

Technische Universität München
Fakultät für Mathematik

**Das Potts-Modell zur
Segmentation verrauschter
Daten**

Diplomarbeit von Katrin Wicker

Aufgabensteller: PD Dr. Hartmut Führ
Betreuer: PD Dr. Volkmar Liebscher
Abgabetermin: 15. Juli 2004

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

München, den 8. Juni 2004,

Inhaltsverzeichnis

1	Zusammenfassung	4
2	Einleitung	7
3	Das Eindimensionale Potts-Funktional und seine Minimierung	10
3.1	Definition des Potts-Funktional	10
3.2	Partitionen	11
3.3	Existenz des Minimierers	14
3.4	Eindeutigkeit des Minimierers	19
3.5	Algorithmus zur Minimierung	23
3.6	Effiziente Momentenberechnung über Intervallen	24
4	Das Zweidimensionale Potts-Funktional und seine Minimierung	27
4.1	Definition des Potts-Funktional	27
4.2	Das Hierarchische Modell	31
4.2.1	Partitionen	31
4.2.2	Existenz des Minimierers	34
4.2.3	Eindeutigkeit des Minimierers	37
4.2.4	Algorithmus zur Minimierung	38

4.2.5	Effiziente Momentenberechnung über Intervallen	40
4.3	Das Wedgelet Modell	44
4.3.1	Partitionen	44
4.3.2	Existenz des Minimierers	51
4.3.3	Eindeutigkeit des Minimierers	52
4.3.4	Algorithmus zur Minimierung	53
4.3.5	Effiziente Momentenberechnung über Intervallen	57
4.3.5.1	Interpolations-Methode	62
4.3.5.2	Unterteilung in diskrete Linien	65
4.3.5.3	Komplexität	69
5	Software	70
6	Anwendungen	77
6.1	Kompression	77
6.2	Laufzeitvergleich	90
6.3	Skalierung des Parameters γ	92
6.4	Wahl des Parameters γ	96
6.5	Beispiele	98
	Literatur	101

1 Zusammenfassung

Das Ziel dieser Arbeit ist eine Untersuchung des Potts-Modells zur Segmentation verrauschter Daten über ein- und zweidimensionaler endlicher Indexmenge $S \subset \mathbb{Z}^2$. Im Vordergrund steht Entwicklung und Analyse verschiedener Minimierungsalgorithmen für das *Potts-Funktional* $H_\gamma(\cdot|y) : \mathbb{R}^S \rightarrow \mathbb{R}$,

$$x \mapsto \gamma \cdot |\{(s, t) \in S \times S : \|s - t\| = 1, x_s \neq x_t\}| + \sum_{s \in S} (x_s - y_s)^2,$$

zu gegebenen Daten $y \in \mathbb{R}^S$ und Parameter $\gamma > 0$.

Der erste Term, der *Regularisierungsterm*, zählt die Anzahl ungleicher benachbarter Werte in x . Der zweite Term, der *Datenterm*, misst den Abstand von x zu den Daten y . Eine Minimierung des Funktionals bedeutet also, eine Darstellung \hat{x} zu finden, die die beiden konkurrierenden Forderungen nach Glattheit einerseits und Datentreue andererseits bestmöglich erfüllt.

Eine solche Minimierung kann im allgemeinen Fall nur approximativ, mit stochastischen Algorithmen vorgenommen werden (siehe z.B. G. WINKLER (2002)). Diese Verfahren haben jedoch einige Nachteile: Zum einen können die Laufzeiten solcher Algorithmen sehr lang sein. Die berechneten Minima können (nach endlicher Laufzeit) zum anderen nicht mit Sicherheit als globale Minima identifiziert werden. Außerdem sind aufgrund der stochastischen Natur solcher Algorithmen Ergebnisse meist nicht reproduzierbar und vergleichbar.

In dieser Arbeit werden daher exakte Minimierungsalgorithmen angegeben, die in akzeptabler Zeit ausgeführt werden können. Im eindimensionalen Fall $S \subset \mathbb{Z}$ wird der in G. WINKLER und V. LIEBSCHER (2002) vorgeschlagene Minimierungsalgorithmus, der auf dynamischer Programmierung beruht, angegeben. In diesem Fall gibt es eine eins-zu-eins Beziehung zwischen den Sprüngen auf x und den Segmenten der durch x induzierten Partition, den Bereichen von S nämlich, auf denen x einen konstanten Wert annimmt. Um die Analyse der Algorithmen einzuleiten, wird in dieser Arbeit zuerst im eindimensionalen Fall diese Analogie aufgezeigt.

Für höhere Dimensionen ist bisher kein Algorithmus zur Minimierung des

Potts-Funktional bekannt. Ein Grund dafür ist die völlig andere geometrische Situation und die damit verbundene Unmöglichkeit, Partitionen im zweidimensionalen rekursiv aufzuzählen. Daher werden für den zweidimensionalen Fall in dieser Arbeit die Menge der erlaubten Partitionen auf zwei verschiedene Weisen *eingeschränkt* und es werden für diese Modifikationen des Potts-Modells exakte Minimierungsalgorithmen angegeben.

Die erste vorgestellte Variante zur Segmentation eines Bildes (das *Hierarchische Modell*) beruht auf der Übertragung der eindimensionalen Geometrie auf die Koordinatenachsen im zweidimensionalen Modell. Die dabei entstehenden Partitionen stellen eine Einteilung des Bildes in vertikale Streifen dar, die jeweils noch durch horizontale Linien unterteilt sein können. Ein Algorithmus zur Minimierung des Funktional im Hierarchischen Modell wird ebenfalls angegeben, dabei kann der eindimensionale Minimierungsalgorithmus für höherdimensionale Daten rekursiv eingesetzt werden. Zur effizienten Implementierung dieses Algorithmus wird darüberhinaus ein Summierungstrick aufgezeigt, der Ermittlung von Momenten in Rechtecken in $O(1)$ ermöglicht.

Die zweite Variante (das *Wedgelet Modell*) erklärt Segmentationen durch Partitionen in Rechtecke mit nachfolgender Unterteilung durch Einziehen zusätzlicher Kanten (*Wedges*). Es wird ein effizienter Algorithmus zur Minimierung eines zu solchen Partitionen gehörigen Funktional angegeben. Die Stärke des vorgestellten Algorithmus besteht in einer äußerst effizienten Berechnungsmethode für die minimierenden Wedges zu vorgegebenem Rechteck. Kernpunkt dabei ist die durch Vorgabe einer endlichen Winkelmenge ermöglichte Tabellierung von kumulativen Matrizen und die damit ermöglichte Ermittlung von Momenten zu jeder Kante in $O(1)$. Der Minimierungsalgorithmus hat für eine feste Winkelmenge Komplexität $O(|S|)$.

Die Minimierungsalgorithmen liegen sowohl für das Hierarchische Modell als auch für das Wedgelet Modell implementiert bereit. Neben den Algorithmen an sich und den dazugehörigen Datenstrukturen wurde auch ein graphisches Benutzerinterface erstellt. Die Software wird unter <http://www.antsinfields.de> öffentlich zugänglich sein.

Der Minimierungsalgorithmus für das Wedgelet-Modell wurde noch einigen Untersuchungen unterzogen. Um diesen auf seine Tauglichkeit als Kompressionsalgorithmus zu testen, wird der *PSNR* - Wert als objektives Maß für

die Güte von komprimierten Bildern für einige Beispiele berechnet und gegen Partitionsgröße, die als Hinweis auf Kompressionsraten dienen kann, aufgetragen. Ein Laufzeitvergleich mit der SlowWedgeletTransform von *BeamLab* (BEA (8. 6. 2004)) wurde durchgeführt. Für die Vergleichbarkeit der Minimierung auf verschieden großen Bildern wird eine Skalierung von γ vorgeschlagen. Schließlich wird noch auf die Problematik einer “optimalen” Wahl des Parameters γ eingegangen.

Diese Zusammenfassung abschließend möchte ich mich bei Hartmut Führ und Volkmar Liebscher für die Vergabe des interessanten Themas und für die nette Betreuung bedanken.

Mein Dank gilt außerdem allen weiteren netten und hilfsbereiten Mitarbeitern und Mitarbeiterinnen am Institut für Biomathematik und Biometrie der GSF.

2 Einleitung

Bevor das Potts-Funktional erklärt werden kann, werden einige wichtige Notationen und Definitionen benötigt. Die Begriffe und Notationen werden zunächst für allgemeine Dimensionen eingeführt, in den späteren Kapiteln wird dann zwischen ein- und zweidimensionalen Fällen unterschieden.

Notation und Definition 2.1 Sei $d \in \mathbb{N}$.

Im Folgenden bezeichne S eine Teilmenge von \mathbb{Z}^d , die **Indexmenge** oder auch **Gitter** genannt wird.

Eine Teilmenge E von \mathbb{R} wird als **Zustandsraum** bezeichnet, die Elemente $e \in E$ heißen **Zustände** (**Intensitäten** oder auch **Grauwerte**).

Die Menge

$$\mathbb{X} := E^S = \{(x_s)_{s \in S} : x_s \in E\}$$

heißt im weiteren Text **Konfigurationsraum**, die x_s werden als **Pixel** bezeichnet und ein $x \in \mathbb{X}$ heißt **Konfiguration** (kontextabhängig heißt x auch **Signal**, **Daten** oder **Bild**).

Zwei Elemente $s, t \in S$ werden **benachbart** genannt, in Zeichen

$$s \sim t,$$

wenn $\|s - t\|_d = 1$ mit $\|s\|_d := \sum_{i=1}^d |s_i|$.

Definition 2.2 Gegeben sei ein Parameter $\gamma > 0$, sowie Daten $y \in \mathbb{X}$.

Das **Potts-Funktional** ist gegeben durch die Abbildung

$$\begin{aligned} H_\gamma : \mathbb{X} \times \mathbb{X} &\rightarrow \mathbb{R}, \\ (x, y) &\mapsto G_\gamma(x) + D(x|y). \end{aligned} \tag{2.1}$$

Der erste Term $G_\gamma(x)$ des Potts-Funktionals ist der **Regularisierungsterm**

$$G_\gamma : \mathbb{X} \rightarrow \mathbb{R}, \quad G_\gamma(x) = \gamma \sum_{\substack{i \sim j \\ i, j \in S}} \delta_{(x_i, x_j)}. \tag{2.2}$$

Der Regularisierungsterm bewertet die Anzahl der Sprünge zwischen Nachbarn, also die Anzahl unterschiedlicher Werte $x_s \neq x_t$ für $s \sim t$.

Als zweiter Term ist hier ein quadratischer **Datenterm** $D(x|y)$ vorgesehen, nämlich

$$D : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}, \quad D(x|y) = \sum_{s \in S} (x_s - y_s)^2. \quad (2.3)$$

Dieser misst die Datentreue des Signals x zu den Daten y .

Im Potts-Funktional H_γ steuert die Konstante $\gamma > 0$ somit das Gewicht von Datentreue gegenüber Regularität.

Das Ziel ist die Minimierung der Funktion

$$H_\gamma(x|y) = G_\gamma(x) + D(x|y) = \gamma \sum_{\substack{i \sim j \\ i, j \in S}} \delta_{(x_i, x_j)} + \sum_{s \in S} (y_s - x_s)^2$$

zu gegebenem y in x .

Die Minimierung der Funktion $H_\gamma(x|y)$ bedeutet nämlich, eine Darstellung \hat{x} zu den Daten y zu finden, die die konkurrierenden Forderungen nach wenigen Sprüngen und Datentreue bestmöglich erfüllt.

Eine solche Minimierung kann im allgemeinen Fall approximativ mit Simulated Annealing und ähnlichen stochastischen Algorithmen vorgenommen werden (siehe z.B. G. WINKLER und V. LIEBSCHER (2002)). Diese haben jedoch einige Nachteile: Zum einen können die Laufzeiten solcher Algorithmen sehr lang sein (auch abhängig von der Anzahl $|E|$ der Grauwerte). Die berechneten Minima können (nach endlicher Laufzeit) zum anderen nicht mit Sicherheit als globale Minima identifiziert werden. Außerdem sind aufgrund der stochastischen Natur solcher Algorithmen Ergebnisse meist nicht reproduzierbar und vergleichbar.

In dieser Arbeit werden daher exakte Minimierungsalgorithmen angegeben, die in akzeptabler Zeit berechnet werden können. Dazu sind aber gewisse Einschränkungen und Modifikationen des Potts-Modells nötig. So wird im folgenden Kapitel zunächst die Dimension $d = 1$ betrachtet und sich somit auf eindimensionale Daten beschränkt. Im darauf folgenden Kapitel wird das Potts-Modell modifiziert und ein hierarchisches Vorgehen für die Segmentation zweidimensionaler Bilder angegeben, bei dem der eindimen-

sionale Minimierungsalgorithmus für höherdimensionale Daten rekursiv eingesetzt werden kann. Schließlich wird eine weitere zweidimensionale Variante beschrieben, die geometrische Eigenschaften der Daten durch Wedgelet-Unterteilungen berücksichtigt.

3 Das Eindimensionale Potts-Funktional und seine Minimierung

In diesem Kapitel wird der Algorithmus von G. WINKLER und V. LIEBSCHER (2002) für die exakte Minimierung des Potts-Funktional $H_\gamma(x|y)$ im eindimensionalen Fall hergeleitet und angegeben. Außerdem wird eine auf Partitionen basierende Notation des Potts-Modells angegeben.

3.1 Definition des Potts-Funktional

Für die eindimensionale Betrachtung des Potts-Modell ist die **Indexmenge** S gegeben durch

$$S := \{1, \dots, N\}$$

für ein $N \in \mathbb{Z}$.

Zwei Punkte s und t aus S sind benachbart (in Zeichen: $s \sim t$), wenn gilt:

$$|s - t| = 1.$$

Das **Signal** x nimmt reelle Werte aus dem **Zustandsraum** $\mathbb{X} = \{(x_s)_{s \in S} : x_s \in \mathbb{R}\}$ an.

