie startet in der linken oberen Ecke des Bildes und endet in der rechten unteren Ecke. Der Unterschied zu der bestehenden Methode liegt in der Orientierung der Sequenz. Die Kompressionsmethode kann wie folgt formuliert werden.

Kompressionsmethode 1
Eingabe: $S = s_1 \dots s_N$, $N = |Y|$ *mit* $s_i \in \mathcal{V}$ *und* $\mathcal{M} = \{M_0, M_1, M_2\}$;

1. *Bestimme das Alphabet* $\mathcal{V}$ *gemäß Definition 33;*
2. *Für* $i = 1, \dots, N$*:*
   (a) *Für* $s_i = v_y$ *bestimme die längste kausale Kante bezüglich* y*, mit der Länge* l_y*;*
   (b) *Ordne* $v_y \in \mathcal{V}$ *einer Kontextklasse in* $\mathcal{M}$ *zu:*

$$v_y \mapsto \begin{cases} M_0 & \text{,wenn } l_y < 5 \\ M_1 & \text{,wenn } 5 \leq l_y < 8 \\ M_2 & \text{,sonst} \end{cases};$$

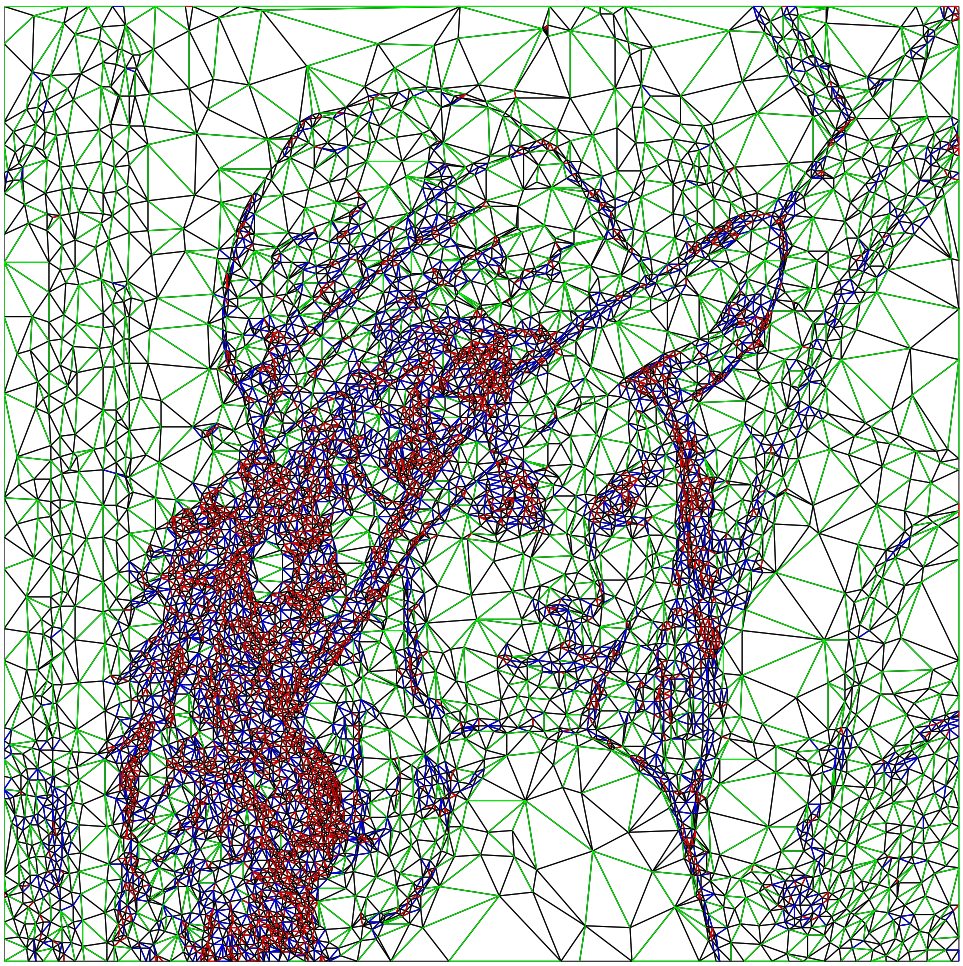


Abbildung 3.9: Lena 512x512, farbige Darstellung für die Einteilung der Grauwertvariationen in die Kontextklassen; rot: Klasse M_0, blau: Klasse M_1, grün: Klasse M_2

Ausgabe: Häufigkeitstabelle $n(v_y|M_j)$, für alle $v_y \in \mathcal{V}$ und $M_j \in \mathcal{M}$;

Daraufhin kann Algorithmus 2 für die kontextuelle adaptive arithmetische Kodierung der quantisierten Grauwerte angewendet und die verschlüsselten Daten als Bitstream gespeichert werden.
Als Testbild verwenden wir *Lena* mit 512×512 Pixel und $|Y| = 8144$ ausgewählten Pixeln. Des Weiteren wählen wir für die Quantisierung der Grauwerte den Quantisierungsfaktor $q = 8$. Mit diesen Grundlage erhalten wir die in Tabelle 3.7 dargestellte Häufigkeitsverteilung der Grauwertvariationen bezüglich der Kontextklassen $M_j \in \mathcal{M}$. Die approximierten Kosten betragen für diese Methode $\tilde{K}_{\mathcal{M}} = 34165$ Bits. In Abbildung 3.9 sehen wir die Einteilung der Grauwertvariationen v_y, für alle $y \in Y$, in die Kontextklassen M_j ($j = 0, 1, 2$).

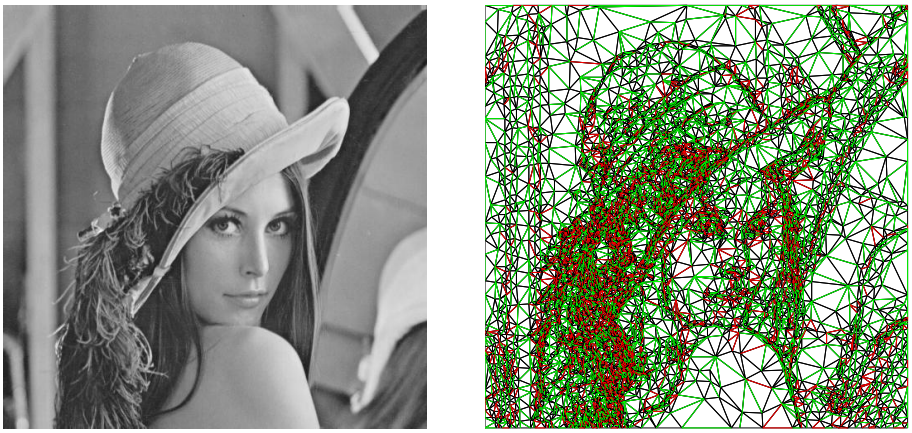


Abbildung 3.10: Lena, 512×512, kritische Kanten der Delaunay Triangulierung $\mathcal{D}(Y)$; rot: $|v| > 3$, grün: $|v| \leq 3$.

Da große Grauwertvariationen hohe Kosten verursachen, versuchen wir in einem zweiten Ansatz die Situationen in Kompressionsmethode 1 zu eliminieren, in denen wir verhältnismäßig lange Kanten, aber große Grauwertvariationen vorfinden. Die farbige Darstellung in Abbildung 3.10 ist so gewählt, dass die kritischen roten Kanten eine Grauwertvariation $|v| > 3$ besitzen und die unkritischen Grünen eine Grauwertvariation $|v| \leq 3$ aufweisen. Die Analyse dieser Darstellung zeigt, dass die längste Kante nicht immer die beste Wahl für die Festlegung der Grauwertvariation darstellt. Insbesondere entlang scharfer Kanten und in den Bereichen mit viel Struktur, kann durch eine angepasste Festlegung der Kontexte ein besseres Ergebnis erzielt werden. Dazu sind weitere Definitionen nötig.

Definition 43 *Es sei Y die Menge der signifikanten Pixel in X. Für jedes Pixel $y \in Y$ sei $l_y \in L(y)$ die Länge der längsten kausalen Kante bezüglich y, mit dem eindeutig festgelegten Pixel $p \in X$, das y entlang dieser Kante gegenüberliegt.*

$$B_l(y) := \{q \in Y_y : [y,q,p] = T \in \mathcal{D}(Y_y)\}$$

bezeichnet die Menge aller Pixel q, die mit den Pixeln y und p ein kausales Dreieck bilden.

Definition 44 *Es sei $y \in Y$ und $B_l(y)$ die Menge der kausalen signifikanten Pixel, die mit der längsten Kante bezüglich y ein kausales Dreieck bilden. Die Menge*

$$\mathcal{D}^*(Y_y) := \{T \in \mathcal{D}(Y_y) : [y,p] \in T\}$$

bezeichnet die Menge der kausalen Dreiecke die durch $B_l(y)$ definiert werden.

Die kritischen Situationen, in denen die längste Kante nicht die beste Wahl für die Bestimmung der Grauwertvariation ist, treten vorwiegend in den Bereichen mit viel Struktur und entlang scharfer Kanten auf. Daher interessieren wir uns für die kausalen Dreiecke bezüglich $y \in Y$, die einen großen *aspect ratio* besitzen. Dazu benötigen wir nun noch folgende Definition.

Definition 45 *Es sei $\mathcal{D}^*(Y_y)$ die Menge der kausalen Dreiecke bezüglich $y \in Y$, die durch $B_l(y)$ definiert werden.*

$$\rho_T^* = \max_{T \in \mathcal{D}^*(Y_y)} \rho_T$$

definiert den größten aspect ratio *des eindeutig festgelegten Dreiecks $T \in \mathcal{D}^*(Y_y)$.*

Basierend auf diesen Definitionen können wir die neue Menge $\mathcal{V}$ der möglichen Grauwertvariationen festlegen.

Definition 46 *Es sei Y die Menge der signifikanten Pixel mit den quantisierten Grauwerten Q_Y und $[y,p] \in B(y)$ die längste kausale Kante bezüglich y, mit dem eindeutig festgelegten Pixel p. Weiter sei $q \in B_l(y)$ das eindeutig festgelegte Pixel, das zusammen mit der längsten Kante $[y,p]$ ein Dreieck $T \in \mathcal{D}^*(Y_y)$ bildet mit ρ_T^*. Es gilt*

$$Y \to \mathcal{V}$$
$$y \mapsto v_y = \begin{cases} Q(f^*(y)) - Q(f^*(p)) & ,\text{wenn } \mathcal{D}^*(Y_y) = \emptyset \\ Q(f^*(y)) - \left(\frac{Q(f^*(p)) + Q(f^*(q))}{2}\right) & ,\text{wenn } \mathcal{D}^*(Y_y) \neq \emptyset \end{cases}.$$

Mit Hilfe des auf diese Weise erzeugten Alphabets $\mathcal{V}$, lässt sich die neue Methode für die Kodierung der Grauwerte wie folgt beschreiben. Die Sequenz wird zeilenweise gelesen. Sie startet wiederum in der linken oberen Ecke des Bildes und endet in der rechten unteren Ecke.