Das **Potts-Funktional** ist gegeben durch

$$H_\gamma : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}, \quad H_\gamma(x|y) = G_\gamma(x) + D(x|y). \quad (3.4)$$

Die Abweichung der Daten y von dem Signal x wird in dem **Datenterm** $D(x|y)$ durch den quadratischen Abstand gemessen:

$$D : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}, \quad D(x|y) = \sum_{s \in S} (x_s - y_s)^2. \quad (3.5)$$

Für ein $\gamma > 0$ beschreibt die Funktion $G_\gamma(x)$ die **Irregularität** der Repräsentation x durch die Anzahl der benachbarten ungleichen Pixel,

$$G_\gamma : \mathbb{X} \rightarrow \mathbb{R}, \quad G_\gamma(x) = \gamma \sum_{\substack{s \sim t \\ s, t \in S}} \delta_{(x_s, x_t)} = \gamma \cdot |\{s \sim t : x_s \neq x_t\}|. \quad (3.6)$$

Gesucht wird jetzt eine Approximation

$$\hat{x} = (\hat{x}_s)_{s \in S} \in \mathbb{R}^S,$$

welche einerseits nah an den Daten y ist (Datenterm $D(\hat{x}|y)$) und andererseits die Anzahl der Sprünge (Regularisierung $G_\gamma(x)$) klein hält, d. h. eine sparsame Darstellung von y ist. Das Ziel ist also nun, die Funktion $H_\gamma(x|y)$ in x bei festem bekanntem y und festem Parameter γ zu minimieren.

3.2 Partitionen

In diesem Abschnitt wird gezeigt, dass die Minimierung der Funktion $H_\gamma(x|y)$ in x gleichbedeutend ist mit der Suche nach einer optimalen Segmentation, also einer Partition des Indexraumes in Intervalle, die mit reellen Werten versehen sind. Zunächst werden die Begriffe Intervall, Partition und Segmentation definiert.

Definition 3.1 Gegeben sei die Indexmenge $S = \{1, \dots, N\}$, $N \in \mathbb{N}$.

Ein **Intervall** I ist die Menge $[s, t] := \{u \in S : s \leq u \leq t\}$ für $s, t \in S$.

Eine **Partition** P der Indexmenge S ist eine Unterteilung von S in Intervalle $P := \{I_1, \dots, I_n\}$ ($1 \leq n \leq N$) mit

$$I_i = [k_i, k_{i+1}], \quad 1 \leq i \leq n,$$

wobei $0 =: k_0 < k_1 < \dots < k_n = N$.

Mit $|P|$ wird die Anzahl der Intervalle in der Partition P bezeichnet.

Eine **Segmentation** $\mathbf{P} := (P, \mu)$ besteht aus einer Partition $P = \{I_1, \dots, I_n\}$ des Indexraumes S und einem Vektor $\mu \in \mathbb{R}^{|P|}$.

Die Menge aller möglichen Segmentationen des Indexraumes S wird mit $\mathfrak{P}(S)$ bezeichnet.

Für die Intervalle einer Partition P gelten offensichtlich folgende Eigenschaften:

- $I_j \subseteq S$, für $1 \leq j \leq n$,
- $I_1 \cup \dots \cup I_n = S$,
- $I_j \cap I_i = \emptyset$ für alle $j \neq i$.

Ein gegebenes Signal x induziert auf eindeutige Weise eine Partition in n ($1 \leq n \leq N$) Intervalle, nämlich eine Unterteilung der Indexmenge S in Intervalle I_i , auf denen das Signal jeweils den konstanten Wert $\mu_i \in \mathbb{R}$ annimmt ($1 \leq i \leq n$).

Definition 3.2 Eine Segmentation $\mathbf{P} = (P, \mu)$ heißt von $x \in \mathbb{X}$ induziert, wenn einerseits aus $s \in I_i$ folgt $\mu_i = x_s$ und andererseits $\mu_i \neq \mu_{i+1}$ für ($1 \leq i \leq n$).

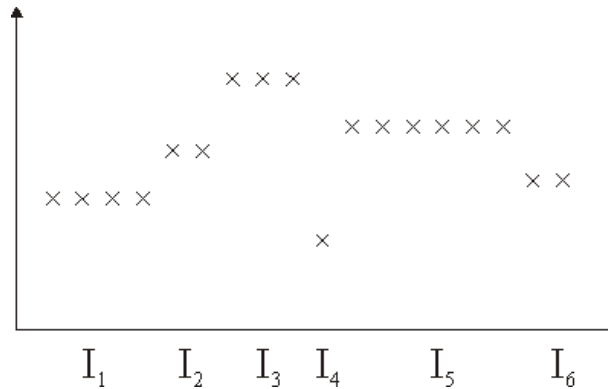


Abb. 1: Partition P in Intervalle I_i , induziert durch ein Signal x mit konstanten Werten μ_i in den Intervallen, $|P| = 6$

Lemma 3.3 *Ein eindimensionales Signal x induziert genau eine Segmentation \mathbf{P} von S .*

Beweis Da die Menge $\{\{s\} : s \in S\}$ zusammen mit $\mu_s = x_s$ eine Segmentation von S darstellt, kann die Existenz einer Segmentation \mathbf{P} konstruktiv gesehen werden, dabei fasst man sukzessive benachbarte Intervalle I_i und I_j mit gleichen Werten $\mu_i = \mu_j$ so lange zusammen bis benachbarte Segmente keine gleichen Pixel enthalten.

Seien nun $\mathbf{P} = (P, \mu)$ und $\mathbf{Q} = (Q, \mu')$ zwei verschiedene durch x induzierte Segmentationen. Dann existieren zwei Intervalle $I \in P$ und $J \in Q$ mit $\emptyset \neq I \cap J \neq I$ und $(I \cap J) \sim (J \setminus I)$, ein klarer Widerspruch zur angegebenen Konstruktion. ■

Die Funktion $H_\gamma(x|y)$ lässt sich als eine Funktion der durch x induzierte Segmentation $\mathbf{P} = (P, \mu)$ auffassen.

Satz 3.4 *Mit einer von x induzierten Segmentation $\mathbf{P} = (P, \mu)$ lässt sich das Potts-Funktional $H_\gamma(x|y)$ wie folgt schreiben:*

$$H_\gamma(x|y) = g_\gamma(P) + d(\mathbf{P}|y).$$

Dabei sind der Regularisierungsterm g_γ und der Datenterm d zur Segmentation \mathbf{P} und Daten y durch

$$g_\gamma(P) := \gamma(|P| - 1) \tag{3.7}$$

und

$$d(\mathbf{P}|y) := \sum_{I \in P} \sum_{s \in I} (y_s - \mu_I)^2, \quad \mu_I = x_s \text{ für ein } s \in I, \tag{3.8}$$

gegeben.

Beweis Für eine feste Segmentation $\mathbf{P} = \{(P_i, \mu_i)_{i=1, \dots, n}\}$ ist die Anzahl der Sprünge des Signals x genau die Anzahl $|P|$ der Intervalle der Partition und es gilt mit Gleichung 3.6:

$$G_\gamma(x) = \gamma \sum_{\substack{i \sim j \\ i, j \in S}} \delta_{(x_i, x_j)} = \gamma \cdot |\{s \sim t : x_s \neq x_t\}| = \gamma(|P| - 1).$$

Da für eine festgelegte Segmentation \mathbf{P} mit der Partition $P = \{I_1, \dots, I_n\}$ die $x_s = \mu_i$ für alle $s \in I_i$ ($1 \leq i \leq n$) sind, gilt für den Datenterm aus Gleichung 3.5:

$$D(x|y) = \sum_{s \in S} (y_s - x_s)^2 = \sum_{I \in \mathbf{P}} \sum_{s \in I} (y_s - x_s)^2 = \sum_{I \in \mathbf{P}} \sum_{s \in I} (y_s - \mu_I)^2.$$

■

Notation 3.5 Das Potts-Funktional für eine feste Segmentation $\mathbf{P} = (P, \mu)$ der Indexmenge S wird mit

$$h_\gamma(\mathbf{P}|y) := g_\gamma(P) + d(\mathbf{P}|y) \quad (3.9)$$

bezeichnet.

3.3 Existenz des Minimierers

Nach dem vorherigen Abschnitt ist die Minimierung von $H_\gamma(x|y)$ bzgl. x also gleichbedeutend mit der Konstruktion einer *optimalen* (d.h. $h_\gamma(\mathbf{P}|y)$ minimierenden) Segmentation \mathbf{P} , sofern benachbarte Intervalle verschiedene Werte annehmen. Aufgrund dieser Restriktion beziehen sich die weiteren Berechnungen auf eine Minimierung von $h_\gamma(\mathbf{P}|y)$, also auf die Suche einer Partition \hat{P} aus der endlichen Menge aller möglichen Partitionen der Indexmenge S .

Notation 3.6 Sei A eine beliebige Teilmenge der Indexmenge $S = \{1, \dots, N\}$. Dann bezeichne

$$\hat{\mu}(A) := \frac{1}{|A|} \sum_{i \in A} y_i, \quad (3.10)$$

und

$$\hat{\sigma}(A) := \sum_{i \in A} (\hat{\mu}(A) - y_i)^2. \quad (3.11)$$

Für das Intervall $Q_l^k := \{l, \dots, k\}$ mit $1 \leq l \leq k \leq N$ bezeichne

$$\mu_l^k := \hat{\mu}(Q_l^k)$$

den empirischen Mittelwert der Daten y , und

$$\sigma_l^k := \hat{\sigma}(Q_l^k)$$

die quadratische Abweichung des Vektors x von den Daten y jeweils im Intervall Q_l^k .

Satz 3.7 Gegeben sei eine Segmentation $\mathbf{P} = (P, \mu)$ der Indexmenge $S = \{1, \dots, N\}$ mit fester Partition $P = (I_1, \dots, I_n)$, $1 \leq n \leq N$, und reellen Werten $\mu = (\mu_1, \dots, \mu_n)$. Der Minimierer der Funktion $h_\gamma(\mathbf{P}|y)$ ist gegeben durch

$$(\hat{\mu}(I_1), \dots, \hat{\mu}(I_n)).$$

Beweis Bei festgelegter Partition P mit Intervallen (I_1, \dots, I_n) ist die Funktion $h_\gamma(\mathbf{P}|y)$ gegeben durch

$$h_\gamma(\mathbf{P}|y) = g_\gamma(P) + d(\mathbf{P}|y) = \gamma|P| + \sum_{i=1}^n \sum_{j \in I_i} (\mu_i - y_j)^2.$$

Da dann $g_\gamma(P)$ konstant ist, ist die Minimierung von $h_\gamma(\mathbf{P}|y)$ somit gleichbedeutend mit der Minimierung des Datenterms $d(\mathbf{P}|y) = \sum_{i=1}^n \sum_{j \in I_i} (\hat{\mu}(I_i) - y_j)^2$. Dieser ist genau dann minimal, wenn jede einzelne Summe $\sum_{j \in I_i} (\hat{\mu}(I_i) - y_j)^2$ für jedes Intervall I minimal ist, also wenn $\sum_{j \in I_i} 2(\hat{\mu}(I_i) - y_j) = 0$ und daraus folgt $|I_i| \cdot \hat{\mu}(I_i) = \sum_{j \in I_i} y_j$. Somit ist jeweils der optimale Wert von $h_\gamma(\mathbf{P}|y)$ das empirische Mittel $\hat{\mu}(I_i)$. ■

Notation 3.8 Der Minimierer von $h_\gamma(\mathbf{P}|y)$ mit $\mathbf{P} = (P, \mu)$ für eine fixierte Partition P mit den Intervallen I_1, \dots, I_n wird mit

$$\hat{\mu}(P) := (\hat{\mu}(I_1), \dots, \hat{\mu}(I_n))$$

bezeichnet.

Satz 3.9 Es existiert ein Minimierer für die Funktion $h_\gamma(\mathbf{P}|y)$.

Beweis Nach Satz 3.7 ist für jede feste Partition der Minimierer bekannt, deshalb muss nur noch über die endliche Menge von möglichen Partitionen der Indexmenge $S = \{1, \dots, N\}$ minimiert werden. Somit ist die Existenz klar. ■

Notation 3.10 Für ein $k \in \{1, \dots, N\}$ bezeichne $\hat{\mathbf{P}}^k$ eine optimale Segmentation zu den Daten (y_1, \dots, y_k) , weiter sei $\hat{\mathbf{P}}_0 = \emptyset$ und $\hat{\mathbf{P}}^1 = \{(I_1^1, \mu_1^1)\}$.

Sei $k, l \in S := \{1, \dots, N\}$ mit $l < k$ und

$$\mathbf{P}^k(l) := (\hat{\mathbf{P}}^l \cup \{(Q_{l+1}^k, \hat{\mu}_{l+1}^k)\}).$$

Kern des Algorithmus für die Minimierung des eindimensionalen Potts-Funktionalis ist nun folgende Beobachtung:

Satz 3.11 Sei $k \in \{1, \dots, N\}$ und \hat{l} sei ein Minimierer von $h_\gamma(\mathbf{P}^k(l)|y)$ in l , also

$$\hat{l} \in \{l \in \{1, \dots, k-1\} : h_\gamma(\mathbf{P}^k(l)|y) \text{ ist minimal}\}.$$

Dann ist $\mathbf{P}^k(\hat{l})$ ein Minimierer der Funktion $h_\gamma(\mathbf{P}^k|y)$, d.h. es gilt

$$h_\gamma(\mathbf{P}^k(\hat{l})|y) = h_\gamma(\hat{\mathbf{P}}^k|y).$$

Für den Beweis wird zunächst folgendes Lemma angegeben:

Lemma 3.12 *Seien \mathbf{P} und \mathbf{P}' Segmentationen von $\{1, \dots, m\}$ bzw. $\{m+1, \dots, N\}$ mit $1 \leq m < N$. Dann gilt*

$$h_\gamma(\mathbf{P} \cup \mathbf{P}'|y) = h_\gamma(\mathbf{P}|y) + h_\gamma(\mathbf{P}'|y) + \gamma.$$

Insbesondere gilt

$$h_\gamma(\mathbf{P} \cup \{(Q_{m+1}^N, \hat{\mu}_{m+1}^N)\}|y) = h_\gamma(\hat{\mathbf{P}}|y) + \gamma + \sigma_{m+1}^N.$$

Beweis

$$\begin{aligned} h_\gamma(\mathbf{P} \cup \mathbf{P}'|y) &= g_\gamma((P_1 \cup P_2)) + d(\mathbf{P} \cup \mathbf{P}'|y) \\ &= \gamma(|P_1 \cup P_2| - 1) + \sum_{I \in (P_1 \cup P_2)} \sum_{j \in I} (\hat{\mu}(I) - y_j)^2 \\ &= \gamma(|P_1| - 1) + \gamma(|P_2| - 1) + \gamma \\ &\quad + \sum_{I \in P_1} \sum_{j \in I} (\hat{\mu}(I) - y_j)^2 + \sum_{I \in P_2} \sum_{j \in I} (\hat{\mu}(I) - y_j)^2 \\ &= g(P_1) + g(P_2) + \gamma + d(\mathbf{P}|y) + d(\mathbf{P}'|y) \\ &= h_\gamma(\mathbf{P}|y) + h_\gamma(\mathbf{P}'|y) + \gamma \end{aligned}$$

$$\begin{aligned} h_\gamma(\mathbf{P} \cup \{(Q_{m+1}^N, \hat{\mu}_{m+1}^N)\}|y) &= h_\gamma(\mathbf{P}|y) + h_\gamma(\{(Q_{m+1}^N, \hat{\mu}_{m+1}^N)\}|y) + \gamma \\ &= h_\gamma(\mathbf{P}|y) + \gamma + \sigma_{m+1}^N \end{aligned}$$

■

Jetzt folgt der Beweis des Satzes 3.11.

Beweis Aus Satz 3.7 und Lemma 3.12 folgt für eine optimale Partition

$$\begin{aligned}
h_\gamma(\hat{\mathbf{P}}^N|y) &= \min_{\tilde{\mathbf{P}} \in \mathfrak{P}(S)} h_\gamma(\tilde{\mathbf{P}}|y) \\
&= \min_{k \in \{1, \dots, N\}} \min_{\tilde{\mathbf{P}} \in \mathfrak{P}(S)} \{h_\gamma(\tilde{\mathbf{P}}, \{(I_k^N, \hat{\mu}_k^N)\}|y)\} \\
&= \min_{k \in \{1, \dots, N\}} \min_{\tilde{\mathbf{P}} \in \mathfrak{P}(\{1, \dots, k\})} (h_\gamma(\tilde{\mathbf{P}}|y) + \gamma + \sigma_k^N) \\
&= \min_{k \in \{1, \dots, N\}} \left[\left(\min_{\tilde{\mathbf{P}} \in \mathfrak{P}(S)} h_\gamma(\tilde{\mathbf{P}}|y) \right) + \gamma + \sigma_k^N \right] \\
&= \min_{k \in \{1, \dots, N\}} (h_\gamma(\hat{\mathbf{P}}^k|y) + \gamma + \sigma_k^N) \\
&= h_\gamma(\mathbf{P}_{\hat{k}}^N|y).
\end{aligned}$$

Der Schluss des Beweises erfolgt per Induktion. ■

Satz 3.13 *Die Minimierer von $h_\gamma(\mathbf{P}|y)$ sind genau die von den Minimierern von $H_\gamma(x|y)$ induzierten Segmentationen.*

Beweis Zu zeigen ist nach Satz 3.4 nur, dass für eine $h_\gamma(\mathbf{P}|y)$ minimierende Segmentation $\mu_k \neq \mu_{k+1}$ für benachbarte Intervalle I_k und I_{k+1} gilt ($1 \leq k \leq n-1$). Seien $I_k, I_{k+1} \in P$ und $\mu_k = \mu_{k+1}$. Dann gilt mit $\gamma > 0$

$$\begin{aligned}
h_\gamma(\mathbf{P}|y) &= \sum_{i \in \{1, \dots, n\} \setminus \{k, k+1\}} \sigma(I_i) + (n-2)\gamma + \sigma(I_k) + \gamma + \sigma(I_{k+1}) \\
&> \sum_{i \in \{1, \dots, n\} \setminus \{k, k+1\}} \sigma(I_i) + (n-2)\gamma + \underbrace{\sigma(I_k \cup I_{k+1})}_{=\sigma(I_k) + \sigma(I_{k+1})}.
\end{aligned}$$

Somit wäre $h_\gamma(\mathbf{P}'|y) < h_\gamma(\mathbf{P}|y)$ für eine Segmentation \mathbf{P}' , in der die Intervalle I_k und I_{k+1} verschmolzen werden, und dann wäre \mathbf{P} nicht optimal. ■

3.4 Eindeutigkeit des Minimierers

Gezeigt wird, dass der Minimierer \hat{x} des Potts-Funktional $H_\gamma(x|y)$ eindeutig ist für Lebesgue-fast alle Daten y . Bei der Herleitung des Beweises wurde sich weitgehend an A. KEMPE (2004) orientiert.

Satz 3.14 *Für jedes $\gamma > 0$ gibt es eine Lebesgue-Nullmenge $N_\gamma \subseteq \mathbb{X}$, so dass für alle $y \notin N_\gamma$ die Funktion $H_\gamma(x|y)$ einen eindeutigen Minimierer \hat{x} besitzt.*

Für den Beweis werden folgende Beobachtungen benötigt:

1. Nach Satz 3.4 lässt sich der Datenterm als

$$d(\mathbf{P}|y) = \sum_{I \in P} \sum_{s \in I} (y_s - \mu_I)^2$$

schreiben und nach Satz 3.7 ist die Familie der Minimierer $\hat{\mu}(P)$ durch

$$\hat{\mu}_I = \frac{1}{|I|} \sum_{s \in I} y_s \quad (3.12)$$

gegeben.

2. Sei $y \in \mathbb{X}$ und es gelte $H_\gamma(x|y) = H_\gamma(x'|y)$ für $x \neq x'$. Die von x bzw. x' induzierten Segmentationen seien $\mathbf{P} = (P, \mu_P)$ bzw. $\mathbf{P}' = (P', \mu_{P'})$. Dann folgt:

$$\left(\sum_{I \in P} \sum_{s \in I} (y_s - \mu_I)^2 \right) - \left(\sum_{J \in P'} \sum_{s \in J} (y_s - \mu'_J)^2 \right) = g_\gamma(P') - g_\gamma(P).$$

3. Seien nun $\mathbf{P} \neq \mathbf{Q}$ zwei Segmentationen und seien μ_P und μ_Q die jeweils dazugehörigen Minimierer des Datenterms.

Definiert wird die Funktion

$$f_\gamma^{\mathbf{P}, \mathbf{Q}} : \mathbb{X} \rightarrow \mathbb{R}, \quad f_\gamma^{\mathbf{P}, \mathbf{Q}}(y) = \frac{1}{\gamma} \left(\sum_{I \in P} \sum_{s \in I} (y_s - \mu_I)^2 - \sum_{J \in Q} \sum_{s \in J} (y_s - \mu'_J)^2 \right)$$

Beweis Mit Hilfe von Lemma 3.15 und den binomischen Formeln lässt sich $f_\gamma^{\mathbf{P},\mathbf{Q}}(y)$ als quadratische Form

$$\begin{aligned}
f_\gamma^{\mathbf{P},\mathbf{Q}}(y) &= \frac{1}{\gamma} \left(\sum_{I \in \mathbf{P}} \sum_{s \in I} (y_s - \mu_I)^2 - \sum_{J \in \mathbf{Q}} \sum_{s \in J} (y_s - \mu_J)^2 \right) \\
&= \frac{1}{\gamma} \left(\sum_{I \in \mathbf{P}} \sum_{s \in I} (y_s^2 - 2y_s \mu_I + \mu_I^2) - \sum_{J \in \mathbf{Q}} \sum_{s \in J} (y_s^2 - 2y_s \mu_J + \mu_J^2) \right) \\
&= \frac{1}{\gamma} \left(\sum_{I \in \mathbf{P}} \sum_{s \in I} \mu_I - \sum_{J \in \mathbf{Q}} \sum_{s \in J} \mu_J \right) \\
&= \frac{1}{\gamma} \left(\sum_{J \in \mathbf{Q}} |J| \mu_J^2 - \sum_{I \in \mathbf{P}} |I| \mu_I^2 \right) \\
&= \frac{1}{\gamma} (y^t (B_Q - B_P) y)
\end{aligned}$$

schreiben. Da $\mathbf{P} \neq \mathbf{Q}$ ist, gilt $B_Q - B_P \neq 0$. Zerlegt man jetzt die Menge $A_\gamma^{\mathbf{P},\mathbf{Q}}$ in zwei Mengen \tilde{U} und \tilde{V} , $\tilde{U} \neq \tilde{V}$ mit $\tilde{U} := \{y \in A_\gamma^{\mathbf{P},\mathbf{Q}} : \nabla f_\gamma^{\mathbf{P},\mathbf{Q}}(y) = 0\}$, $\tilde{V} := A_\gamma^{\mathbf{P},\mathbf{Q}} \setminus \tilde{U}$ (∇ ist der Gradient).

Dann gelten folgende Inklusionen:

$$\tilde{U} \subset \{y \in A_\gamma^{\mathbf{P},\mathbf{Q}} : (B_Q - B_P)y = 0\} =: U,$$

$$\tilde{V} \subset \{y \in A_\gamma^{\mathbf{P},\mathbf{Q}} : (B_Q - B_P)y \neq 0\} =: V.$$

Da $B_Q - B_P \neq 0$, ist die Dimension des linearen Unterraums U echt kleiner als die Dimension von \mathbb{X} . Ist λ das Lebesgue-Maß auf \mathbb{R}^N , so ergibt sich: $\lambda(U) = 0 = \lambda(\tilde{U})$. Für $y \in V$ hat der Gradient folgende Form:

$$\nabla f_\gamma^{\mathbf{P},\mathbf{Q}}(y) = 2(B_Q - B_P)y \neq 0.$$

Für jedes $y \in V$ gibt es eine offene Umgebung $W(y) \subset \mathbb{X}$, so dass $W(y) \cap V$ eine C^∞ -Untermannigfaltigkeit von $W(y)$ ist (M. HIRSCH (1976)). Es gilt $\dim W(y) \cap V < \dim \mathbb{X}$ und damit gilt

$$\lambda(W(y) \cap V) = 0.$$

Nach B. v. QUERENBURG (1976) besitzt die Menge $V \subset \mathbb{R}^S$ ausgestattet mit der relativen Topologie die Lindelöf-Eigenschaft, welche besagt, dass es eine abzählbare Teilmenge $\{u_i : i \in \mathbb{N}\} \subset V$ gibt, so dass $V \subset$

$\bigcup_{i \in \mathbb{N}} W(u_i)$. Da $\tilde{V} \subset V$ ist, gilt $\lambda(\tilde{V}) \leq \lambda(V) \leq \sum_{i \in \mathbb{N}} \lambda(W(u_i) \cap V) = 0$.

Damit gilt

$$\lambda(A_\gamma^{\mathbf{P}, \mathbf{Q}}) \leq \lambda(\tilde{U}) + \lambda(\tilde{V}) = 0,$$

woraus die Behauptung folgt. ■

Beweis von Satz 3.14 Für den Beweis wird jetzt gezeigt, dass die Menge

$N_\gamma = \{y \in \mathbb{X} : x \mapsto H_\gamma(x|y) \text{ hat mindestens zwei verschiedene Minimierer}\}$

eine Lebesgue-Nullmenge ist.

Nach Lemma 3.16 sind alle $A_\gamma^{\mathbf{P}, \mathbf{Q}}$ Lebesgue-Nullmengen und ihre endliche Vereinigung

$$A_\gamma = \bigcup_{\mathbf{P} \neq \mathbf{Q}} A_\gamma^{\mathbf{P}, \mathbf{Q}}$$

ist auch eine Nullmenge. Zu zeigen bleibt noch $N_\gamma \subset A_\gamma$. Dafür wird ein $y \in N_\gamma$ gewählt, und seien \hat{x} und \hat{z} zwei dazugehörige Minimierer mit $\hat{x} \neq \hat{z}$ und \mathbf{P} bzw. \mathbf{Q} die davon jeweils induzierten Segmentationen. Dann folgt aus Gleichung 3.13 mit $f_\gamma^{\mathbf{P}, \mathbf{Q}}(y) \in \{-|S|, \dots, |S|\}$

$$N_\gamma \subset \bigcup_{\mathbf{P} \neq \mathbf{Q}} A_\gamma^{\mathbf{P}, \mathbf{Q}}$$

und damit ist N_γ Lebesgue-Nullmenge. ■

Sei ν ein Borel-Maß auf dem Zustandsraum \mathbb{X} mit Lebesgue-Dichte. Dann ist jede Lebesgue-Nullmenge auch ν -Nullmenge. Das führt zu folgendem Satz:

Satz 3.17 *Für jedes Borel-Maß auf \mathbb{X} mit Lebesgue-Dichte und für jedes $\gamma > 0$ ist der Minimierer \hat{x} des Potts-Funktional $H_\gamma(x|y)$ eindeutig für fast alle $y \in \mathbb{X}$.*

3.5 Algorithmus zur Minimierung

Eine Minimierung von $H_\gamma(x|y)$ für ein festes $\gamma > 0$ in x ist gleichbedeutend mit der Suche nach einer optimalen Segmentation für $h_\gamma(\mathbf{P}|y)$ auf der Indexmenge $S = \{1, \dots, N\}$. Um diese optimale Segmentation zu finden, lässt sich ein Algorithmus angeben, der induktiv über die Länge der Datenpunkte läuft. Für einen einzelnen Datenpunkt ist die optimale Segmentation trivial. Kennt man die optimalen Segmentationen mit deren Bewertungen für das Teilsignal $\{1, \dots, k\}$ für alle $1 \leq k \leq N - 1$, so ergibt sich nach Satz 3.11 die optimale Segmentation für $k = N$ aus der minimalen Summe zum einen des Datenterms zum Intervall Q_l^n und zum anderen aus der Bewertung der Segmentation auf $\{1, \dots, l\}$.