Kompressionsmethode 2
Eingabe: $S = s_1 \ldots s_N$, $N = |Y|$ mit $s_i \in \mathcal{V}$ und $\mathcal{M} = \{M_0, M_1, M_2\}$;

1. *Bestimme das Alphabet $\mathcal{V}$ gemäß Definition 46;*
2. *Für $i = 1, \ldots, N$:*
   (a) *Für $s_i = v_y$ bestimme die längste kausale Kante bezüglich y, mit der Länge l_y;*

(b) Ordne $v_y \in \mathcal{V}$ einer Kontextklasse in $\mathcal{M}$ zu:

$$v_y \mapsto \begin{cases} M_0 & ,\text{wenn } l_y < 5 \\ M_1 & ,\text{wenn } 5 \leq l_y < 8 \\ M_2 & ,\text{sonst} \end{cases};$$

Ende;
Ausgabe: Häufigkeitstabelle $n(v_y|M_j)$, für alle $v_y \in \mathcal{V}$ und $M_j \in \mathcal{M}$;

Daraufhin kann Algorithmus 2 für die kontextuelle adaptive arithmetische Kodierung der quantisierten Grauwerte angewendet und die verschlüsselten Daten als Bitstream gespeichert werden.
Für *Lena* mit 512×512 Pixel, $|Y| = 8144$ und $q = 8$ ergibt sich die in Tabelle (3.7) dargestellte Häufigkeitsverteilung. Die approximierten kombinatorischen Kosten betragen für diese Methode $\tilde{K}_{\mathcal{M}} = 33542$ Bits.

	Kompressionsmethode 1			**Kompressionsmethode 2**		
v_i	M_0	M_1	M_2	M_0	M_1	M_2
-20	2	0	0	0	0	0
-19	1	3	0	1	1	0
-18	4	2	0	0	0	0
-17	7	3	0	0	1	0
-16	5	2	0	0	0	0
-15	15	6	0	3	0	0
-14	18	6	3	4	2	0
-13	22	10	2	10	1	0
-12	25	14	4	16	5	0
-11	39	15	4	19	6	0
-10	42	18	10	26	10	2
-9	53	11	9	36	17	3
-8	69	30	7	53	23	5
-7	78	36	24	77	41	11
-6	80	38	20	103	56	19
-5	112	58	31	149	76	39
-4	131	90	61	182	127	42
-3	160	137	98	230	163	110
-2	236	234	225	238	224	162
-1	323	340	385	301	297	394
0	366	386	577	315	336	492
1	357	303	320	280	282	451
2	238	213	207	266	248	229
3	206	144	91	241	187	101

4	147	105	42	233	134	64
5	125	67	17	167	86	29
6	111	50	20	157	61	18
7	98	55	6	95	53	12
8	69	40	9	70	32	7
9	58	25	6	47	21	2
10	42	26	2	30	13	5
11	40	10	9	25	9	0
12	36	15	2	11	7	0
13	19	9	3	11	3	0
14	22	7	0	3	1	0
15	21	6	2	7	1	0
16	18	4	1	6	1	0
17	8	2	0	4	1	0
18	7	4	0	1	2	0
19	5	2	0	0	0	0
20	1	0	0	1	0	0
21	1	1	0	0	0	0
22	1	1	0	0	0	0

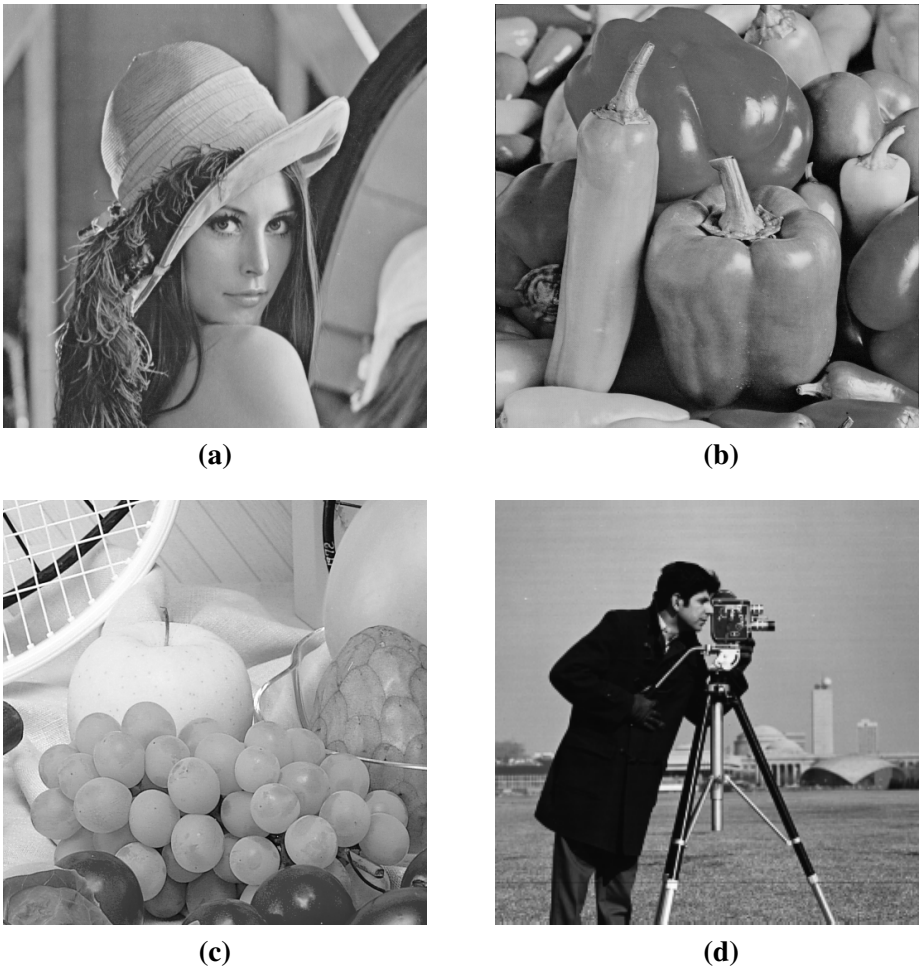
Tabelle 3.7: Lena (512x512), $|Y| = 8144$, $q = 8$, Häufigkeitsverteilung der Grauwertvariationen, Kompressionsmethode 1 und 2

Die Analyse der Gegenüberstellung der zwei neu entwickelten Methoden, in Tabelle 3.7, zeigt, dass wir mit *Kompressionsmethode 2* eine effizientere Festlegung der Grauwertvariationen erzeugt haben.

Wir werden im vierten Kapitel sehen, dass diese Methode speziell für strukturreiche Bilder, wie *Lena*, sehr gute Kompressionsergebnisse liefert.

3.4 Statistische Auswertungen / Ergebnisse

Für die Auswertung der neuen Ansätze ziehen wir die Testbilder *Lena*, *Peppers* und *Fruits* mit jeweils 512×512 Pixeln und das Testbild *Camera* mit 256×256 Pixeln heran. Um ein gute Aussage über die Qualität der neuen Methoden zu treffen, betrachten wir die jeweiligen Bilder bezüglich verschiedener Werte für die Anzahl $|Y|$ der signifikanten Pixel, die mittels dem Adaptive Thinning Algorithmus identifiziert wurden. Um die Ergebnisse miteinander vergleichen zu können, wählen wir für die Bilder der Größe 512×512, $b = 8$ und max $= 12$. Für das Testbild *Camera* setzen wir $b = 7$ und max $= 5$. Wir werden im Folgenden die tatsächlichen Kosten der Kodierung bezüglich der neuen Methoden bestimmen und sie



(a) **(b)**

(c) **(d)**

Abbildung 3.11: Drei Bilder der Größe 512×512: **(a)** *Lena*; **(b)** *Peppers*; **(c)** *Fruits*; Ein Bild der Größe 256×256: **(d)** *Camera*.

mit den Kosten den aktuellen Methoden des Kompressionsverfahren CAT aus den Abschnitten 2.2.2 und 2.2.3 vergleichen.

3.4.1 Pixel Positionen

Wir beginnen mit der Auswertung der Kompressionsmethoden für die Pixel Positionen und erhalten bezüglich *Lena* die tatsächlichen Kosten der Kodierung, die in Tabelle 3.8 dargestellt sind. Abbildung 3.12 zeigt die Kostenersparnis der besten Methoden bezüglich CAT. Wir erhalten folgende Ergebnisse:

- Alle entwickelten Kompressionsmethoden liefern eine deutliche Verbesserung bezüglich der bestehenden Methode.

- *Kompressionsmethode 6* erzeugt für jeden Wert $|Y|$ die größte Reduzierung in den Kosten um durchschnittlich 0.92%.

- Das beste Ergebnis, mit einer Kostenreduzierung von 1.19% erzielt *Kompressionsmethode 6* für $|Y| = 12144$ ausgewählte Pixel.

Bezüglich dem Testbild *Peppers* erhalten wir für die tatsächlichen Kosten der Kodierung der Pixel Positionen die in Tabelle (3.9) dargestellten Werte. Abbildung 3.12 zeigt die Kostenersparnis der besten Methoden bezüglich CAT. Wir erhalten folgende Ergebnisse:

- Für jeden Wert $|Y|$ ist wiederum *Kompressionsmethode 6*, neben *Kompressionsmethode 4*, den anderen Methoden überlegen.

- Der durchschnittliche Gewinn in den Kosten der Kodierung liegt für *Kompressionsmethode 6* bei 0.31%.

- Die größte Reduzierung in den Kosten um 0.77% erhalten wir für $|Y| = 8144$

$\|Y\|$	**CAT**	**M 1**	**M 2**	**M 3**	**M 4**	**M 5**	**M 6**	%
4144	*28391*	28263	28311	28343	28231	28207	**28183**	0.73
4644	*30927*	30815	30863	30895	30783	30767	**30743**	0.59
5144	*33447*	33255	33295	33327	33231	33215	**33199**	0.74
5644	*35823*	35671	35719	35743	35599	35607	**35583**	0.67
6144	*38167*	37967	38023	38023	37895	37927	**37863**	0.80
6644	*40383*	40183	40231	40247	40111	40143	**40087**	0.73
7144	*42727*	42495	42543	42543	42391	42439	**42359**	0.86
7644	*44951*	44687	44727	44727	44583	44623	**44567**	0.85
8144	*47039*	46759	46807	46807	46671	46687	**46647**	0.83
8644	*49063*	48735	48783	48751	48583	48655	**48567**	1.01
9144	*51095*	50735	50791	50751	50599	50679	**50575**	1.02
9644	*53183*	52815	52879	52791	52647	52727	**52623**	1.05
10144	*55207*	54823	54871	54807	54615	54727	**54599**	1.10
10644	*57127*	56719	56783	56687	56471	56591	**56463**	1.16
11144	*58999*	58599	58663	58543	58327	58479	**58319**	1.15
11644	*60895*	60503	60543	60415	60215	60375	**60207**	1.13
12144	*62695*	62279	62311	62151	61959	62151	**61951**	1.19

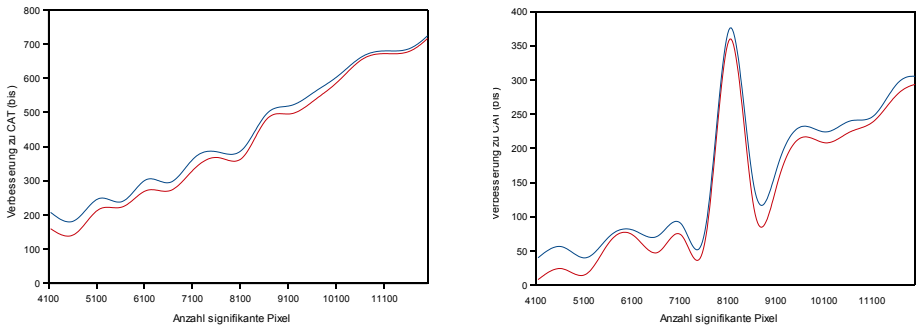
Tabelle 3.8: Lena (512x512), $b = 8$, max $= 12$, Kodierungskosten der Pixel Positionen für CAT und die neu entwickelten Methoden; fett: Methode mit den geringsten Kosten; kursiv: Methode mit den höchsten Kosten; rechts: prozentuale Verbesserung der besten Methode bezüglich CAT

ausgewählte Pixel.