Algorithmus 1 (Minimierung des Potts-Funktional, eindimensional)

Allokation

\hat{l} : ARRAY N OF INT; h : ARRAY N OF REAL;

Minimierung

$h_\gamma(1) := 0, l(1) := 0, k := 2$

WHILE $k \leq N$ DO

$\hat{l}[k] := \operatorname{argmin}_{l \in \{1, \dots, k-1\}} (h[l] + \gamma + \sigma_{l+1}^k)$

$h[k] := (h[\hat{l}] + \gamma + \sigma_{\hat{l}+1}^k)$

$k := k + 1$

END;

Rekonstruktion

$k := N$

WHILE $k > 0$ DO

$\hat{P}^k = P_{\hat{l}[k]}^k$

$k := \hat{l}[k]$

END;

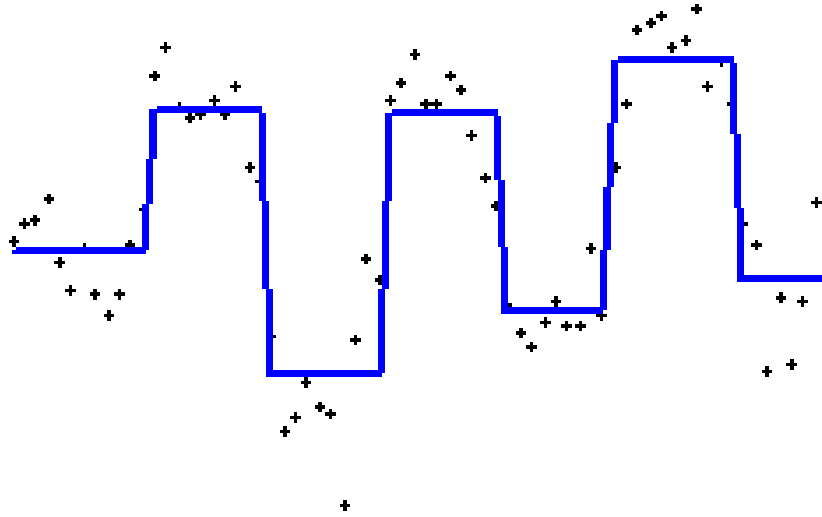


Abb. 2: Beispiel für eine Rekonstruktion durch den Algorithmus mit Daten aus einem fMRT-Experiment

3.6 Effiziente Momentenberechnung über Intervallen

Bei der Durchführung des Algorithmus wird in jedem Schritt der Mittelwert und die quadratische Abweichung im jeweiligen Intervall benötigt. Um diese Werte möglichst schnell zur Verfügung zu stellen, kann man für kleine Datenlängen die Werte für jedes Intervall vorher tabellieren, benötigt dafür also $O(N^2)$ Speicherplatz.

Für größere Datenlängen und zur effizienten Tabellierung wird nun ein Summierungstrick angegeben, mit dem Mittelwert und Abweichung für jedes Intervall in $O(1)$ berechnet werden kann. Dieser hat einen Speicherbedarf von $O(N)$.

Bemerkung 3.1 Der Ausdruck $O(g(n)) = f(n)$ mit dem Landau-Symbol O bedeutet hier und im weiteren Text, dass $|f(n)| \leq c|g(n)|$. Dabei ist die Konstante $c > 0$ und die Funktion f bestimmt die Anzahl der Basisoperationen, die zur Durchführung des Algorithmus benötigt werden, abhängig von der Länge n der eingegebenen Daten.

Zuerst müssen die Summen über alle Werte y_i und $(y_i)^2$ in den Intervallen $\{1, \dots, k\}$ ($1 \leq k \leq N$) gespeichert werden:

$$m_k := \sum_{i=1}^k y_i$$

$$s_k := \sum_{i=1}^k (y_i)^2,$$

zusätzlich $m_0 := 0$ und $s_0 := 0$.

Der Mittelwert μ_l^k über das Intervall $\{l, \dots, k\}$ berechnet sich zu

$$\mu_l^k = \frac{(m_k - m_{l-1})}{k - l + 1}.$$

Die quadratische Abweichung σ_l^k über das Intervall $\{l, \dots, k\}$ ist

$$\begin{aligned} \sigma_l^k &= \sum_{i=l}^k (y_i - \mu_l^k)^2 \\ &= \sum_{i=l}^k y_i^2 - 2 \sum_{i=l}^k y_i \mu_l^k + \sum_{i=l}^k (\mu_l^k)^2 \\ &= \sum_{i=l}^k y_i^2 - \frac{1}{k - l + 1} \left(\sum_{s=l}^k y_s \right)^2 \\ &= (s_k - s_{l-1}) - \frac{1}{k - l + 1} (m_k - m_{l-1})^2. \end{aligned}$$

Benutzt man diese Form der schnellen Berechnung, so gilt der folgende Satz:

Satz 3.18 *Der Algorithmus 1 hat die Komplexität $O(N^2)$ und benötigt Speicher der Größenordnung $O(N)$.*

Beweis Der Algorithmus braucht N Minimierungsschritte, in jedem Schritt k ($1 \leq k \leq N$) benötigt er $k - 1$ Schritte zur Minimierung des Funktionals. Jeder dieser Schritte kann mit obiger schneller Berechnung in $O(1)$ durchgeführt werden. Damit ergibt sich die Komplexität der Minimierung zu $\sum_{k=1}^N (k - 1)O(1) = O(N^2)$. Der iterative Rekonstruktionsschritt des Algorithmus benötigt maximal N Schritte, somit ist die Gesamtkomplexität auch $O(N^2)$. ■

4 Das Zweidimensionale Potts-Funktional und seine Minimierung

Für die Minimierung des zweidimensionalen Potts-Modell werden zwei Varianten hergeleitet und angegeben. Zunächst werden Definitionen und Notationen für das zweidimensionale Modell eingeführt.

4.1 Definition des Potts-Funktional

Für das zweidimensionale Potts-Modell ist die **Indexmenge** S gegeben durch

$$S = \{1, \dots, N_1\} \times \{1, \dots, N_2\}$$

für $N_1, N_2 \in \mathbb{Z}$.

Betrachtet werden nun **Bilder** $y = (y_s)$ mit $s \in S$. Die Elemente aus dem zugehörigen Zustandsraum $\mathbb{X} = \{(x_s)_{s \in S} : x_s \in E\}$, $E \subseteq \mathbb{R}$ werden als **Pixel** bezeichnet.

Für $|E| < \infty$ bezeichnet $|E|$ den Graustufenvorrat des Bildes.

Gegeben sind wieder Daten $y = (y_s)_{s \in S}$ mit Beobachtungen $y_s \in \mathbb{R}$ und ein Signal $x = (x_s)_{s \in S}$.

Das **Potts-Funktional** ist gegeben durch

$$H_\gamma : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}, \quad H_\gamma(x|y) = D(x|y) + G_\gamma(x).$$

Der **Datenterm** $D(x|y)$ ist hier wieder gegeben durch den quadratischen Abstand der Daten y zu der Rekonstruktion x :

$$D : \mathbb{X} \times \mathbb{X} \rightarrow \mathbb{R}, \quad D(x|y) = \sum_{s \in S} (x_s - y_s)^2.$$

Für den Regularisierungsterm wird zunächst die Nachbarschaftsbeziehung für Pixel im zweidimensionalen Fall angegeben:

Wie in Definition 2.1 sind zwei Pixel x_s und x_t im zweidimensionalen benachbart (in Zeichen $s \sim t$), wenn $\|s - t\|_2 = 1$ gilt, d.h. wenn entweder $(s_1 = t_1, s_2 \sim t_2)$ oder $(s_1 \sim t_1, s_2 = t_2)$ gilt.

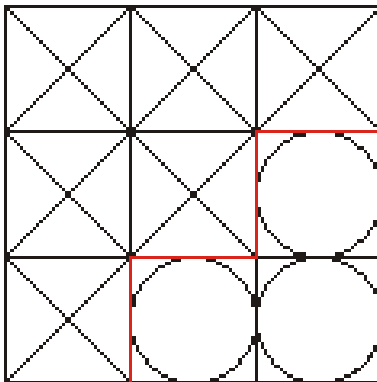


Abb. 3: Nachbarschaft zwischen unterschiedlichen Pixeln \times und \circ

Der Regularisierungsterm des eindimensionalen Potts-Funktional bewertet die Anzahl benachbarter ungleicher Pixel. Im zweidimensionalen Fall sind benachbarte Pixel durch eine Kante der Länge 1 getrennt. Für die Anzahl der Sprünge zwischen Nachbarn in der Indexmenge S wird die Gesamtkantenlänge zwischen benachbarten ungleichen Pixeln durch

$$K(x) := \sum_{\substack{s \sim t \\ s, t \in S}} \delta_{x_s, x_t}$$

gemessen.

Damit lautet der **Regularisierungsterm**

$$G_\gamma(x) = \gamma K(x).$$

Die Minimierung des Potts-Funktional $H_\gamma(x|y)$ zu gegebenen Daten y und reellem $\gamma > 0$ bedeutet im zweidimensionalen Fall wieder die Suche nach einer Approximation \hat{x} , welche einerseits die Abweichung der Daten zur Repräsentation \hat{x} klein hält und andererseits die Gesamtkantenlänge zwischen benachbarten ungleichen Pixeln minimiert.

Diese Aussage soll auf Segmentationen übertragen werden, daher nun folgende Definition allgemeiner Partitionen über der zweidimensionalen Indexmenge S . Für das Hierarchische Modell und das Wedgelet Modell wird später nur eine Teilklasse dieser Partitionen zugelassen werden.

Definition 4.1 Eine Segmentation $\mathbf{P} = (P, \mu)$ der Indexmenge S ist eine Partition P , d.h. eine Menge $\{P_1, \dots, P_n\}$ von Teilmengen (Segmenten) von S , versehen mit reellen Werten μ_1, \dots, μ_n , so dass gilt

- $S = \bigcup_{i=1}^n P_i$,
- $P_i \cap P_j = \emptyset$ für alle $i \neq j$,
- P_i ist bezüglich der Nachbarschaft " \sim " von Pixeln zusammenhängend.

Die Segmentation \mathbf{P} heißt von $x \in \mathbb{X}$ induziert, wenn gilt:

- $\mu_i = x_s$ für alle $s \in P_i$,
- $\mu_i \neq \mu_j$, falls $P_i \sim P_j$.

P_i und P_j heißen benachbart, in Zeichen $P_i \sim P_j$, genau dann, wenn $s \in P_i$ und $t \in P_j$ existieren mit $s \sim t$.

Lemma 4.2 Sei $x \in \mathbb{X}$.

Es existiert genau eine von x induzierte Segmentation \mathbf{P} .

Beweis Der Beweis ist analog zum Beweis von Lemma 3.3: Da die Menge $\{\{s\}, s \in S\}$ zusammen mit den Pixelwerten eine Partition von S darstellt, kann die Existenz der Segmentation \mathbf{P} konstruktiv gesehen werden, dabei werden benachbarte Teilmengen P_i und P_j mit gleichen Werten $\mu_i = \mu_j$ so lange sukzessive zusammengefasst, bis benachbarte Segmente keine gleichen Pixel enthalten.

Seien nun \mathbf{P} und \mathbf{Q} zwei verschiedene durch x induzierte Segmentationen. Dann existieren zwei Segmente $I \in P$ und $J \in Q$ mit $\emptyset \neq I \cap J \neq I$ und $(I \cap J) \sim (J \setminus I)$, ein klarer Widerspruch zur angegebenen Konstruktion. ■

Mit der Definition 4.1 und Lemma 4.2 bedeutet die Minimierung des Potts-Funktionalen somit die Einstellung des Gleichgewichts zwischen dem Daten-term einerseits und der Kantenlänge der Segmentation andererseits, nämlich der Länge über alle Ränder von Teilmengen der Segmentation.

Wie bereits im ersten Kapitel bemerkt, gestaltet sich die Berechnung des Minimierers schon im zweidimensionalen Fall schwierig, es gibt nämlich kein Verfahren, mit dem dieser in akzeptabler Zeit exakt berechnet werden kann. Lediglich approximative Verfahren stehen zur Verfügung.

Es werden daher im weiteren Text die Menge der “erlaubten” Segmentationen auf zwei verschiedene Weisen *ingeschränkt*. Dabei wird zum einen nicht notwendigerweise die Länge der Kanten über die Partitionen, sondern die Anzahl der Segmente in einer Partition betrachtet. Zum anderen wird die Forderung nach verschiedenen Werten auf benachbarten Segmenten aufgeweicht.

Die erste vorgestellte Variante zur Partitionierung des Bildes (das “Hierarchische Modell”) beruht auf der Übertragung der eindimensionalen Geometrie auf die Koordinatenachsen im zweidimensionalen Modell. Die zweite Variante (das “Wedgelet Modell”) erklärt Segmentationen durch Partitionen in Rechtecke (“dyadische Quadrate”) mit nachfolgender Unterteilung durch Einziehen zusätzlicher Kanten (“Wedges”).

4.2 Das Hierarchische Modell

4.2.1 Partitionen

Bei dem Hierarchischen Modell werden die Segmentationen als eigentlicher Suchraum für die Minimierung des Potts-Funktional $H_\gamma(x|y)$ folgendermaßen erklärt (siehe auch K. WICKER (2002)):

Notation 4.3 Sei $l, k \in \mathbb{Z}^2$. Es heißt $l \leq k$ genau dann, wenn $l_1 \leq k_1$ und $l_2 \leq k_2$ ist.

Die **Intervalle** in \mathbb{Z}^2 werden mit

$$I_l^k := \{j \in \mathbb{Z}^2 : l \leq j \leq k\}$$

für $l, k \in \mathbb{Z}^2$ bezeichnet. I_1^1 bezeichnet das einelementige Intervall $\{(1, 1)\} = I_{(1,1)}^{(1,1)}$.

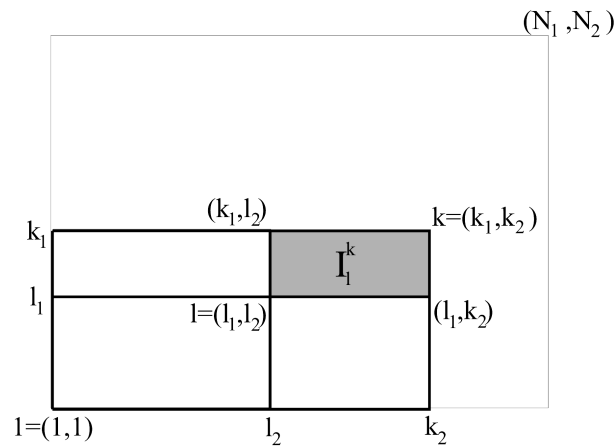


Abb. 4: Verallgemeinertes Intervall I_l^k

Eine **Segmentation** \mathbf{P} ist nun eine mit konstanten reellen Werten ausgestattete Partition in Intervalle, die folgendermaßen erreicht wird:
 In der Indexmenge $S = \{1, \dots, N_1\} \times \{1, \dots, N_2\}$ wird zunächst in vertikaler Richtung eine Segmentation festgelegt, und anschließend wird für jedes Intervall der vertikalen Segmentation eine Unterteilung in Intervalle in horizontaler Richtung bestimmt.

Eine **Partition** P wird folgendermaßen festgelegt:

$$P = \bigcup_{i=1}^n P_i, \quad P_i = \{P_{i1}, \dots, P_{im_i}\}, \quad m_i \in \mathbb{N}, 1 \leq i \leq n$$

mit

$$P_{ij} = I_i \times I_{ij}, \quad 1 \leq j \leq m_i, 1 \leq i \leq n.$$

Dabei ist $I := \{I_1, \dots, I_n\}$ eine Partition der vertikalen Richtung $\{1, \dots, N_1\}$ und $\{I_{i1}, \dots, I_{im_i}\}$ für jedes i mit $1 \leq i \leq n$ eine Partition der horizontalen Richtung $\{1, \dots, N_2\}$.

Jedem Segment P_{ij} der Partition P wird nun ein konstanter reeller Wert μ_{ij} zugeordnet ($1 \leq j \leq m_i, 1 \leq i \leq n$).

Eine **Segmentation** lässt sich also schreiben als

$$\mathbf{P} = \{(P_{ij}, \mu_{ij})_{1 \leq j \leq m_i, 1 \leq i \leq n}\}.$$

Für fixiertes i mit $1 \leq k \leq n$ wird im Folgenden mit \mathbf{P}_k die horizontale Teilpartition

$$\mathbf{P}_k := \{(P_{kj}, \mu_{kj})_{1 \leq j \leq m_k}\}$$

bezeichnet.

Ist $J = P_{ij}$, so wird im Folgenden statt μ_{ij} auch μ_J geschrieben.

$|P|$ sei als die Anzahl der Intervalle der Partition P definiert und mit \mathfrak{P} wird die Menge aller möglichen Partitionen über S bezeichnet.

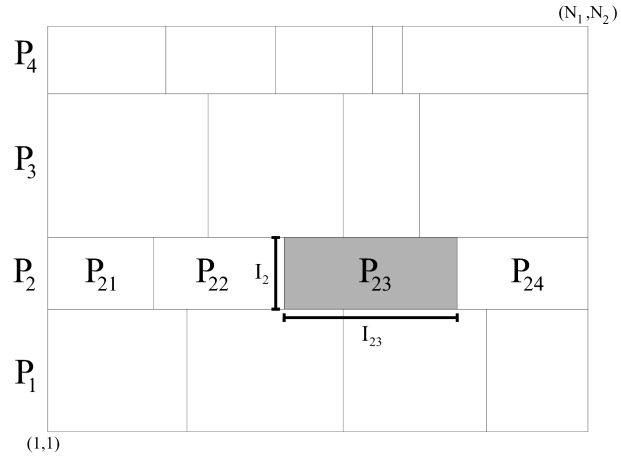


Abb. 5: 2-d Partition P des Hierarchischen Modell

Das betrachtete Funktional für eine feste Segmentation $\mathbf{P} = (P, \mu)$ lautet nun

$$h_\gamma(\mathbf{P}|y) := g_\gamma(P) + d(\mathbf{P}|y)$$

mit Regularisierungsterm

$$g_\gamma(P) := \gamma|P|$$

und Datenterm

$$d(\mathbf{P}|y) := \sum_{J \in P} \sum_{j \in J} (\hat{\mu}(J) - y_j)^2.$$

Wie oben ist auch hier $\hat{\mu}(Q)$ der Mittelwert der Daten im Segment Q ,

$$\hat{\mu}(J) := \frac{1}{|J|} \sum_{i \in J} y_i, \quad J \in P.$$

Lemma 4.4 *Das Funktional $h_\gamma(\mathbf{P}|y)$ lässt sich folgendermaßen schreiben:*

$$h_\gamma(\mathbf{P}|y) = \sum_{i=1}^n h_\gamma^*(\mathbf{P}_i|y).$$

Beweis Der Beweis erfolgt durch:

$$\begin{aligned} h_\gamma(\mathbf{P}|y) &= \gamma|P| + \sum_{J \in P} \sum_{j \in J} (\mu(J) - y_j)^2 \\ &= \gamma \sum_{i=1}^n |P_i| + \sum_{i=1}^n \sum_{J \in P_i} \sum_{j \in J} (\mu(J) - y_j)^2 \\ &= \sum_{i=1}^n h_\gamma^*(\mathbf{P}_i|y). \end{aligned}$$

■

Das Ziel ist wieder eine optimale, also das Funktional $h_\gamma(\mathbf{P}|y)$ minimierende, Segmentation zu finden. Mit Bemerkung 4.4 ist die hierarchische Struktur des Funktionals und somit auch des Minimierungsproblems deutlich zu erkennen. Eine Minimierung von $h_\gamma(\mathbf{P}|y)$ bedeutet offensichtlich, eine optimale Segmentation in vertikaler Richtung zu finden, während für jede solche dann eine Minimierung von h_γ^* vorgenommen wird. Dies wird im folgenden Abschnitt ausgeführt.

4.2.2 Existenz des Minimierers

Satz 4.5 *Bei festgelegter Segmentation \mathbf{P} mit Partition $P = (P_1, \dots, P_n)$ wird $h_\gamma(\mathbf{P}|y)$ in μ wiederum durch*

$$\mu = (\hat{\mu}(P_i))_{P_i \in P}$$

minimiert, dabei ist

$$\hat{\mu}(P_i) := \frac{1}{|P_i|} \sum_{i \in P_i} y_i, \quad P_i \in P.$$

Beweis Der Beweis erfolgt analog zu dem Beweis von Satz 3.7. ■

Darüber hinaus hat das Minimierungsproblem aber folgende, wichtige Eigenschaft:

Satz 4.6 *Jede Teilpartition \mathbf{P}_k ($1 \leq k \leq n$) der Segmentation \mathbf{P} lässt sich analog zu Satz 3.11 so finden, dass $h_\gamma(\mathbf{P}_k|y)$ in \mathbf{P}_k minimiert wird, d.h. es gilt:*

Seien Segmentationen $\mathbf{Q} = \{\mathbf{P}_1, \dots, \mathbf{P}_{k-1}, \mathbf{Q}_k, \mathbf{P}_{k+1}, \dots, \mathbf{P}_n\}$ und $\mathbf{Q}' = \{\mathbf{P}_1, \dots, \mathbf{P}_{k-1}, \mathbf{Q}'_k, \mathbf{P}_{k+1}, \dots, \mathbf{P}_n\}$ gegeben.

Dann gilt

$$h_\gamma(\mathbf{Q}_k \cup \mathbf{Q}'_k|y) = h_\gamma(\mathbf{Q}_k|y) + h_\gamma(\mathbf{Q}'_k|y) + \gamma.$$

Insbesondere gilt

$$h_\gamma(\mathbf{Q}_k \cup \{(I, \hat{\mu}(I))\}|y) = h_\gamma(\hat{\mathbf{Q}}_k|y) + \gamma + \sigma_{m+1}^{N_2}.$$

Dabei ist hier $I = P_k \times \{m+1, \dots, N_2\}$ und $\sigma_{m+1}^{N_2} := \sum_{i \in I} (y_i - \mu(I))^2$.

Beweis Die Teilpartitionen \mathbf{Q}_k und \mathbf{Q}'_k haben die Form $\mathbf{Q}_k = \{(P_k \times Q_{kj}, \mu_{ij})_{1 \leq j \leq m_k}\}$ und $\mathbf{Q}'_k = \{(P_k \times Q_{kj}, \mu_{ij})_{1 \leq j \leq m'_k}\}$ ($1 \leq m_k, m'_k \leq N_2$) und es liegt somit die Struktur von Satz 3.11 vor:

Es gilt nämlich für $1 \leq r \leq m_k$

$$h_\gamma(\mathbf{Q}_k) = h_\gamma(\{(P_k \times Q_{kj}, \mu_{ij})_{1 \leq j \leq r}\}) + \gamma + h_\gamma(\{(P_k \times Q_{kj}, \mu_{ij})_{r+1 \leq j \leq m_k}\}).$$

Damit lässt sich auch der Beweis von Satz 3.11 übernehmen. ■

Der Satz 4.6 zeigt, dass das Teilproblem, bei fixierter vertikaler Segmentation eine optimale Segmentation zu finden, analog zum eindimensionalen Fall gelöst werden kann. Es liegt also auch schon ein Algorithmus für dieses Teilproblem vor. Der nächste Satz zeigt unter Verwendung dieses Resultates, dass damit rekursiv auch die beste Gesamtpartition gefunden werden kann.

Lemma 4.7 Sei \mathbf{P} eine Segmentation von $\{1, \dots, l\} \times \{1, \dots, N_2\}$ und \mathbf{P}' eine Segmentation von $\{l+1, \dots, N_1\} \times \{1, \dots, N_2\}$ ($1 \leq l \leq N_2$).
Dann gilt

$$h_\gamma(\mathbf{P} \cup \mathbf{P}'|y) = h_\gamma(\mathbf{P}|y) + h_\gamma(\mathbf{P}'|y)$$

Beweis Siehe Bemerkung 4.4. ■

Lemma 4.7 zeigt, dass die Struktur aus Satz 4.6 auch auf die vertikale Richtung übertragen werden kann und somit auch hier das zu Satz 3.11 analoge Resultat angegeben werden kann:

Zu fixierter Vertikalpartition bezeichne nun $\hat{\mathbf{P}}_i$, für $1 \leq i \leq n$ die jeweils optimale horizontale Segmentation, darüber hinaus sei noch $\hat{\mathbf{Q}}_l^k$ der Minimierer von $h_\gamma(\mathbf{Q}_l^k)$, die Partition Q_l^k habe folgende Form:

$$Q_l^k = ((J \times J_1), \dots, (J \times J_m)),$$

wobei J das vertikale Intervall $\{l, \dots, k\}$ sei, und (J_1, \dots, J_m) die dazugehörige Partition von $\{1, \dots, N_2\}$ (in horizontaler Richtung).