Interessant ist die Betrachtung des Testbildes *Fruits*. Die tatsächlichen Kosten der Kodierung sind in Tabelle 3.10 dargestellt. Abbildung 3.13 zeigt die Kostener-

$\|Y\|$	**CAT**	**Methode 4**	**Methode 6**	%
4144	*29191*	29183	**29151**	0.14
4644	*31839*	31815	**31783**	0.18
5144	*34367*	34351	**34327**	0.12
5644	*36847*	36783	**36775**	0.20
6144	*39319*	39247	**39239**	0.20
6644	*41687*	41639	**41615**	0.17
7144	*43983*	43911	**43895**	0.20
7644	*46279*	46207	**46191**	0.19
8144	*48727*	48367	**48351**	0.77
8644	*50719*	50607	**50575**	0.28
9144	*53055*	52903	**52879**	0.33
9644	*55207*	54991	**54975**	0.42
10144	*57455*	57247	**57231**	0.39
10644	*59543*	59319	**59303**	0.40
11144	*61655*	61415	**61407**	0.40
11644	*63751*	63471	**63455**	0.46
12144	*65839*	65543	**65535**	0.46

Tabelle 3.9: Peppers (512x512), $b = 8$, max $= 12$, Kodierungskosten der Pixel Positionen für CAT und die neu entwickelten Methoden 4 und 6; fett: Methode mit den geringsten Kosten; kursiv: Methode mit den höhsten Kosten; rechts: prozentuale Verbesserung der besten Methode bezüglich CAT



(a) Lena 512×512 (b) Peppers 512×512

Abbildung 3.12: Reduzierung der Kodierungskosten der Pixel Positionen bezüglich CAT; rot: Methode 4, blau: Methode 6

sparnis der besten Methoden bezüglich CAT. Wir erhalten folgende Ergebnisse:

- Für $|Y| \leq 6644$ ist *Kompressionsmethode 5* mit einer durchschnittlichen Reduzierung um 0.28% den anderen Kompressionmethoden vorzuziehen.
- Steigt die Anzahl der signifikanten Pixel allerdings über 6644 Pixel, erzielt *Kompressionsmethode 2*, mit einer durchschnittliche Kostenreduzierung um 0.33% deutlich bessere Ergebnisse.
- Die größte Ersparnis von 0.39% erzielen wir unter Verwendung von *Kompressionsmethode 2* für $|Y| = 6644$.

Als letztes betrachten wir die tatsächlichen Kosten der Kodierung der Pixel Positionen für das Bild *Camera*. Die Kosten für die unterschiedlichen Werte $|Y|$ sind in Tabelle 3.11 dargestellt. Abbildung 3.13 zeigt die Kostenersparnis der besten Methoden bezüglich CAT. Wir erhalten folgende Ergebnisse:

- Für $|Y| \leq 2036$ ist *Kompressionsmehtode 6* mit der durchschnittlichen Reduzierung der Kosten um 1.58% die erste Wahl.

$\|Y\|$	**CAT**	**Methode 2**	**Methode 5**	%
4144	*28615*	28591	**28487**	0.45
4644	*31231*	31215	**31143**	0.28
5144	*33695*	33703	**33599**	0.28
5644	*36231*	36231	**36135**	0.26
6144	*38687*	38671	**38567**	0.31
6644	*40999*	40959	**40839**	0.39
7144	*43159*	43111	**43023**	0.32
7644	*45359*	45327	**45223**	0.30
8144	*47631*	47607	**47527**	0.22
8644	*49823*	49759	**49711**	0.22
9144	*52111*	52031	**51991**	0.23
9644	*54159*	54103	**54095**	0.12
10144	*56231*	**56111**	56135	0.21
10644	*58207*	**58023**	58111	0.32
11144	*60223*	**60007**	60087	0.36
11644	*62223*	**61983**	62111	0.39
12144	*64159*	**63919**	64095	0.37

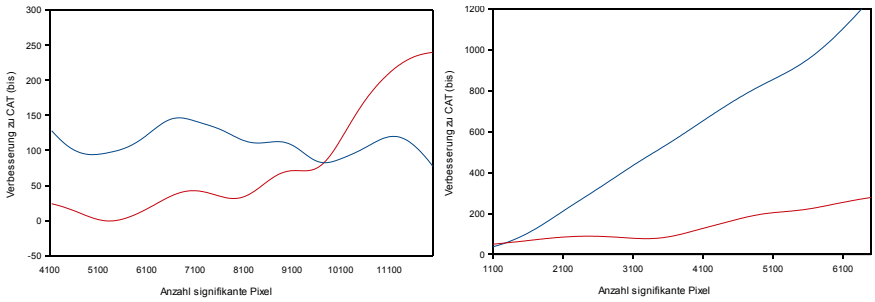
Tabelle 3.10: Fruits (512*x*512), $b = 8$, max $= 12$, Kodierungskosten der Pixel Positionen für CAT und die neu entwickelten Methoden 2 und 5; fett: Methode mit den geringsten Kosten; kursiv: Methode mit den höhsten Kosten; rechts: prozentuale Verbesserung der besten Methode bezüglich CAT

- Gilt $|Y| > 2036$ ist *Kompressionsmethode 2* den anderen Methoden überlegen.
- Das beste Ergebnis für *Kompressionsmethode 2* erhalten wir für $|Y| = 6536$. Hier beträgt der Gewinn 4.98%.

Die Analyse zeigt, dass speziell für strukturreiche Bilder, wie *Lena*, *Kompressionsmethode 6* sehr gute Ergebnisse liefert. Für einfache kleinere Bilder mit viel Kanteninformation ist *Kompressionsmethode 2* die bessere Wahl. Diese Ergebnisse

$\|Y\|$	**CAT**	**Methode 2**	**Methode 6**	%
1036	*6553*	6521	**6489**	0.98
1536	*8945*	8873	**8801**	1.61
2036	*11201*	10993	**10961**	2.14
2536	*13249*	**12953**	13017	2.23
3036	*15145*	**14713**	14937	2.85
3536	*16833*	**16313**	16593	3.09
4036	*18401*	**17761**	18089	3.48
4536	*19937*	**19185**	19609	3.77
5036	*21409*	**20553**	21009	4.00
5536	*22761*	**21849**	22433	4.01
6036	*24153*	**23081**	23801	4.44
6536	*25385*	**24121**	24977	4.98

Tabelle 3.11: Camera (256x256), $b = 7$, max $= 5$, Kodierungskosten der Pixel Positionen für CAT und die neu entwickelten Methoden 2 und 6; fett: Methode mit den geringsten Kosten; kursiv: Methode mit den höhsten Kosten; rechts: prozentuale Verbesserung der besten Methode bezüglich CAT



(a) Fruits 512×512 (b) Camera 256×256

Abbildung 3.13: Reduzierung der Kodierungskosten der Pixel Positionen bezüglich CAT; rot: Methode 5, blau: Methode 2

werden wir für die Entwicklung des neuen Kompressionsverfahren CAT* verwenden.

3.4.2 Grauwerte

Wir beginnen nun die Analyse der Kompressionsmethoden für die Grauwerte und erhalten bezüglich dem Testbild *Lena*, in Tabelle 3.12, die Darstellung der tatsächlichen Kosten der Kodierung. Abbildung 3.14 zeigt die Kostenersparnis der beiden Methoden bezüglich CAT. Wir erhalten folgende Ergebnisse:

- *Kompressionsmethode 2* liefert für jeden dargestellten Wert $|Y|$ die größte Reduzierung der Kosten.
- Die durchschnittliche Reduzierung der Kosten liegt für *Kompressionsmethode 1* bei 279 Bits. *Kompressionsmethode 2* hingegen liefert eine durchschnittliche Verbesserung von 1489 Bits, dies sind 4.27% bezüglich der bestehenden Methode.
- Für $|Y| = 12144$ erhalten wir die größte Ersparnis von 5.74%.

$\|Y\|$	**CAT**	**Methode 1**	**Methode 2**	%
4144	*17432*	17200	**16992**	2.52
4644	*19352*	19160	**18800**	2.85
5144	*21400*	21144	**20656**	3.48
5644	*23360*	23136	**22664**	2.98
6144	*25320*	25080	**24464**	3.38
6644	*27256*	26992	**26224**	3.79
7144	*29024*	28848	**27984**	3.58
7644	*30952*	30728	**29664**	4.16
8144	*32904*	32664	**31424**	4.50
8644	*34776*	34528	**33160**	4.65
9144	*36728*	36360	**34960**	4.81
9644	*38576*	38216	**36488**	5.41
10144	*40320*	40056	**38384**	4.80
10644	*42264*	42008	**40096**	5.13
11144	*44072*	43728	**41744**	5.28
11644	*45976*	45576	**43432**	5.53
12144	*47832*	47376	**45088**	5.74

Tabelle 3.12: Lena (512x512), $q = 8$, Kodierungskosten der Grauwerte

Die tatsächlichen Kosten für die Kodierung der Grauwerte bezüglich des Testbilds *Peppers* sind in Tabelle (3.9) dargestellt. Abbildung 3.14 zeigt die Kostenersparnis der beiden Methoden bezüglich CAT. Wir erhalten folgende Ergebnisse:

- Auch für das Testbild *Peppers* liefert *Kompressionsmethode 2* für jeden dargestellten Wert $|Y|$ die größte Reduzierung in den Kosten bezüglich der Methode bei CAT.
- Die durchschnittliche Reduzierung der Kosten liegt für *Kompressionamethode 1* bei 94 Bits. *Kompressionsmethode 2* hingegen liefert eine durchschnittliche Verbesserung von 1215 Bits. Dies ist eine durchschnittliche Ersparnis von 3.45% bezüglich der bestehenden Methode.
- Für $|Y| = 12144$ erhalten wir die größte Ersparnis von 4.68%.

Für den Testfall *Fruits* sind in Tabelle 3.9 die tatsächlichen Kosten für die Kodierung der Grauwerte zusammengefasst. Abbildung 3.15 zeigt die Kostenersparnis der beiden Methoden bezüglich CAT. Wir erhalten folgende Ergebnisse:

- In diesem Testfall erhalten wir für *Kompressionsmethode 2* bis zu dem Wert $|Y| = 8144$ nicht die gewünschten Ergebnisse, hier überwiegt *Kompressionsmehtode 1*.