Satz 4.8 Sei $k, l \in \{1, \dots, N_1\}$ mit $l < k$ und sei \mathbf{P}^l [$\hat{\mathbf{P}}^l$] eine [optimale] Segmentation über $\{1, \dots, N_1\} \times \{1, \dots, l\}$. Sei weiterhin

$$\mathbf{P}^k(l) := (\hat{\mathbf{P}}^l \cup \hat{\mathbf{Q}}_l^k).$$

Sei \hat{l} ein Minimierer von $h_\gamma(\mathbf{P}^k(l)|y)$ in l , also

$$\hat{l} \in \{l \in \{0, \dots, k-1\} : h_\gamma(\mathbf{P}^k(l)|y) \text{ ist minimal} \},$$

dann ist $\mathbf{P}^k(\hat{l})$ ein Minimierer der Funktion $h_\gamma(\mathbf{P}^k|y)$, d.h. es gilt

$$h_\gamma(\mathbf{P}^k(\hat{l})|y) = h_\gamma(\hat{\mathbf{P}}^k|y), \quad k > 1.$$

Beweis Mit Lemma 4.7 ist der Beweis analog zum Beweis von Satz 3.11. ■

4.2.3 Eindeutigkeit des Minimierers

Es kann für ein zweidimensionales Modell mit den selben Argumenten wie für den eindimensionalen Fall die Eindeutigkeit des Minimierers $\hat{\mathbf{P}}$ gezeigt werden. Es gilt also auch hier folgender Satz:

Satz 4.9 *Für jedes $\gamma > 0$ gibt es eine Lebesgue-Nullmenge $N_\gamma \in \mathbb{X}$, so dass für alle $y \notin N_\gamma$ die Funktion $H_\gamma(\mathbf{P}|y)$ einen eindeutigen Minimierer $\hat{\mathbf{P}}$ besitzt.*

Für die Übertragung des Beweises im eindimensionalen Fall (s. Satz 3.14) wird folgendes Lemma benötigt:

Lemma 4.10 *Gegeben sei eine allgemeine Partition $P = \{P_i, i = 1 \dots n\}$ der endlichen Indexmenge $S = \{1, \dots, N_1\} \times \{1, \dots, N_2\}$. Sei $\mu_k = \mu(P_k) = \frac{1}{|P_k|} \sum_{i \in P_k} y_i$ und sei \tilde{y} der Vektor, in dem die Daten über S zeilenweise aufgereiht sind, d.h.*

$$\tilde{y} := (y_i)_{1 \leq i \leq |S|}, \tilde{y}_{l+kN_1} := y_{lk}, \quad 1 \leq l \leq N_1, 1 \leq k \leq N_2.$$

Sei weiterhin $P' := \{P'_i, i = 1 \dots n\}$ die Übertragung der Partition P auf eine Partition über $\{1, \dots, |S|\}$, d.h.

$$P'_k := \{i + jl, (i, j) \in P_k\}.$$

Dann gilt

$$\sum_{i=1}^n |P_i| \mu_i^2 = \tilde{y}^t B y,$$

wobei B die $|S| \times |S|$ Matrix mit Einträgen

$$B_{i,j} := \begin{cases} \frac{1}{|P_k|} & \text{falls } i \in P'_k \wedge j \in P'_k \quad \text{für } 1 \leq k \leq n \\ 0 & \text{sonst} \end{cases}.$$

Beweis Der Beweis erfolgt mit:

$$\begin{aligned}
\tilde{y}^t B y &= \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \tilde{y}_i B_{ij} \tilde{y}_j \\
&= \sum_{i=1}^{|S|} \sum_{j=1}^{|S|} \sum_{k=1}^n \mathbb{1}_{\{i \in P_k\}} \mathbb{1}_{\{j \in P_k\}} \frac{1}{|P_k|} \tilde{y}_i \tilde{y}_j \\
&= \sum_{k=1}^n \frac{1}{|P_k|} \sum_{i \in P_k} \sum_{j \in P_k} \tilde{y}_i \tilde{y}_j \\
&= \sum_{k=1}^n |P_k| \mu_k^2.
\end{aligned}$$

■

Beweis von Satz 4.9 Der Beweis von Satz 3.14 kann übernommen werden, lediglich für das Lemma 3.15 wird eine Präzisierung für den zweidimensionalen Fall benötigt, diese ist im Lemma 4.10 angegeben. Alle Schritte nach Lemma 3.15 im eindimensionalen Fall sind dann gemäß Lemma 4.10 auf den Indexraum $\{1, \dots, |S|\}$ und damit auf den zweidimensionalen Fall übertragbar. ■

4.2.4 Algorithmus zur Minimierung

Bei der Minimierung des Funktionals h_γ des Hierarchischen Modells muss gemäß Satz 4.8 in vertikaler Richtung eine optimale Segmentation bestimmt werden, wobei während der Berechnung für jede vertikale Segmentation die optimale horizontale Segmentation gemäß Satz 4.6 berechnet werden muss. Beide Schritte geschehen völlig analog zu Algorithmus 1 zur eindimensionalen Minimierung im Abschnitt 3.5.

Der Algorithmus lässt sich also folgendermaßen angeben:

Algorithmus 2 (Minimierung des hierarchischen Funktionals)

Horizontalminimierung

GetMin(*from*, *to*: INT)

\hat{l} : ARRAY N_2 OF INT; h : ARRAY N_2 OF REAL;

$h[1] := 0$, $l[1] := 0$, $k := 2$

WHILE $k \leq N_2$ DO

$\hat{l}[k] := \arg \min_{l \in \{1, \dots, k-1\}} (h[l] + \hat{\sigma}(\{from, \dots, to\} \times \{l, \dots, k\}) + \gamma)$

$h[k] := \min_{l \in \{1, \dots, k-1\}} (h[l] + \hat{\sigma}(\{from, \dots, to\} \times \{l, \dots, k\}) + \gamma)$

$k := k + 1$

END;

Allokation

\hat{l} : ARRAY N_1 OF INT; h : ARRAY N_1 OF REAL;

Minimierung

$h_\gamma(1) := 0$, $l(1) := 0$, $k := 2$

WHILE $k \leq N$ DO

$\hat{l}[k] := \operatorname{argmin}_{l \in \{1, \dots, k-1\}} (h[l] + \operatorname{GetMin}(l, k))$

$h[k] := (h[\hat{l}] + \operatorname{GetMin}(l, k))$

$k := k + 1$

END;

Rekonstruktion

$k := N$

WHILE $k > 0$ DO

$\hat{P}^k = P^k(\hat{l})$

$k := \hat{l}[k]$

END;

4.2.5 Effiziente Momentenberechnung über Intervallen

Um einen Minimierer für das Funktional effizient bestimmen zu können, müssen die Segmentationen und damit die Mittelwerte und Summe von Quadraten schnell in einer festen Anzahl von Schritten berechnet werden können. Eine Tabellierung dieser Werte über alle benötigten Indexpaare ist im Allgemeinen indiskutabel. Es wird gezeigt, dass eine Tabellierung von $O(N_1 \cdot N_2)$ Größen jedoch genügt, um die benötigten Werte während der Minimierung in $O(1)$ berechnen zu können.

Notation 4.11 Für das Intervall I_l^k , $1 \leq l_1, k_1 \leq N_1$, $1 \leq l_2, k_2 \leq N_2$ bezeichne

$$\mu_l^k := \hat{\mu}(I_l^k) = \frac{1}{|I_l^k|} \sum_{i \in I_l^k} y_i$$

den empirischen Mittelwert der Daten y , und

$$\sigma_l^k := \hat{\sigma}(I_l^k) = \sum_{i \in I_l^k} (y_i - \mu_l^k)^2$$

die quadratische Abweichung der Daten y von dem Mittelwert μ_l^k jeweils im Intervall I_l^k .

Zur schnellen Berechnung von μ_l^k und σ_l^k müssen die Summen über alle Werte y_i und $(y_i)^2$ in den Intervallen I_{11}^{uv} ($1 \leq k \leq N_1$, $1 \leq l \leq N_2$) gespeichert werden:

$$m_{uv} := \sum_{i=1}^u \sum_{j=1}^v y_{ij}, \quad s_{uv} := \sum_{i=1}^u \sum_{j=1}^v (y_{ij})^2,$$

zusätzlich $m_{i0} := 0$, $m_{0j} := 0$ und $s_{i0} := 0$ und $s_{0j} := 0$ für alle $0 \leq u \leq N_1$ und $0 \leq v \leq N_2$.

Lemma 4.12 Mit den tabellierten Werten m und s errechnen sich Summen m_l^k von y über ein Rechteck mit

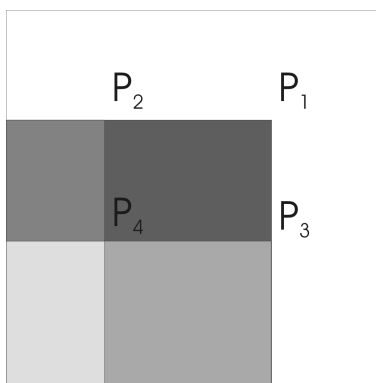
$$m_l^k = m_{k_1 k_2} - m_{k_1 l_2} - m_{l_1 k_2} + m_{l_1 l_2}.$$

Beweis Der Beweis erfolgt mit:

$$\begin{aligned}
 m_l^k &= \sum_{t \in I_l^k} y_t = \sum_{\substack{l_1 < i \leq k_1 \\ l_2 < j \leq k_2}} y_{ij} \\
 &= \sum_{\substack{0 < i \leq k_1 \\ 0 < j \leq k_2}} y_{ij} - \sum_{\substack{0 < i \leq k_1 \\ 0 < j \leq l_2}} y_{ij} - \sum_{\substack{0 < i \leq l_1 \\ 0 < j \leq k_2}} y_{ij} + \sum_{\substack{0 < i \leq l_1 \\ 0 < j \leq l_2}} y_{ij} \\
 &= m_{k_1 k_2} - m_{k_1 l_2} - m_{l_1 k_2} + m_{l_1 l_2}.
 \end{aligned}$$

■

Analog berechnet man s_l^k ($(1, 1) \leq l \leq k \leq (N_1, N_2)$). Dies wird auch geometrisch durch folgende Abbildung deutlich, wobei $P_1 = (k_1, k_2)$, $P_2 = (l_1, k_2)$, $P_3 = (k_1, l_2)$ und $P_4 = (l_1, l_2)$ sind:



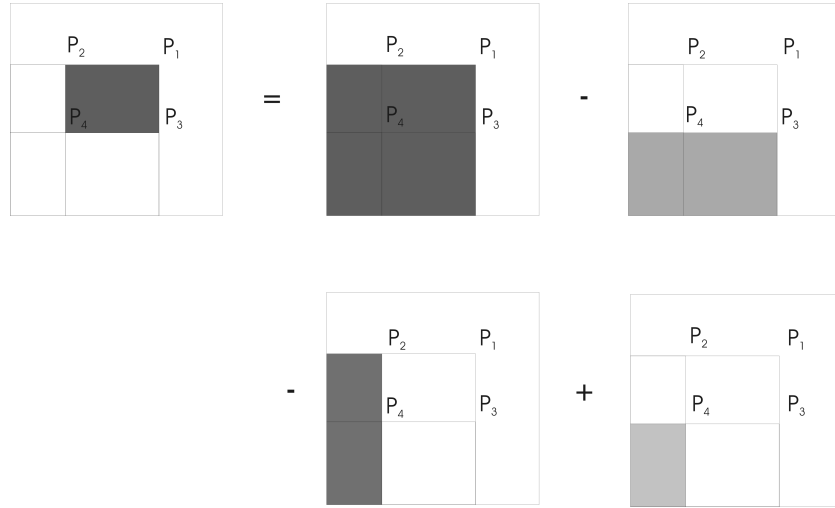


Abb. 6: Berechnung der Summe in einem Rechteck über die Matrix der kumulativen Summen

Somit berechnet sich der Mittelwert in einem Rechteck I_l^k durch:

$$\mu_l^k = \frac{m_{k_1 k_2} - m_{k_1 l_2} - m_{l_1 k_2} + m_{l_1 l_2}}{(k_1 - l_1 + 1)(k_2 - l_2 + 1)}.$$

Die Abweichungsquadrate ergeben sich mit

$$\begin{aligned} \sigma_l^k &= \sum_{i \in I_l^k} (y_i - \mu_l^k)^2 \\ &= \sum_{i \in I_l^k} y_i^2 - 2 \sum_{i \in I_l^k} y_i \mu_l^k + \sum_{i \in I_l^k} (\mu_l^k)^2 \\ &= \sum_{i \in I_l^k} y_i^2 - 2 \mu_l^k \sum_{i \in I_l^k} y_i + |I_l^k| (\mu_l^k)^2 \\ &= \sum_{i \in I_l^k} y_i^2 - 2 \frac{1}{|I_l^k|} \left(\sum_{i \in I_l^k} y_i \right)^2 + |I_l^k| \frac{1}{|I_l^k|^2} \left(\sum_{i \in I_l^k} y_i \right)^2 \\ &= \sum_{i \in I_l^k} y_i^2 - \frac{1}{|I_l^k|} \left(\sum_{i \in I_l^k} y_i \right)^2 \end{aligned}$$

zu

$$\sigma_l^k = (s_{k_1 k_2} - s_{k_1 l_2} - s_{l_1 k_2} + s_{l_1 l_2}) - \frac{(m_{k_1 k_2} - m_{k_1 l_2} - m_{l_1 k_2} + m_{l_1 l_2})^2}{(k_1 - l_1 + 1)(k_2 - l_2 + 1)}.$$

Die Matrizen $m = (m_{uv})_{(11) \leq u \leq v \leq (N_1 N_2)}$ und $s = (s_{uv})_{(11) \leq u \leq v \leq (N_1 N_2)}$ können in $O(N_1 N_2)$ berechnet werden. Dabei benutzt man die Eigenschaft, dass

$$m_{u_1 u_2}^{v_1(u_2+1)} = m_{u_1 u_2}^{v_1 u_2} + y_{v_1(u_2+1)}$$

und

$$m_{u_1 u_2}^{(v_1+1)u_2} = m_{u_1 u_2}^{v_1 u_2} + y_{(v_1+1)u_2}$$

gilt.

Benutzt man diese Form der schnellen Berechnung und Tabellierung, so gilt der folgende Satz:

Satz 4.13 *Der Algorithmus hat die Komplexität $O((N_1 N_2)^2)$ und benötigt Speicher der Größenordnung $O(N_1 N_2)$.*

Beweis Algorithmus 2 hat die gleiche Struktur wie der eindimensionale Algorithmus 1, allerdings wird in jedem (vertikalen) Schritt nochmals ein eindimensionaler Algorithmus (horizontaler Schritt) aufgerufen. Damit ergibt sich die Komplexität $O(N_1^2) \cdot O(N_2^2) = O((N_1 N_2)^2)$. Der Speicherbedarf besteht aus den zwei $N_1 \times N_2$ Matrizen m und s , sowie den temporären Vektoren l und h der Länge N_1 und l' und h' der Länge N_2 . ■

4.3 Das Wedgelet Modell

Wie bereits in Kapitel 4 bemerkt, handelt es sich bei dem Wedgelet-Modell um eine weitere Variante der Restriktion der “erlaubten” möglichen Zerlegungen für zweidimensionale Daten y .

Als erlaubt gelten Partitionen, die aus der Indexmenge S durch eine gewisse Unterteilung in Rechtecke mit anschließender Zerlegung durch Kanten (Wedges) entstehen. Solche Partitionen werden als Wedgelets bezeichnet. Dies wird im folgenden Abschnitt präzisiert. Danach wird auf eine effiziente Methode zur Minimierung eines zu solchen Partitionen gehörigen Funktionals eingegangen.