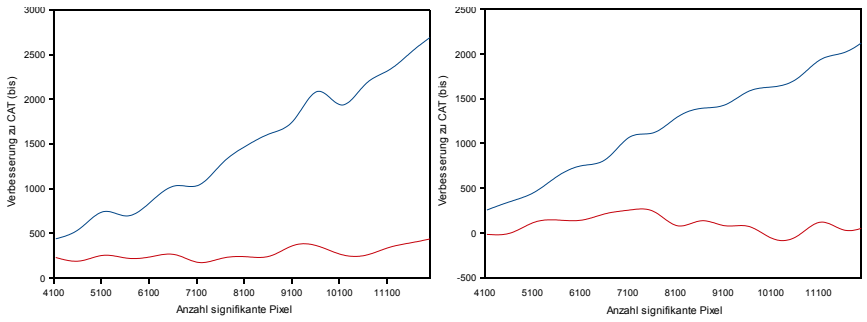
$\|Y\|$	**CAT**	**Methode 1**	**Methode 2**	%
4144	17120	*17136*	**16864**	1.50
4644	*19032*	19032	**18680**	1.85
5144	*20904*	20784	**20448**	2.18
5644	*22904*	22760	**22264**	2.79
6144	*24744*	24600	**23992**	3.04
6644	*26664*	26448	**25840**	3.09
7144	*28632*	28376	**27560**	3.74
7644	*30408*	30168	**29288**	3.68
8144	*32304*	32224	**31008**	4.01
8644	*34152*	34016	**32760**	4.08
9144	*35840*	35760	**34408**	4.00
9644	*37600*	37528	**36016**	4.21
10144	39384	*39448*	**37752**	4.14
10644	41040	*41080*	**39320**	4.19
11144	*42896*	42776	**40960**	4.51
11644	*44632*	44600	**42616**	4.52
12144	*46496*	46416	**44320**	4.68

Tabelle 3.13: Peppers (512x512), $q = 8$, Kodierungskosten der Grauwerte

- Ab dem Wert $|Y| = 8644$ ist *Kompressionsmehtode 2* die beste Wahl.
- Die durchschnittliche Reduzierung der Kosten liegt für *Kompressionamethode 1* bei 98 Bits. *Kompressionsmethode 2* liefert eine durchschnittliche Verbesserung von 58 Bits.
- Die durchschnittliche Ersparnis, wenn zwischen den Methoden hin- und hergeschaltet werden kann, liegt bei 0.67% bezüglich der bestehenden Methode.
- Das beste Ergebnis liefert *Kompressionsmehtode 2* für $|Y| = 12144$. Hier erhalten wir eine Verbesserung von 1.92%.

Als letztes betrachten wir die tatsächlichen Kosten der Kodierung der Grauwerte für das Bild *Camera*. In Tabelle 3.11 sind wiederum die Kosten für die unterschiedlichen Werte $|Y|$ dargestellt. Abbildung 3.15 zeigt die Kostenersparnis der beiden Methoden bezüglich CAT. Wir erhalten folgende Ergebnisse:

- Für das Testbild *Camera* liefert *Kompressionsmehtode 1* für die Werte $|Y| \leq 3536$ die besten Ergebnisse. *Kompressionsmethode 1* erzielt für diese Werte keine Verbesseung bezüglich CAT.
- Für die Werte $|Y| > 3536$ kann mit *Kompressionsmehtode 2* eine größere Ersparnis als mit *Kompressionsmehtode 1* erzielt werden. Die durchschnittliche Anzahl an Bits, die wir mit den neu entwickelten Methoden gewinnen, betragen für *Kompressionsmehtode 1* 101 Bits und für *Methode 2* 141 Bits. Wir können insgesamt mit den entwickelten Methoden eine Ersparnis von 1.03% erzielen.
- Das beste Ergebnis erhalten wir für $|Y| = 6536$ unter Verwendung von *Kompressionsmethode 2*.



(a) Lena 512×512 (b) Peppers 512×512

Abbildung 3.14: Reduzierung der Kodierungskosten der Grauwerte bezüglich CAT; rot: Methode 1, blau: Methode 2

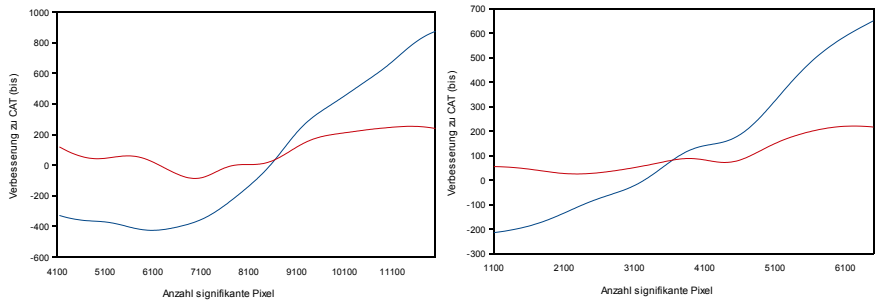
Wir haben gesehen, dass die neu entwickelte *Kompressionsmehtode 2* speziell für strukturreiche Bilder, wie *Lena*, sehr gute Ergebnisse liefert. Doch nicht nur in diesem Testfall erhalten wir eine große Ersparnis in den Kosten. Auch für *Peppers*

$\|Y\|$	**CAT**	**Methode 1**	**Methode 2**	%
4144	16512	**16392**	*16840*	0.73
4644	18432	**18400**	*18808*	0.17
5144	20360	**20328**	*20704*	0.16
5644	22104	**22000**	*22528*	0.47
6144	23960	**23944**	*24400*	0.07
6644	**25856**	25928	*26256*	0.00
7144	**27832**	27960	*28200*	0.00
7644	29696	**29664**	*29952*	0.11
8144	**31544**	31552	*31672*	0.00
8644	*33416*	33408	**33408**	0.02
9144	*35264*	35120	**35000**	0.75
9644	*37192*	36992	**36832**	0.97
10144	*38952*	38744	**38496**	1.17
10644	*40944*	40704	**40368**	1.41
11144	*42624*	42376	**41960**	1.56
11644	*44536*	44272	**43688**	1.90
12144	*46312*	46080	**45424**	1.92

Tabelle 3.14: Fruits (512x512), $q = 8$, Kodierungskosten der Grauwerte

$\|Y\|$	**CAT**	**Methode 1**	**Methode 2**	%
1036	4560	**4504**	*4776*	1.23
1536	6896	**6840**	*7096*	0.81
2036	9200	**9184**	*9352*	0.17
2536	11552	**11528**	*11608*	0.21
3036	13816	**13768**	*13880*	0.35
3536	*16112*	**16040**	16048	0.45
4036	*18376*	18256	**18200**	0.96
4536	*20472*	20464	**20360**	0.55
5036	*22776*	22600	**22480**	1.30
5536	*24960*	24768	**24472**	1.96
6036	*27168*	26936	**26584**	2.15
6536	*29352*	29136	**28696**	2.23

Tabelle 3.15: Camera (256x256), $q = 8$, Kodierungskosten der Grauwerte für die Kompressionsmethode CAT und die neu entwickelten Kompressionsmethodne 1 und 2



(a) Fruits 512×512 (b) Camera 256×256

Abbildung 3.15: Reduzierung der Kodierungskosten der Grauwerte bezüglich CAT; rot: Methode 1, blau: Methode 2

ist *Kompressionsmethode 2* die erste Wahl. Für die Testbilder *Fruits* und *Camera* lassen sich für größere Werte für $|Y|$ unter Verwendung von *Kompressionsmethode 2* ebenfalls eine nennenswerte Reduzierung in den Kosten erzielen. Diese Ergebnisse werden wir für die Entwicklung des neuen Kompressionsverfahren CAT* nutzen.

Kapitel 4

Kompressionsverfahren CAT*

In diesem Kapitel werden wir unsere neu entwickelte Bildkompressionsmethode CAT* vorstellen und unser Augenmerk speziell auf den allgemeinen Verlauf des Verfahrens richten. Den Abschluss bilden die numerischen Ergebnisse des Verfahrens CAT* im Vergleich zu CAT und JPEG2000.

4.1 Beschreibung des Verfahrens

Das auf der anisotropen Delaunay Triangulierung basierende Kompressionsverfahren CAT* beschäftigt sich mit der Umrechnung digitaler Bildaten in einen *Bitstream*, der auf einem Medium gespeichert oder mittels einer Netzwerkverbindung übertragen werden kann. Um den Speicherplatzbedarf zu reduzieren und die Übertragungsgeschwindigkeit zu erhöhen, sollte die Länge dieses Bitstreams möglichst kurz gehalten werden, während die erforderliche Qualität des Bildes erhalten bleiben soll. Für diesen Prozess ist es erforderlich, die folgenden vier Punkte Schritt für Schritt abzuarbeiten.

(1) Reduktion der Daten;

(2) Kompression der Daten beim Sender;

(3) Übermittlung der kodierten Daten vom Sender zum Empfänger;

(4) Rekonstruktion der Daten;

Das Bild sei wiederum über der Menge X der äquidistant verteilten Pixel erklärt, deren Positionen ein zweidimensionales rechteckiges Gitter bilden. In Kombination mit den zugehörigen Grauwerten $\{f(x) : x \in X\}$ erhalten wir eine bivariate diskrete Funktion $f : X \to \mathbb{R}$, die das Bild darstellt.

(1) Reduktion der Daten

Für das neu entwickelte Verfahren verwenden wir für die Reduktion der Bilddaten die Bildapproximationsmethode mit Hilfe der anisotropen Delaunay Triangulierung, die in Abschnitt 2.1 vorgestellt wurde. Die beste lineare Spline-Approximation $s* \in \mathcal{S}_{\mathcal{D}}^{0}$, mit der zugehörigen Menge Y der ausgewählten Knoten, approximiert das Bild. Die daraus resultierende, reduzierte Bilddarstellung $\{(y, f^*(y)) : y \in Y\}$ wird im zweiten Schritt des Prozesses kodiert bzw. komprimiert.

(2) Kompression der Daten beim Sender

Als Grundlage für die Kompression der reduzierten Bilddarstellung wählen wir einen kontextuellen adaptiven arithmetischen Kodierer, der die Daten in der folgenden Reihenfolge komprimiert:

- Kompression der Pixel Positionen
- Kompression der quantisierten Grauwerte

Als Grundlage für die Kompression der Pixel Positionen dienen die in Abschnitt 3.2 neu entwickelten Methoden. Für Bilder der Größe $n > 65536$, mit $n = |X|$, wählen wir für die Kompression der Pixel Positionen *Kompressionsmethode 6*. Für Bilder der Größe $n \leq 65536$, mit $n = |X|$, wählen wir für die Kompression der Pixel Positionen *Kompressionsmethode 2*. Die Differenzierung resultiert aus der Tatsache, dass für kleine Bilder eine verhältnismäßig größere Anzahl ausgewählter Knoten $|Y|$ benötigt wird als bei größeren Bilder, um eine gute Qualität der Rekonstruktion zu erhalten.
Im nächsten Schritt erfolgt die Kompression der im Vorhinein quantisierten Grauwerte. Als Grundlage bedienen wir uns der Ergebnisse aus Abschnitt 3.3 und wählen für die Kompression der quantisierten Grauwerte die neu entwickelte *Kompressionsmethode 2*.