4.3.1 Partitionen

Zunächst wird die Zerlegung der Indexmenge $S = \{1, \dots, W\} \times \{1, \dots, H\}$ in sogenannte dyadische Rechtecke erklärt. Es wird im folgenden grundsätzlich vereinfachend angenommen, dass W und H Potenzen von 2 sind d.h. $W = 2^w$ und $H = 2^h$ für geeignete $w, h \in \mathbb{N}$.

Definition 4.14 Ein *dyadisches Rechteck* Q in \mathbb{Z}^2 ist ein verallgemeinertes Intervall der Form $Q = a \times b$ wobei $a =](k-1)2^l, k2^l] \cap \mathbb{Z}$ und $b =](k'-1)2^{l'}, k'2^{l'}] \cap \mathbb{Z}$ mit $l, l', k, k' \in \mathbb{N}$.

Eine **dyadische Partition** $P = \{Q_1, \dots, Q_n\}$ ($1 \leq n \leq |S|$) ist eine Unterteilung der Indexmenge S in dyadische Rechtecke Q_i ($1 \leq i \leq n$), für die folgendes gilt (vgl. 3.2):

- $Q_j \subseteq S$, für $1 \leq j \leq n$,
- $Q_1 \cup \dots \cup Q_n = S$,
- $Q_j \cap Q_i = \emptyset$ für alle $j \neq i$.

Die Bedingung $Q_j \subseteq S$ impliziert unmittelbar das folgende Lemma.

Lemma 4.15 *Für jedes Element $a \times b$ einer Partition der Indexmenge $S = \{1, \dots, 2^w\} \times \{1, \dots, 2^h\}$ mit*

$$a =](k-1)2^l, k2^l] \cap \mathbb{Z} \quad \text{und} \quad b =](k'-1)2^{l'}, k'2^{l'}] \cap \mathbb{Z}$$

gilt

$$l \in \{1, \dots, w\}, l' \in \{1, \dots, h\}, k \in \{1, \dots, 2^{w-l}\} \quad \text{und} \quad k' \in \{1, \dots, 2^{h-l'}\}.$$

Bemerkung 4.1 *In Anwendungen, bei denen in der Regel die Breite und Höhe eines Bildes nicht unbedingt Potenzen von 2 darstellen, werden die Unterteilungen durch sukzessive Halbierung der Indexgrenzen erreicht. Dabei sorgt für ungerade Längen zusätzlich eine Rundungskonvention für Unzweideutigkeit bei der Zerlegung. Man lässt Mengen der Form $a \times b$ zu mit*

$$a = [(k-1)\frac{2^w+1}{2^l}, k\frac{2^w+1}{2^l}] \cap \mathbb{Z}, \quad b = [(k'-1)\frac{2^h+1}{2^{l'}}, k'\frac{2^h+1}{2^{l'}}] \cap \mathbb{Z},$$

$$l \in \{1, \dots, w\}, l' \in \{1, \dots, h\}, k \in \{1, \dots, 2^{w-l}\} \quad \text{und} \quad k' \in \{1, \dots, 2^{h-l'}\}.$$

Es wird nun der Begriff der rekursiven dyadischen Partition eingeführt.

Definition 4.16 *Eine **rekursive dyadische Partition** P des dyadischen Rechteckes S erhält man nach folgender Vorschrift:*

- Die triviale Partition $P = \{S\}$ ist eine rekursive dyadische Partition.
- Ist $P = \{Q_1, \dots, Q_i, \dots, Q_m\}$ eine rekursive dyadische Partition und sind $R_{11}, R_{12}, R_{21}, R_{22}$ disjunkte dyadische Rechtecke mit $R_{11} \cup R_{12} \cup R_{21} \cup R_{22} = Q_i$ für ein $i \in \{1, \dots, m\}$ dann ist

$$P = \{Q_1, \dots, Q_{i-1}, R_{11}, R_{12}, R_{21}, R_{22}, Q_{i+1}, \dots, Q_m\}$$

ebenfalls eine rekursive dyadische Partition von S .

Um diese Partition P zu erhalten, werden offensichtlich die Rechtecke mit einem **quadsplit** unterteilt. Ein quadsplit ist der Vorgang, der ein dyadisches Rechteck Q in vier nebeneinanderliegende dyadische Rechtecke R_{11}, R_{12}, R_{21} und R_{22} unterteilt, deren Vereinigung Q ist.

Die entstandenen dyadischen Rechtecke werden folgendermaßen erreicht:

Lemma 4.17 *Gegeben sei ein dyadisches Rechteck $Q = U \times V$ wobei U und V die Intervalle $U :=]0, u] \cap \mathbb{Z}$ und $V :=]0, v] \cap \mathbb{Z}$ seien und bezeichne*

$$A_1 = \left]0, \frac{u}{2}\right] \cap \mathbb{Z}, \quad A_2 = \left]\frac{u}{2}, u\right] \cap \mathbb{Z}$$

$$B_1 = \left]0, \frac{v}{2}\right] \cap \mathbb{Z}, \quad B_2 = \left]\frac{v}{2}, v\right] \cap \mathbb{Z}.$$

Dann sind die durch einen quadsplit entstandenen vier dyadischen Rechtecke durch

$$R_{ij} = A_j \times B_i$$

für $i, j = 1, 2$ gegeben.

Beweis Die Zahl u ist eine Potenz von 2, d.h. $u = 2^k$ für ein $k \in \mathbb{N}$. Damit gilt für die Längen der Intervalle $]0, c] \cap \mathbb{Z}$, $]c, u] \cap \mathbb{Z}$ mit $]0, c] \cap \mathbb{Z} \cup]c, u] \cap \mathbb{Z} =]0, u] \cap \mathbb{Z}$ $2^{k-1} = u/2$, denn $2^{k'} + 2^{k''} = 2^k$ gilt nur für $k' = k'' = 2^{k-1}$. Somit ist $c = u/2$. Dasselbe gilt für das Intervall $]0, v] \cap \mathbb{Z}$. ■

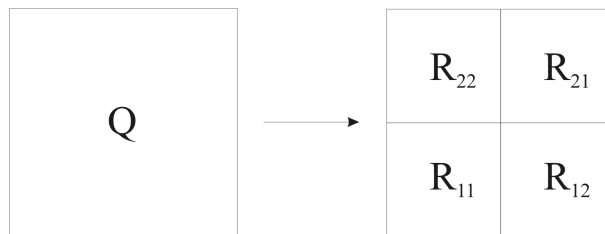


Abb. 7: Rechteck Q und die Partition in $R_{11}, R_{12}, R_{21}, R_{22}$ durch einen quadsplit

Satz 4.18 *Die rekursiven dyadischen Partitionen von S sind dyadische Partitionen von S .*

Beweis

Sei die rekursive dyadische Partition durch $\{Q_1, \dots, Q_n\}$, $n \in \mathbb{N}$ gegeben. Es muss nachgewiesen werden, dass die Eigenschaften $Q_i \subseteq S$, $Q_1 \cup \dots \cup Q_n = S$ und $Q_j \cap Q_i = \emptyset$ für $i \neq j$, $1 \leq i, j \leq n$ gelten. Das ist sofort ersichtlich aus der Definition der rekursiven dyadischen Partition. Außerdem besteht die rekursive dyadische Partition per definitionem aus dyadischen Rechtecken; somit ist die Aussage bewiesen. ■

Der Begriff des quadsplits legt eine Verknüpfung mit der Datenstruktur **quadtree** nahe.

Definition 4.19 *Ein **quadtree** ist ein endlicher Baum, dessen Knoten entweder 4 oder gar keine Kinder haben.*

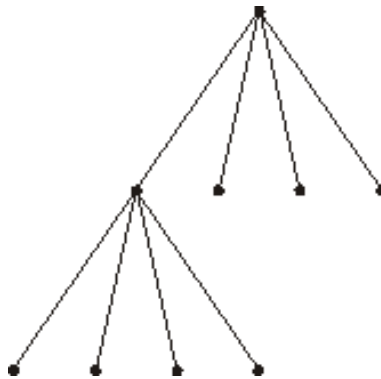


Abb. 8: Beispiel für einen quadtree

Wird auf ein dyadisches Rechteck ein quadsplit angewendet, so entstehen 4 dyadische Rechtecke. Wird wiederum in einem dieser neu entstandenen Rechtecke ein quadsplit angewendet, so sind in der dann entstandenen Partition 3 zusätzliche Rechtecke vorhanden, dies führt zu folgender Aussage:

Lemma 4.20 Für eine rekursive dyadische Partition P bezeichne $|P|$ die Anzahl der dyadischen Rechtecke in der Partition. Diese lässt sich über die Anzahl i angewendeter quadsplits angeben: $|P| = 3i + 1$.

Da der Begriff der rekursiven Partition mit dem quadtree verknüpft ist, führt der Weg von $[1, W] \cap \mathbb{Z} \times [1, W] \cap \mathbb{Z}$ zu einem kleinsten dyadischen Rechteck der Größe 1×1 über $\log_2 W$ Knoten, was der Höhe des Baumes entspricht. Daraus folgt sofort:

Folgerung 4.21 Ein dyadisches Rechteck R mit maximaler Seitenlänge 1, das vollständig in dem Rechteck $Q = [1, W] \cap \mathbb{Z} \times [1, H] \cap \mathbb{Z}$ (mit $W \leq H$) liegt, kann durch wenigstens $\frac{(4 \log_2 W - 1)}{3}$ quadsplits aus Q erzeugt werden. Dabei sind $4 \log_2 W + 1$ dyadische Rechtecke entstanden.

Durch einfaches Abzählen erhält man:

Lemma 4.22 Wird ein dyadisches Rechteck Q der Größe $2^i \times 2^j$ (mit $i \leq j$) solange mit quadsplits zerlegt, bis die dyadischen Rechtecke Seitenlänge 1 besitzen, so sind dabei $\frac{4^{i+1} - 1}{3}$ dyadische Rechtecke entstanden.

Einer **Wedgelet-Partition** $P = \{Q_1, \dots, Q_n\}$ liegt eine Unterteilung der Indexmenge S in dyadische Rechtecke zugrunde. Jedes dieser dyadischen Rechtecke Q_i , ($1 \leq i \leq n$) kann dann zusätzlich mit einer Kante unterteilt werden.

Notation und Definition 4.23

1. Eine **Kante** $k(p, \alpha)$ mit $p \in S$ und Winkel $\alpha \in] - \pi, \pi]$ ist die Gerade durch den Punkt p , so dass die Gerade den Winkel α mit der Horizontalen einschliesst, d.h.

$$k(p, \alpha) := \{r \in \mathbb{R}^2 : r = p + \lambda \begin{pmatrix} \cos \alpha \\ \sin \alpha \end{pmatrix}, \lambda \in \mathbb{R}\}.$$

Die Einschränkung des Winkelbereichs auf $] - \pi, \pi]$ ist gerechtfertigt durch die Identität $k(p, \alpha) = k(p, \alpha + \pi)$.

2. Die Unterteilung des dyadischen Rechteckes $Q \subset \mathbb{Z}^2$ in zwei Segmente, die durch das Einfügen einer Kante $k(p, \alpha)$ mit $p \in Q$ entsteht, wird mit $W_{\alpha, p}(Q)$ bezeichnet. Es gilt $W_{\alpha, p}(Q) := \{A, B\}$ mit

$$A := \{u \in Q : (u_2 - p_2) \cos \alpha \geq (u_1 - p_1) \sin \alpha\}$$

$$B := \{u \in Q : (u_2 - p_2) \cos \alpha < (u_1 - p_1) \sin \alpha\}$$

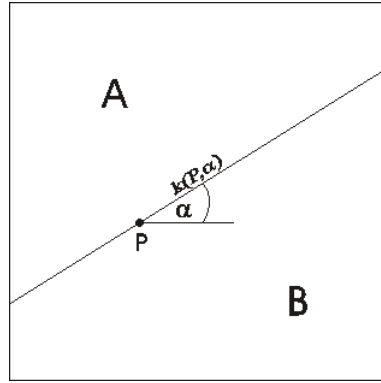


Abb. 9: Die Unterteilung $\Phi_{k(p, \alpha)}(Q)$ durch die Kante $k(p, \alpha)$

3. Gegeben sei eine Winkelmenge $\Delta := \{\square, \alpha_1, \dots, \alpha_k\}$ mit $\alpha_i \in]-\pi, \pi]$ für alle $1 \leq i \leq k$. Dabei steht \square dafür, dass nicht durch eine Kante unterteilt wird, d.h. $W_{\square, p}(Q) := \{Q\}$ für alle p .

Eine **Wedgelet-Partition** P_Δ von S zur Winkelmenge Δ ist eine Partition, für welche eine dyadische Partition P von S existiert, so dass

$$P_\Delta = \{W_{p(Q), \alpha(Q)}(Q) : Q \in P\} = \bigcup_{Q \in P} W_{p(Q), \alpha(Q)}(Q)$$

mit $p(Q) \in Q$ und $\alpha(Q) \in \Delta$ für alle $Q \in P$.

4. Mit $|P_\Delta|$ wird die Anzahl der Komponenten der Partition bezeichnet, wobei ein dyadisches Rechteck mit zusätzlicher Unterteilung durch eine Winkelgerade aus zwei Komponenten besteht, ein Rechteck ohne eine weitere Winkelunterteilung besteht aus einer Komponente:

$$|P_\Delta| := \sum_{Q \in P} |W_{p(Q), \alpha(Q)}(Q)|.$$

Jedem Segment der Wedgelet-Partition wird nun ein konstanter reeller Wert zugeordnet, das führt zu der Definition der **Wedgelet-Segmentation**.

Definition 4.24 Gegeben sei eine Wedgelet-Partition P_Δ zu einer dyadischen Partition P auf S und einer Winkelmenge Δ . Zusätzlich seien die konstanten reellen Werte $\mu := (\mu_1, \dots, \mu_n)$ gegeben, $n = |P_\Delta|$.