(3) Übermittlung der kodierten Daten vom Sender zum Empfänger

Der durch die Kompression der reduzierten Bilddarstellung erzeugte *Bitstream* kann nun auf einem Medium gespeichert oder über Netzwerkverbindungen versendet werden.

(4) Rekonstruktion der Daten

Erhält der Empfänger die übermittelte Nachricht, erfolgt die Dekodierung und Rekonstruktion der verschlüsselten Daten gemäß Abschnitt 2.3.

4.2 Numerische Ergebnisse

In diesem Abschnitt werden wir die neu entwickelte Kompressionsmethode CAT* bezüglich ihrer Effizienz im Vergleich zu JPEG2000 und dem bestehenden Verfahren CAT untersuchen. Für JPEG2000 verwenden wir die Kakadu Implementierung, die unter [23] verfügbar ist. Die präsentierten Ergebnisse der Kompressionsverfahren CAT und CAT* entsprechen den realen Kosten, basierend auf den vollständig implementierten Kompressions- und Dekompressions-Schemata. Für die Auswertung des neu entwickelten Kompressionsverfahrens CAT* verwenden wir die drei Testbilder *Lena*, *Peppers* und *Fruits* mit jeweils 512×512 Pixel und das Testbild *Camera* mit 256×256 Pixel. Die Farbinformation der Testbilder beschränkt sich auf jeweils 256 mögliche Graustufen.

Tabelle 4.1 zeigt für die vier Testbilder und die drei verschiedenen Kompressionsverfahren JPEG2000, CAT und CAT* den *peak signal to noise ratio* (PSNR, gemessen in dB) in Bezug auf die komprimierte Dateigröße (gemessen in *bit per pixel*, bpp). Abbildung 4.1 stellt die zugehörige *rate-distortion Kurve* dar. Der PSNR wird als Funktionwert über der zugrundeliegenden Dateigröße betrachtet. In Tabelle 4.2 sind für die vier Testfälle und die drei Kompressionsverfahren die Bitraten in Abhängigkeit vom PSNR dargestellt. Da die Werte des PSNR für die verschiedenen Kompressionsmethoden selten übereinstimmen, wählen wir als Grundlage für den Vergleich der Bitraten leicht voneinander abweichende PSNR-Werte. Abbildung 4.2 stellt den Vergleich als *rate-distortion Kurve* über dem PSNR dar.

	Jpeg2000		**CAT**		**CAT***	
	PSNR (dB)	**bpp**	**PSNR (dB)**	**bpp**	**PSNR (dB)**	**bpp**
Lena	33.00	*0.1900*	33.12	0.1748	33.11	**0.1723**
	33.42	*0.2100*	33.50	0.1918	33.50	**0.1890**
	33.42	*0.2200*	33.40	0.2092	33.50	**0.1890**
	34.13	*0.2400*	34.09	0.2258	34.10	**0.2222**
	34.35	*0.2500*	34.36	0.2422	34.38	**0.2378**
	34.56	*0.2600*	34.59	0.2580	34.59	**0.2530**
	34.86	*0.2800*	34.79	0.2737	34.80	**0.2683**
	35.03	*0.2900*	35.00	0.2895	35.00	**0.2832**
	35.16	*0.3000*	35.17	0.3050	35.17	**0.2978**
	35.31	0.3100	35.33	*0.3198*	35.33	**0.3118**
	35.50	**0.3200**	35.46	*0.3350*	35.46	0.3263
	35.63	**0.3300**	35.61	*0.3500*	35.61	0.3399
	35.80	**0.3400**	35.76	*0.3644*	35.76	0.3547
	35.92	**0.3500**	35.89	*0.3791*	35.90	0.3683
	36.02	**0.3600**	36.02	*0.3932*	36.02	0.3817

Peppers	33.43	*0.2300*	33.14	0.1767	33.54	**0.1755**
	33.8	*0.2500*	33.82	0.1941	33.82	**0.1925**
	34.3	*0.2900*	34.05	0.2108	34.05	**0.2090**
	34.43	*0.3000*	34.4	0.2444	34.4	**0.2412**
	34.54	*0.3100*			34.53	**0.2573**
	34.66	*0.3200*			34.67	**0.2726**
	35.44	*0.4100*			35.4	**0.3762**
	35.47	*0.4200*			35.47	**0.3905**
	35.59	*0.4300*			35.61	**0.4191**
Fruits	33.27	*0.2500*	33.14	0.1721	33.14	**0.1712**
	33.46	*0.2600*	33.46	0.1894	33.46	**0.1890**
	33.82	*0.2800*	33.79	0.2062	33.79	**0.2057**
	34.02	*0.2900*	34.08	0.2225	34.08	**0.2218**
	34.39	*0.3200*	34.33	0.2390	34.34	**0.2385**
	34.50	*0.3300*	34.55	0.2550	34.55	**0.2547**
	34.94	*0.3700*	34.90	0.2863	34.91	**0.2857**
	35.08	*0.3700*	35.10	0.3020	35.10	**0.3017**
	35.57	*0.4100*	35.57	0.3485	35.57	**0.3469**
	36.00	*0.4500*	35.98	0.3923	35.98	**0.3890**
	36.11	*0.4600*	36.09	0.4073	36.10	**0.4031**
	36.21	*0.4700*	36.20	0.4214	36.21	**0.4171**
Camera	29.61	*0.3900*	29.86	0.3113	29.86	**0.3104**
	30.90	*0.4900*	30.66	0.3784	30.66	**0.3748**
	31.44	*0.5400*	32.04	0.5027	32.04	**0.4938**
	32.79	*0.6400*	32.60	0.5612	32.60	**0.5487**
	33.05	*0.6700*	33.11	0.6166	33.11	**0.6034**
	33.61	*0.7100*	33.66	0.6742	33.66	**0.6566**
	34.13	*0.7600*	34.14	0.7282	34.14	**0.7068**
	34.52	*0.7900*	34.62	0.7831	34.62	**0.7578**
	35.06	*0.8600*	35.06	0.8352	35.07	**0.8059**

Tabelle 4.2: Vergleich der Testbilder Lena, Fruits, Peppers und Camera in Bezug auf JPEG2000, CAT und CAT*. Dargestellt ist die Bitrate bpp in Abhängigkeit vom PSNR.

Die Auswertung zeigt, dass die Kompressionsmethode CAT*, die in dieser Arbeit entwickelt und implementiert wurde, für alle Testfälle der Kompressionsmethode CAT überlegen ist. Darüber hinaus liefert CAT* überwiegend bessere Ergebnisse als JPEG2000. Im Speziellen kann beobachtet werden, dass für stark geometrische Bilder, wie *Fruits*, die Kompressionsverfahren CAT und CAT* der Kompressionsmethode JPEG deutlich überlegen sind. Auch für Bilder mit viel Kanteninformation, wie im Fall *Camera*, sind die Verfahren CAT und CAT* der Kompres-

		PSNR (dB)		
	bpp	**JPEG2000**	**CAT**	**CAT***
Lena	0.19	*33.00*	33.50	**33.50**
	0.21	33.42	*33.40*	**33.80**
	0.24	*34.13*	34.36	**34.38**
	0.27	*34.70*	34.79	**34.80**
	0.30	*35.16*	35.17	**35.17**
	0.34	**35.80**	*35.46*	35.61
	0.35	**35.92**	*35.61*	35.76
	0.38	**36.25**	*35.89*	36.02
	0.41	**36.56**	*36.16*	36.27
Peppers	0.18	*32.48*	33.14	**33.54**
	0.19	*32.64*	33.82	**33.82**
	0.21	*33.07*	34.05	**34.05**
	0.23	*33.43*	34.23	**34.23**
	0.38	*35.14*		**35.4**
	0.39	*35.27*		**35.47**
	0.40	*35.34*		**35.52**
	0.42	*35.47*		**35.61**
Fruits	0.17	*31.55*	33.14	**33.14**
	0.19	*32.10*	33.46	**33.46**
	0.21	*32.53*	33.79	**33.79**
	0.24	*33.01*	34.33	**34.34**
	0.29	*34.02*	34.90	**34.91**
	0.32	*34.39*	35.26	**35.26**
	0.38	*35.22*	35.86	**35.86**
	0.42	*35.71*	36.20	**36.21**
Camera	0.17	*25.48*	27.55	**27.55**
	0.24	*27.14*	28.88	**28.89**
	0.44	*30.21*	31.37	**31.37**

Tabelle 4.1: Vergleich der Testbilder Lena, Fruits, Peppers und Camera in Bezug auf JPEG2000, CAT und CAT*. Dargestellt ist der PSNR in Abhängigkeit der Bitrate (bpp).

sionsmethode JPEG2000 vorzuziehen. Für sehr strukturreiche Bilder, wie *Lena*, überwiegt JPEG2000 ab 0.34 bpp. Dies ist darin begründet, dass die hohe Auflösung der Struktur, unter der Verwendung der Verfahren CAT oder CAT*, eine größere Anzahl signifikanter Pixel benötigt. Die waveletbasierte Methode JPEG2000 hingegen, liefert in diesem Fall für höhere Bitraten bessere Ergebnisse. Allerdings kann die neue Methode CAT*, durch die neu entwickelten Kontexte für die Kompression der Pixel Positionen und der Grauwerte, diesem Problem in einem ge-

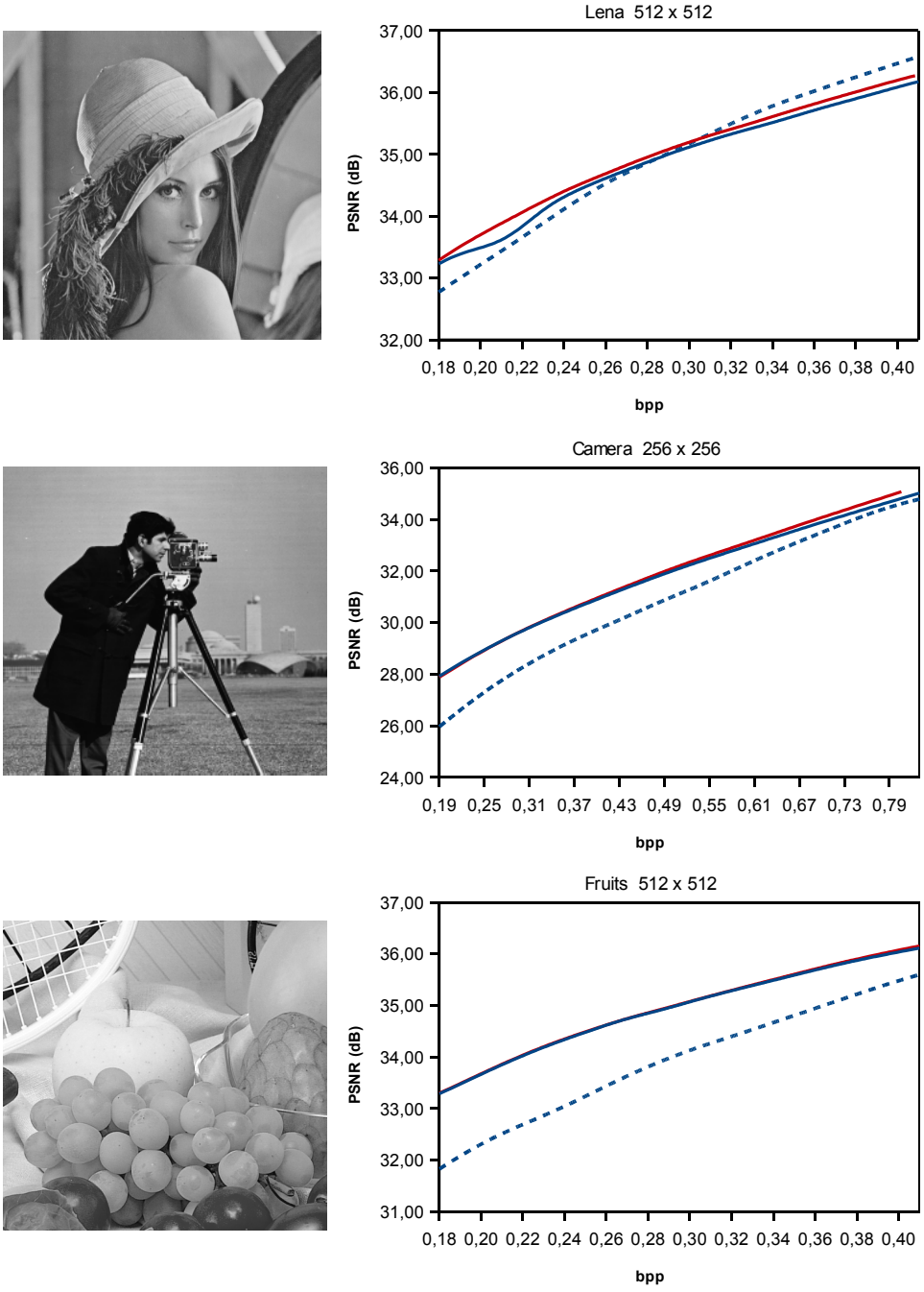


Abbildung 4.1: Vergleich der Bilder Lena, Camera und Fruits: JPEG2000, CAT und CAT* (bpp vs PSNR); blaue durchbrochene Linie: JPEG2000, blaue Linie: CAT, rote Linie: CAT*.

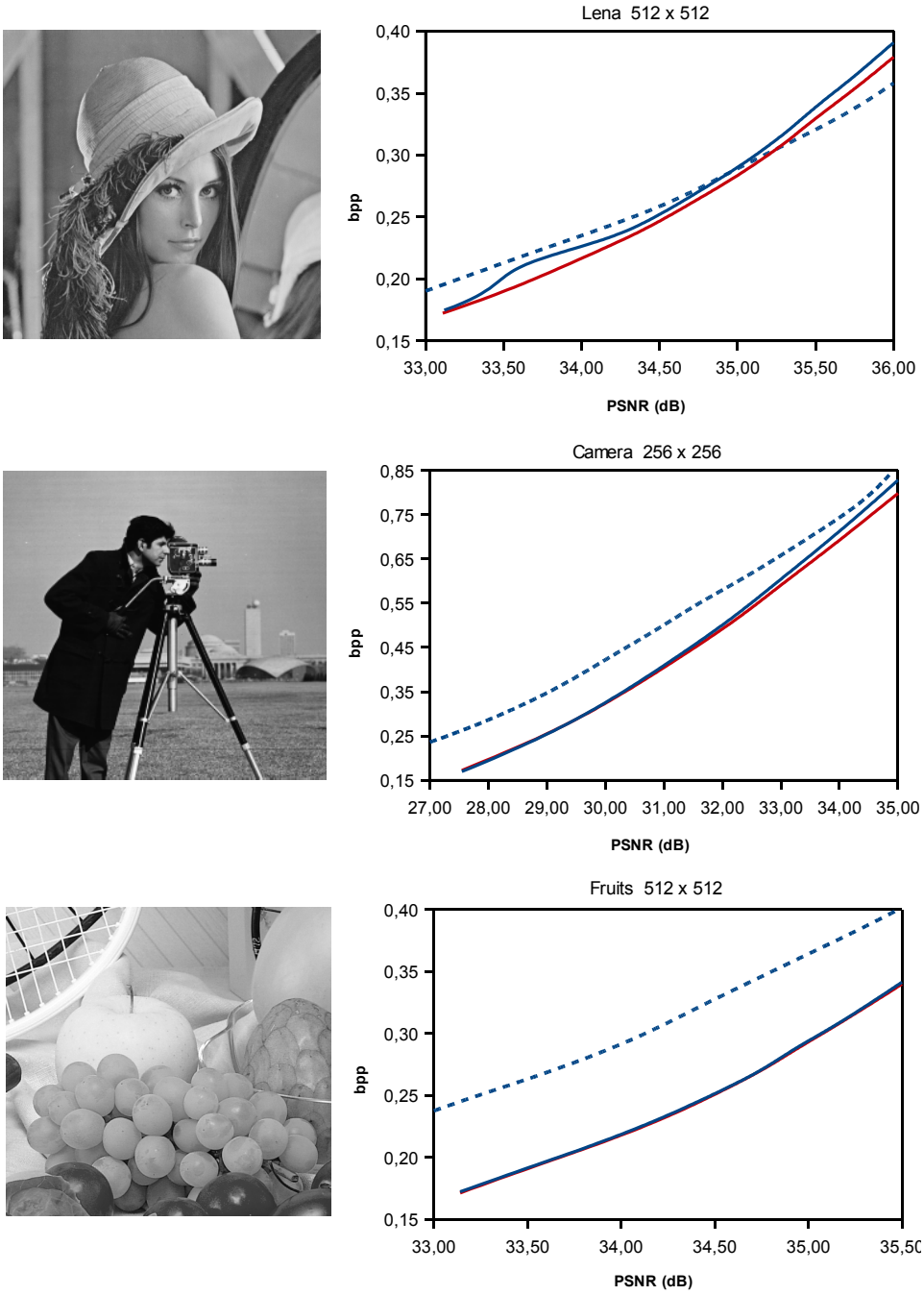


Abbildung 4.2: Vergleich der Bilder Lena, Camera und Fruits: JPEG2000, CAT und CAT* (PSNR vs bpp); blaue durchbrochene Linie: JPEG2000, blaue Linie: CAT, rote Linie: CAT*.

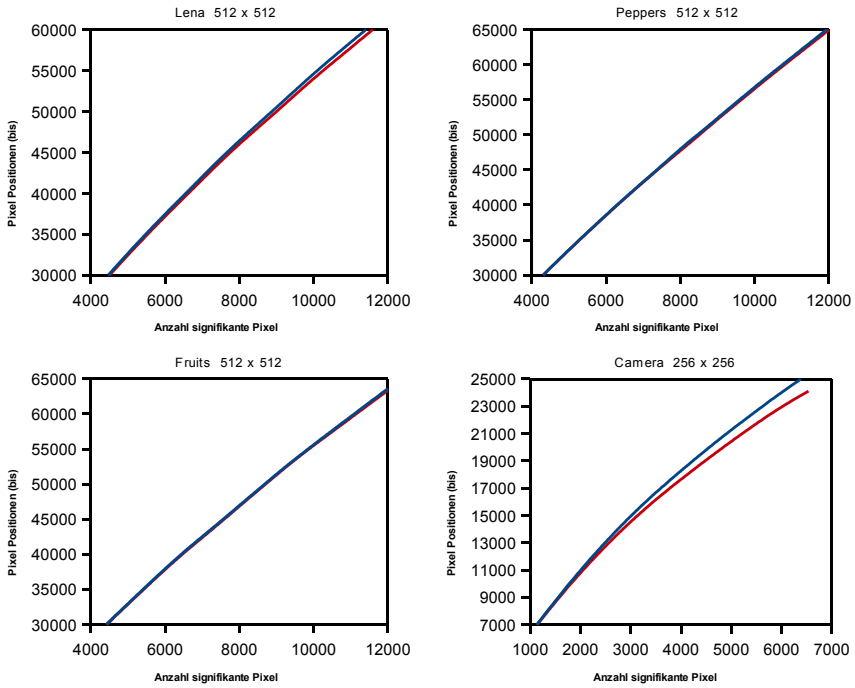


Abbildung 4.3: Vergleich der Kodierungskosten der Pixel Positionen, gemessen in Bits, für Lena, Peppers, Camera und Fruits. Verglichen werden die Verfahren CAT (blaue Kurve) und CAT* (rote Kurve) (Anzahl signifikante Pixel vs Kosten).

wissen Maße entgegenwirken. Kompressionsmethode CAT* kann die Strukturinformationen besser erfassen und liefert damit, speziell für hohe Bitraten, bessere Ergebnisse als Kompressionsmethode CAT, wie im Testfall *Lena*, *Peppers* und *Camera* deutlich zu erkennen ist. Die Effizienz der Kompression der Pixel Postitionen und der Grauwerte des Verfahrens CAT*, im Vergleich zu der bestehenden Kompressionsmethode CAT, ist in den Abbildungen 4.3 und 4.4 dargestellt. Es ist zu erkennen, dass für alle Testbilder, speziell für eine hohe Anzahl signifikanter Pixel $|Y|$, die Kosten und damit der bpp-Wert sinkt.

Abbildung 4.5 zeigt den visuellen Vergleich der Kompressionsmethoden CAT* und JPEG2000 für die Testbilder *Lena* und *Peppers* bei einer niedrigen Bitrate. Des Weiteren ist der resultierende PSNR für die jeweiligen Bitraten und die zugehörige Delaunay Triangulierung dargestellt. Abbildung 4.6 zeigt die rekonstruierten Testbilder *Fruits* und *Camera* mit einer mittleren und einer hohen Bitrate. Verglichen werden wiederum die Methoden CAT* und JPEG2000.

Abschließend betrachten wir den visuellen Vergleich der Kompressionverfahren CAT* und JPEG2000 bezüglich drei besonders aussagekräftiger Testfälle. Abbildung 4.7 zeigt das Testbild *Peppers* bei einer Bitrate von 0.18 bpp. Die Abbil-

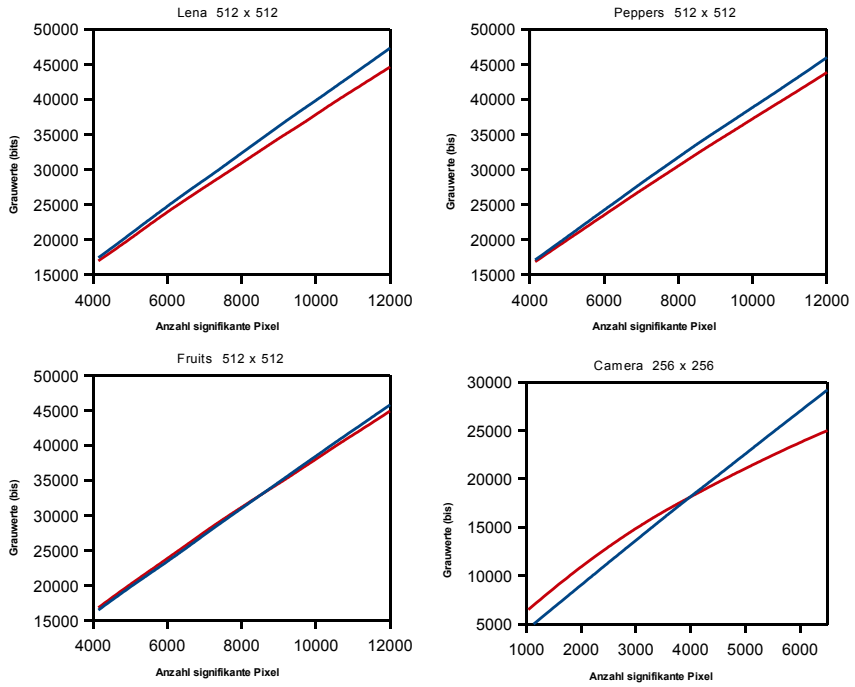


Abbildung 4.4: Vergleich der Kodierungskosten der Grauwerte, gemessen in Bits, für Lena, Peppers, Camera und Fruits. Verglichen werden die Verfahren CAT (blaue Kurve) und CAT* (rote Kurve) (Anzahl signifikante Pixel vs Kosten).

dung 4.8 bzw. 4.9 zeigt das Testbild *Fruits* bzw. *Camera* bei einer Bitrate von 0.17 bpp. Hier ist zu beachten, dass speziell für das Bild *Camera* die neu entwickelte Methode CAT* und CAT, im Gegensatz zu JPEG2000, die oszillierenden Artefakte entlang der Kanten vermeidet. Wie die Testfälle zeigen, erhält das Kompressionsverfahren CAT* und CAT die Kantendarstellung und wirkt entrauschend.

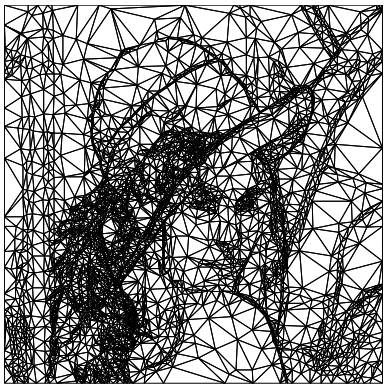
Zusammenfassend ist zu erwähnen, dass Kompressionsmethode CAT* (wie auch Kompressionsmethode CAT) für stark geometrische Bilder und für Bilder mit weiten strukturarmen Bereichen dem Kompressionsverfahren JPEG2000 überlegen ist. Des Weiteren konnten für strukturreiche Bilder eine deutliche Verbesserungen in Bezug auf CAT erzielt werden. Jedoch kann in diesem Fall CAT*, sowie CAT, lediglich für kleine Bitraten mit JPEG2000 konkurrieren.

JPEG2000: 0.18 bpp, 32.77 dB

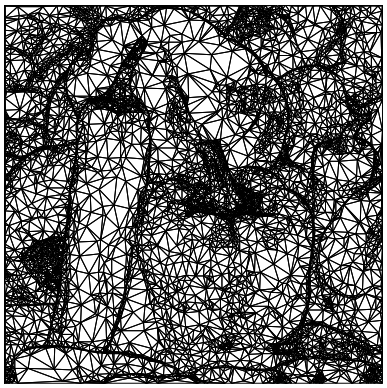
JPEG2000: 0.28 bpp, 34.16 dB

CAT*: 0.18 bpp, 33.11 dB

CAT*: 0.28 bpp, 34.67 dB



CAT*: Delaunay Triangulierung



CAT*: Delaunay Triangulierung

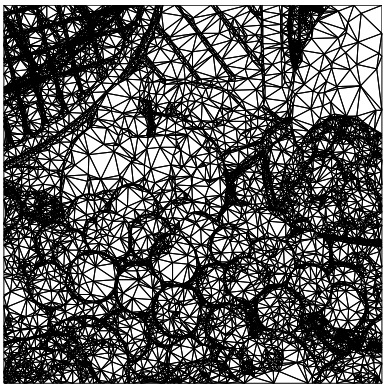
Abbildung 4.5: Vergleich der Kompressionsmethoden JPEG2000 und CAT*; dargestellt sich die Testbilder Lena und Peppers mit neidrigen Bitraten und dem resultierenden PSNR und die zugehörige Delaunay Triangulierung.

JPEG2000: 0.38 bpp, 35.22 dB

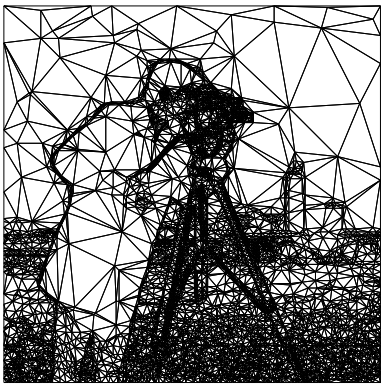
JPEG2000: 0.83 bpp, 34.85 dB

CAT*: 0.38 bpp, 35.86 dB

CAT*: 0.83 bpp, 35.07 dB



CAT*: Delaunay Triangulierung



CAT*: Delaunay Triangulierung

Abbildung 4.6: Vergleich der Kompressionsmethoden JPEG2000 und CAT*; dargestellt sich die Testbilder Lena und Peppers mit neidrigen Bitraten und dem resultierenden PSNR und die zugehörige Delaunay Triangulierung.

(a) **JPEG2000.** PSNR: 32.48; bpp: 0.18

(b) **CAT*.** PSNR: 33.54; bpp: 0.18

Abbildung 4.7: Testbild *Peppers*. Vergleich zwischen JPEG2000 und der Kompressionsmethode CAT*

(a) **JPEG2000.** PSNR: 31.55; bpp: 0.17

(b) **CAT*.** PSNR: 33.14; bpp: 0.17

Abbildung 4.8: Testbild *Fruits*. Vergleich zwischen JPEG2000 und der Kompressionsmethode CAT*

(a) **JPEG2000.** PSNR: 25.48; bpp: 0.17 (b) **CAT*.** PSNR: 27.55; bpp: 0.17

Abbildung 4.9: Testbild *Camera*. Vergleich zwischen JPEG2000 und der Kompressionsmethode CAT*

Kapitel 5

Schlussbemerkung

Als Grundlage dieser Arbeit diente die Aussage, dass die anisotropen Triangulierungen für breite Klassen von Funktionen gegenüber isotropen Methoden höhere Approximationsraten erzielen können. Im Kapitel 1 wurde dies diskutiert und die relevanten Begriffe eingeführt. Insbesondere wurde für Bilder mit überwiegender Kanteninformation gezeigt, dass anisotrope Triangulierungen im Vergleich zu den waveletbasierten Approximationsmethoden deutlich bessere Ergebisse erzielen. Aufbauend auf diesen Erkenntnissen wurde das auf der anisotropen Delaunay Triangulierung basierende, kontextuelle Bildkompressionsverfahren CAT beschrieben, das bereits gute Kompressionsraten im Vergleich zum *state of the art* waveletbasierten Kompressionsverfahren JPEG2000 liefert.
Da die unterliegenden Kontexte teilweise ad hoc gewählt sind, wurde im Rahmen dieser Arbeit das Kompressionsverfahren CAT erweitert und neue kontextuelle Kompressionsmethoden entwickelt. Dazu wurde die starke Heterogenität der anisotropen Triangulierungen genutzt, die mit Hilfe des Adaptive Thinning Algorithmus erzeugt werden. Basierend auf diesen heterogenen Strukturen wurden die neuen Kontexte für die kontextuelle Kompression erarbeitet.

Entstanden ist das neue Bildkompressionsverfahren CAT*, das im Rahmen dieser Diplomarbeit entwickelt und implementiert wurde. Mit dem Kompressionsverfahren CAT* werden deutliche Verbesserungen der Qualität und der Kompressionsraten bezüglich CAT und JPEG2000 erzielt. Für stark geometrische, kantenreiche und strukturarme Bilder ist das Kompressionsverfahren CAT* den untersuchten Methoden CAT und JPEG2000 überlegen. Auch für strukutrreiche Bilder können durch die neu gewählten Kontexte in CAT*, im Vergleich zu CAT und JPEG2000 bei niedriger Bitrate, bessere Kompressionsraten bei gleichbleibender Qualität erzielt werden. Insbesondere für die Grauwerte liefert die neu entwickelte Kompressionsmethode gute Ergebnisse. Da für eine gute Auflösung der Strukturen von strukturreichen Bildern eine hohe Anzahl signifikanter Pixel für die Verfahren CAT und CAT* benötigt werden, ist die waveletbasierte Kompressionsmethode

JPEG2000 für hohe Bitraten die bessere Wahl.

Im Allgemeinen stellt CAT* somit eine gutes alternatives Verfahren dar, dass in der Bildverarbeitung und auch speziell in der medizinischen Bildverarbeitung durchaus als gängiges Verfahren angewendet werden kann, da es den Anforderungen, wie eine hohe Kompressionrate und eine gute Qualtität der Rekonstruktion, genügt. In gewisser Weise wurde dies jedoch durch eine größere Komplexität und eine erhöhte Laufzeit des Verfahrens CAT* erkauft. Dies ist ein Punkt, der zukünftig durch eine effizientere Implementierung noch verbessert werden könnte. Des Weiteren könnten in der Zukunft noch weitere Kompressionsmethoden für die Pixel Positionen als auch für die Grauwerte entwickelt werden, um den Speicherplatzbedarf, bei gleichbleibender Qualität der Rekonstruktion, weiter zu minimieren:

1. Die Kompressionsmethode für die Grauwerte könnte weiter verfeinert werden, indem man die Variation zwischen dem Grauwert des aktuellen Knotens und dem gemittelten Grauwert der Knoten berechnet, die entlang der längsten kausalen Kante und der zweitlängsten kausalen Kante dem aktuellen Knoten gegenüberliegen. Damit ließe sich ein neues Alphabet für die Kodierung der Grauwerte festlegen. Die Einteilung in die Kontextklassen würde wiederum von der Länge der längsten Kante abhängen.

2. Eine neue Methode könnte entwickelt werden, bei der wir uns entlang der jeweils längsten Kante bezüglich des aktuellen Knotens bewegen und die Grauwertvariationen der gegenüberliegenden Knoten für die Festlegung des Alphabets verwenden. Auch hier würde die Länge der längsten Kante des aktuellen Knotens die Kontextklasse festlegen. Dabei muss beachtet werden, dass für jeden Knoten eine Grauwertvariation bestimmt werden kann.

3. Des Weiteren könnte die Kompressionsmethode für die Pixel Positionen verbessert werden, indem die Reihenfolge der Kodierung geändert wird. Man könnte versuchen das Bild nicht zeilen- oder spaltenweise zu scannen, sondern von außen nach innen. D.h. man beginnt in einer Ecke des Bildes und arbeitet sich spiralförmig entlang der Bildkanten in die Mitte des Bildes vor. Die umgekehrte Reihenfolge könnte ebenfalls untersucht werden.

Mein Dank gilt Prof. Dr. Rupert Lasser und speziell Dr. Laurent Demaret für die interessante Aufgabenstellung und die sehr gute Betreuung während der Erstellung dieser Diplomarbeit.

Abbildungsverzeichnis

1.1 Schädel-Computertomographie, http://www.givision.ch/ 6
1.2 Eine konforme und ein nicht-konforme Triangulierung 8
1.3 Vier unterschiedliche aspect ratios 10
1.4 Die abgebildete Triangulierung ist eine Delaunay Triangulierung der Knotenmenge 11
1.5 Eine gleichmäßige, isotrope und anisotrope Triangulierungen . . . 12

2.1 Lena, eine Delaunay Triangulation 35
2.2 Eine Zelle der Delaunay Triangulierung 36
2.3 Kontextbox der bestehenden Methode für die Kompression der Pixel Positionen 43
2.4 Drunken Sunset 44

3.1 Signifikante Pixel nach Adaptive Thinning 49
3.2 Die verschiedenen Kategorien der Bildbereiche 50
3.3 Orientierung der Sequenz S für die neu entwickelten Methoden . . 54
3.4 Kontextbox der neu entwickelten Kompressionsmethode 2 56
3.5 Kontextbox der neu entwickelten Kompressionsmethode 3 57
3.6 Lena 512x512, farbige Darstellung der Einteilung in die Kontextklassen für $\max = 12$ und $b = 8$. 59
3.7 Kontextbox der neu entwickelten Kompressionsmethode 4 60
3.8 Kontextbox der neu entwickelten Kompressionsmethode 5 62
3.9 Lena 512x512, farbige Darstellung für die Einteilung der Grauwertvariationen in die Kontextklassen; rot: Klasse M_0, blau: Klasse M_1, grün: Klasse M_2 68
3.10 Lena, 512×512, kritische Kanten der Delaunay Triangulierung $\mathcal{D}(Y)$; rot: $|v| > 3$, grün: $|v| \leq 3$ 69
3.11 Drei Bilder der Größe 512×512: **(a)** *Lena*; **(b)** *Peppers*; **(c)** *Fruits*; Ein Bild der Größe 256×256: **(d)** *Camera*. 73
3.12 Lena/Peppers; Reduzierung der Kodierungskosten der Pixel Positionen bezüglich CAT 75

3.13 Fruits/Camera; Reduzierung der Kodierungskosten der Pixel Positionen bezüglich CAT 77
3.14 Lena/Peppers; Reduzierung der Kodierungskosten der Grauwerte bezüglich CAT 80
3.15 Fruits/Camera; Reduzierung der Kodierungskosten der Grauwerte bezüglich CAT 82

4.1 Vergleich der Bilder Lena, Camera und Fruits: JPEG2000, CAT und CAT* (bpp vs PSNR); die durchbrochene Linie: JPEG2000, blaue Linie: CAT, rote Linie: CAT* 88
4.2 Vergleich der Bilder Lena, Camera und Fruits: JPEG2000, CAT und CAT* (PSNR vs bpp); blaue durchbrochene Linie: JPEG2000, blaue Linie: CAT, rote Linie: CAT* 89
4.3 Vergleich der Kodierungskosten der Pixel Positionen für Lena, Peppers, Camera und Fruits: CAT und CAT* (Anzahl signifikante Pixel vs Kosten) 90
4.4 Vergleich der Kodierungskosten der Grauwerte für Lena, Peppers, Camera und Fruits: CAT und CAT* (Anzahl signifikante Pixel vs Kosten) 91
4.5 Vergleich der Kompressionsmethoden JPEG2000 und CAT*; dargestellt sich die Testbilder Lena und Peppers mit den neidrigen Bitraten und dem resultierenden PSNR und die zugehörige Delaunay Triangulierung. 92
4.6 Vergleich der Kompressionsmethoden JPEG2000 und CAT*; dargestellt sich die Testbilder Fruits und Camera mit mittleren und hohen Bitraten und dem resultierenden PSNR und die zugehörige Delaunay Triangulierung. 93
4.7 Testbild *Peppers*. Vergleich zwischen JPEG2000 und der Kompressionsmethode CAT* 94
4.8 Testbild *Fruits*. Vergleich zwischen JPEG2000 und der Kompressionsmethode CAT* 94
4.9 Testbild *Camera*. Vergleich zwischen JPEG2000 und der Kompressionsmethode CAT* 95

Tabellenverzeichnis

1.1 Häufigkeitstabelle 31

3.1 Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$; Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 1 54

3.2 Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$; Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 2 56

3.3 Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$, Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 3 58

3.4 Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$, Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 4 61

3.5 Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$, Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 5 63

3.6 Lena (512x512), $|Y| = 12144$, $b = 8$, max $= 12$, Häufigkeitsverteilung der signifikanten und entfernten Pixel bezüglich der Kontextklassen mit den approximierten Kosten für die Kontextklassen für Kompressionsmethode 6 64

3.7 Lena (512x512), $|Y| = 8144$, $q = 8$, Häufigkeitsverteilung der Grauwertvariationen, Kompressionsmethode 1 und 2 72

3.8 Lena (512x512), $b = 8$, max $= 12$, Kodierungskosten der Pixel Positionen für CAT und die neu entwickelten Methoden; fett: Methode mit den geringsten Kosten; kursiv: Methode mit den höchsten Kosten; rechts: prozentuale Verbesserung der besten Methode bezüglich CAT 74
3.9 Peppers (512x512), $b = 8$, max $= 12$, Kodierungskosten der Pixel Positionen für CAT und die neu entwickelten Methoden 4 und 6; fett: Methode mit den geringsten Kosten; kursiv: Methode mit den höhsten Kosten; rechts: prozentuale Verbesserung der besten Methode bezüglich CAT 75
3.10 Fruits (512x512), $b = 8$, max $= 12$, Kodierungskosten der Pixel Positionen für CAT und die neu entwickelten Methoden 2 und 5; fett: Methode mit den geringsten Kosten; kursiv: Methode mit den höhsten Kosten; rechts: prozentuale Verbesserung der besten Methode bezüglich CAT 76
3.11 Camera (256x256), $b = 7$, max $= 5$, Kodierungskosten der Pixel Positionen für CAT und die neu entwickelten Methoden 2 und 6; fett: Methode mit den geringsten Kosten; kursiv: Methode mit den höhsten Kosten; rechts: prozentuale Verbesserung der besten Methode bezüglich CAT 77
3.12 Lena (512x512), $q = 8$, Kodierungskosten der Grauwerte 78
3.13 Peppers (512x512), $q = 8$, Kodierungskosten der Grauwerte . . . 79
3.14 Fruits (512x512), $q = 8$, Kodierungskosten der Grauwerte 81
3.15 Camera (256x256), $q = 8$, Kodierungskosten der Grauwerte für die Kompressionsmethode CAT und die neu entwickelten Kompressionsmethodne 1 und 2 81

4.2 Vergleich der Testbilder Lena, Fruits, Peppers und Camera in Bezug auf JPEG2000, CAT und CAT*. Dargestellt ist die Bitrate bpp in Abhängigkeit vom leicht variierendem PSNR 86
4.1 Vergleich der Testbilder Lena, Fruits, Peppers und Camera in Bezug auf JPEG2000, CAT und CAT*. Dargestellt ist der PSNR in Abhängigkeit der Bitrate (bpp) 87

Literaturverzeichnis

[1] G.E. Blelloch: Introduction to Data Compression. Computer Science Department, Carnegie Mellon University, October 2001

[2] M. Christoph and K. Sergey: Adaptive LZW-JPEG-Komprimierung, Kompressionsverfahren für medizinische Bilder. http://www.aerzteblatt.de/v4/archiv/artikel.asp?id=24538

[3] B. Forster: Lossy Bildkompression, Techniken und Bildqualität. Technische Universität München, 2008.

[4] L. Demaret, N. Dyn, and A. Iske: Image compression by linear splines over adaptive triangulations. Signal Processing **86**(7), July 2006, 1604-1616.

[5] L. Demaret, A. Iske, and W. Khachabi: Contextual Image Compression from Adaptive Sparse Data Representation.

[6] L. Demaret and A. Iske: Anisotropic Triangulation Methods in Adaptive Image Approximation.

[7] L. Demaret and A. Iske: Adaptive image approximation by linear splines over locally optimal Delaunay Triangulation. IEEE Signal Processing Letters **13**(5), May 2006, 281-284.

[8] L. Demaret, N. Dyn, M.S. Floater, and A. Iske: Adaptive thinning for terrain modelling and image compression. In: *Advances in Multiresolution for Geometric Modelling*, N.A. Dodgson, M.S. Floater, and M.A. Sabin (eds.), Springer-Verlag, Berlin, 2004, 321–340.

[9] E. Bodden, M. Clasen, and J. Kneis: Arthmetic Coding revealed - A guided tour from theory to praxis. Sable Research Group, School of Computer Science, McGill University, SABLE-TR-2007-5, Montréal, CA, 2007.

[10] I.H. Witten, R.M. Neal, and J.G. Cleary: Arithmetic Coding for Data Compression. Department of Computer Science, University of Calgary, Communications of the ACM, Calgary, CA, 1987.

[11] A. Said: Introduktion to Arithmetic Coding - Theorie and Practice. Hewlett-Packard Laboratories Report, HPL-2004-76, Palo Alto, CA, 2004.

[12] J.R. Shewchuk: Delaunay Refinement Mesh Generation. School of Computer Science, Carnegie Mellon University, CMU-CS-97-137, Pittsburgh, PA, 1997.

[13] L. Demaret and A. Iske: Scattered data coding in digital image compression. In: *Curve and Surface Fitting: Saint-Malo 2002*, A. Cohen, J.-L. Merrien, and L.L. Schumaker (eds.), Nashboro Press, Brentwood, 2003, 107–117.

[14] L. Demaret, A. Iske, and W. Khachabi: Sparse representation of video data by adaptive tetrahedralisations. In: *Locally Adaptive Filters in Signal and Image Processing*, L. Florack, R. Duits, G. Jongbloed, M.-C. van Lieshout, and L. Davies (eds.) Springer, Dordrecht, 2010, 197–220.

[15] J. Shewchuk: What is a good linear finite element? Interploation, conditioning, anisotropy and quality measures. Preprint, December 2002.

[16] D. Breass: Finite Elements. Theory, Fast Solvers and Applications in Solid Mechanics. 3rd edition, Cambridge University Press, Cambridge, UK, 2007.

[17] J. Bramble und S. Hilbert: Estimation of linear functionals on Sobolev spaces with application to Fourier transforms and spline interpolation. SIAM J. Numer. Anal. 7(1), 1970.

[18] B. Karavainov und P. Petrushev: Nonlinear piecewise polynomial approximation beyond Besov spaces. Appl. Comput. Harmon. Anal. 15(3), 2003, 177-223.

[19] D. Mumford und J. Shah: Optimal approximation of piecewise smooth functions and associated variational problems. Comm. in Pure and Appl. Math. 43, 1989, 577-685.

[20] S. Dekel, D. Leviatan, and M. Sharir: On bivariate smoothness spaces associated with nonlinear approximation. Constructive Approximation **20**, 2004, 625–646.

[21] N. Dyn, M. Floater, and A. Iske: Adaptive Thinning for Bivariate Scattered Data. JCAM 145, 2002, 505-517.

[22] L. Demaret und A. Iske: Advances in digital image compression by adaptive thinning. Annals of the MCFA 3, 105-109.

[23] D. Taubmann: http://www.kakadusoftware.com.