Eine **Wedgelet-Segmentation** \mathbf{P} zur Partition P_Δ ist ein Paar

$$\mathbf{P} := (P_\Delta, \mu),$$

wobei die Wedgelet-Partition P_Δ , in der jede Komponente mit einem μ_i ausgestattet wird.

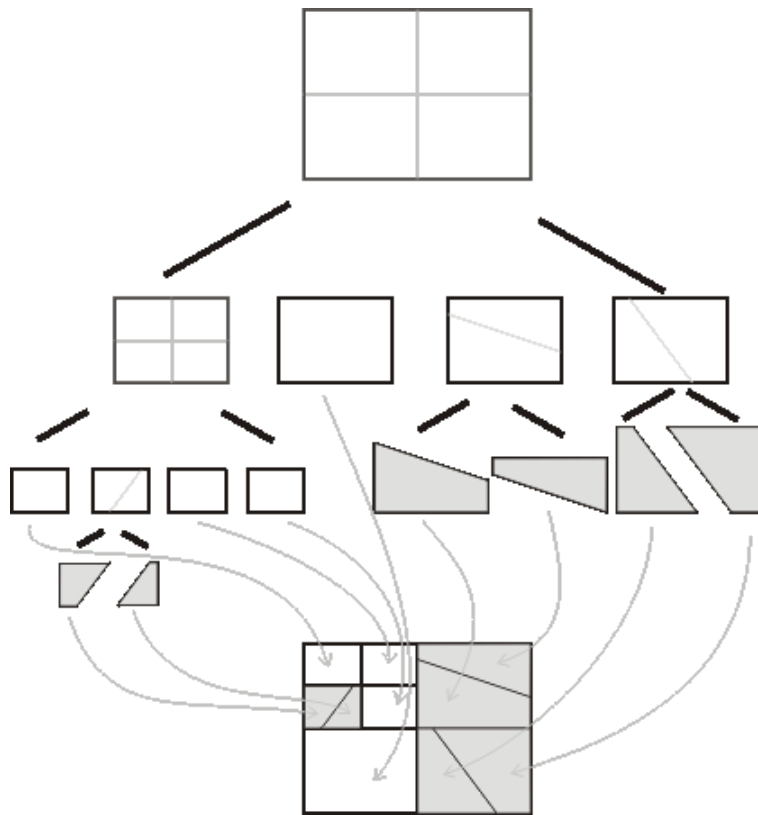


Abb. 10: Beispiel für eine Wedgelet-Segmentation. Vgl auch das zugehörige Beispiel für den quadtree (Abb. 9)

Sei P_Δ eine dyadische Partition auf S und sei Δ eine festgelegte Winkelmenge.

Das betrachtete Funktional $h_\gamma(\mathbf{P}|y)$ zu der Wedgelet-Segmentation $\mathbf{P} := (P_\Delta, \mu)$ ist gegeben durch

$$h_\gamma(\mathbf{P}|y) := g_\gamma(P) + d(\mathbf{P}|y),$$

wobei der Regularisierungsterm g_γ , gegeben durch

$$g_\gamma(P) := \gamma|P|,$$

die Anzahl der Komponenten der Wedgelet-Segmentation bewertet.

Der Datenterm

$$d(\mathbf{P}|y) := \sum_{Q \in P_\Delta} \sum_{j \in Q} (\mu(Q) - y_j)^2$$

misst den Abstand von x zu den gegebenen Daten y .

Wie oben ist auch hier $\hat{\mu}(Q)$ der Mittelwert der Daten im Segment Q ,

$$\hat{\mu}(Q) := \frac{1}{|Q|} \sum_{i \in Q} y_i, \quad Q \in P_\Delta.$$

Gesucht wird wieder eine optimale Segmentation \mathbf{P} , die das Funktional $h_\gamma(\mathbf{P}|y)$ minimiert.

4.3.2 Existenz des Minimierers

Für eine festgelegte Wedgelet-Segmentation \mathbf{P} mit der Wedgelet-Partition $P_\Delta = (Q_1, \dots, Q_n)$ ist der Minimierer von $h_\gamma(\mathbf{P}|y)$ analog zu Satz 3.7 durch

$$\hat{\mu}(P_\Delta) = (\hat{\mu}(Q_1), \dots, \hat{\mu}(Q_n))$$

gegeben.

Satz 4.25 *Es existiert ein Minimierer für das Funktional $h_\gamma(\mathbf{P}|y)$.*

Beweis Für eine feste Wedgelet-Partition ist der Minimierer bekannt. Deshalb muss der Minimierer nur noch aus der endlichen Menge aller dyadischen Partitionen und allen möglichen Winkelunterteilungen der dyadischen Rechtecke (mit Winkeln aus der endlichen Menge Δ) gesucht werden. ■

Bemerkung 4.2 *Die Aussage des Satzes 4.25 ist auch dann noch richtig, wenn man alle Winkel $\alpha \in]-\pi, \pi]$ und alle Startpunkte $p \in \mathbb{R}^2$ zulässt. Denn selbst dann ist die Anzahl möglicher Partitionen endlich.*

4.3.3 Eindeutigkeit des Minimierers

Für das Wedgelet-Modell kann der Beweis der Eindeutigkeit im Falle des hierarchischen Modelles exakt übernommen werden. Die in Lemma 3.15 angegebene Struktur, die eine lineare Darstellung des Ausdrucks $\sum_{i=1}^n |P_i| \cdot \mu_i^2$, erlaubt ist auch hier gültig und wird wiederholt angegeben:

Es gilt mit der durch

$$B_{i,j} := \begin{cases} \frac{1}{|Q_k|} & \text{falls } i \in Q'_k \wedge j \in Q'_k \quad \text{für ein } 1 \leq k \leq n \\ 0 & \text{sonst} \end{cases}$$

definierten $|S| \times |S|$ Matrix B nämlich, dass $\sum_{i=1}^n |Q_i| \cdot \mu_i^2 = \tilde{y}^t B y$ und damit kann wie im eindimensionalen Fall argumentiert werden (vgl. 4.2.3).

Damit gilt auch für das Wedgelet-Modell h_γ folgender Satz:

Satz 4.26 *Für jedes $\gamma > 0$ gibt es eine Lebesgue-Nullmenge $N_\gamma \in \mathbb{X}$, so dass für alle $y \notin N_\gamma$ die Funktion $h_\gamma(\mathbf{P}|y)$ einen eindeutigen Minimierer $\hat{\mathbf{P}}$ besitzt.*

Beweis Der Beweis verläuft mit obiger Matrix analog zu dem Beweis von Satz 4.9 und damit analog zu Satz 3.14. ■

4.3.4 Algorithmus zur Minimierung

Anders als im hierarchischen Modell kann beim Wedgelet-Modell nicht mehr rekursiv über die Dimensionen iteriert werden. Trotzdem ist eine sehr schnelle Minimierung des Funktionals im Wedgelet-Modell möglich. Entscheidend dafür ist die rekursive Struktur der dyadischen Partitionen (quadtree). Der Wert des Funktionals muss zu seiner Minimierung nämlich nur an den Knoten des Baumes verglichen werden. Neben der "lokalen" Minimierung durch Angabe von Winkel und Startpunkt genügt in jedem Knoten die Kenntnis des minimalen Wertes des Funktionals an den Nachfolger Knoten:

Sei $\mathfrak{P}(Q)$ die Menge aller Partitionen von $Q \subset S$ und sei $M(Q)$ das Minimum von $h_\gamma(\mathbf{P}_Q|y_Q)$ über Q , d.h.

$$M(Q) \leq h_\gamma(\mathbf{P}_Q|y)$$

für alle $\mathbf{P}_Q \in \mathfrak{P}(Q)$.

Für $\hat{\alpha} \in \Delta$ und $\hat{p} \in S$ gelte

$$h_\gamma(W_{\hat{p},\hat{\alpha}}|y) \leq h_\gamma(W_{p,\alpha}|y)$$

für alle $\alpha \in \Delta$ und $p \in Q$. Dann gilt mit dyadischer Unterteilung Q_{11} , Q_{12} , Q_{21} , Q_{22} von Q

$$M(Q) = \min\{M(Q_{11}) + M(Q_{12}) + M(Q_{21}) + M(Q_{22}), h_\gamma(W_{\hat{p},\hat{\alpha}}|y)\}.$$

Diese Rekursion lässt sich algorithmisch folgendermaßen wiedergeben:

Algorithmus 3 (Minimierung des Wedgelet Modells)

Input

Q : ARRAY OF ARRAY OF INT, $\Delta = (\square, \alpha_1, \dots, \alpha_k)$, $\gamma > 0$

Allokation

$minw, min_{11}, min_{12}, min_{21}, min_{22}$: REAL;

Minimierung

PROCEDURE MIN(Q, γ): REAL;

$minw := \mathbf{MinWedge}(Q, \Delta, \gamma)$;

$min_{ij} := \mathbf{MIN}(Q_{ij}, \gamma)$, $i, j = 1, 2$

IF $min_{11} + min_{12} + min_{21} + min_{22} < minw$ **THEN**

choose Subpartition; **RETURN** $min_{11} + min_{12} + min_{21} + min_{22}$

ELSE

choose Wedge; **RETURN** $minw$

END;

Die im Algorithmus lediglich zitierte Prozedur *MinWedge* sucht zu jedem Rechteck Q die optimale Unterteilung durch Einziehen von Kanten. Die genaue Form dieser Prozedur ist abhängig von der Methode, mit der die Diskretisierung der Kanten vorgenommen wird, von der Auswahl der Startpunkte der Kanten und vor allem von der Methode, die zur möglichst effizienten Berechnung der Summen in den Komponenten der Partition gewählt wird. Darauf wird im folgenden Abschnitt noch eingegangen.

Der skizzierte Algorithmus beschreibt zwar die rekursive Struktur der Minimierung, geht jedoch nicht auf Details bzgl. des Aufbaus des quadrees ein. Diese sind Standard in der Programmierung dynamischer Strukturen und werden daher hier weggelassen.



Abb. 11: Original *Camera* (links) und Resultat (rechts) mit $\gamma = 0,2$.

Die Prozedur MIN wird für jeden der $O(|S|)$ Knoten (s. Lemma 4.22) genau einmal aufgerufen. Eine Angabe der Komplexität des Problems ist erst mit der Angabe der Komplexität von *MinWedge* möglich. Für die folgende Abschätzung der Gesamtkomplexität des Algorithmus wird angenommen, dass die Indexmenge S ein Quadrat mit Seitenlänge 2^v ($v \in \mathbb{N}$) ist, d.h.

$$S := \{1, \dots, 2^v\} \times \{1, \dots, 2^v\}.$$

Es gilt folgendes Lemma:

Lemma 4.27 *Die Minimierung von $h_\gamma(\mathbf{P}|y)$ in \mathbf{P} kann mit Algorithmus 3 in*

- $O(|S|^2)$ Schritten durchgeführt werden, sofern die lokale Minimierung “MinWedge” Komplexität $O(N^2)$ hat und in
- $O(|S|\log|S|)$ Schritten durchgeführt werden, sofern die lokale Minimierung “MinWedge” Komplexität $O(N)$ hat und in
- $O(|S|)$ Schritten durchgeführt werden, sofern die lokale Minimierung “MinWedge” Komplexität $O(\sqrt{N})$ hat.

Beweis Angenommen, die Prozedur *MinWedge* habe lineare Komplexität. Daraus ergibt sich eine Anzahl Schritte für die Minimierung

$$\begin{aligned}
C(v) &= \sum_{i=0}^v 4^i \cdot \underbrace{\text{Anzahl Schritte in MIN}}_{c_1} + 4^i \cdot \underbrace{\text{Anzahl Schritte in } \textit{MinWedge}}_{(4^v/4^i)c_2} \\
&= c_1 \frac{4^{v+1} - 1}{3} + c_2 v 4^v \\
&= c_1 \frac{4|S| - 1}{3} + c'_2 |S| \log |S|.
\end{aligned}$$

Somit ergibt sich für lineare Komplexität von *MinWedge* eine Gesamtkomplexität von $O(|S| \log |S|)$.

Nun habe die Prozedur *MinWedge* quadratische Komplexität. Daraus ergibt sich eine Anzahl an Schritten für die Minimierung durch

$$\begin{aligned}
C(v) &= \sum_{i=0}^v 4^i \cdot \underbrace{\text{Anzahl Schritte in MIN}}_{c_1} + 4^i \cdot \underbrace{\text{Anzahl Schritte in } \textit{MinWedge}}_{(4^v/4^i)^2 c_2} \\
&= c_1 \frac{4^{v+1} - 1}{3} + c_2 16^v \frac{4 - (\frac{1}{4})^v}{3} \\
&= c_1 \frac{4|S| - 1}{3} + c'_2 \frac{4|S|^2 - |S|}{3}.
\end{aligned}$$

Somit ergibt sich für quadratische Komplexität von *MinWedge* eine Gesamtkomplexität von $O(|S|^2)$.

Angenommen, die Prozedur *MinWedge* habe Komplexität $O(\sqrt{N})$ (N sei die Anzahl Pixel, die *MinWedge* bearbeitet), dann gilt

$$\begin{aligned}
C(v) &= \sum_{i=0}^v 4^i \cdot \underbrace{\text{Anzahl Schritte in MIN}}_{c_1} + 4^i \cdot \underbrace{\text{Anzahl Schritte in } \textit{MinWedge}}_{\sqrt{4^v/4^i} c_2} \\
&= c_1 \frac{4^{v+1} - 1}{3} + c_2 (2^{v+1} - 1) 2^v \\
&= c_1 \frac{4|S| - 1}{3} + c_2 (2|S| - \sqrt{|S|}).
\end{aligned}$$

■

Diese Lemma zeigt, dass mit einer Komplexität $O(\sqrt{N})$ von *MinWedge* sich eine Gesamtkomplexität von $O(|S|)$ ergibt!

4.3.5 Effiziente Momentenberechnung über Intervallen

Nach Lemma 4.27 hängt die Komplexität des Algorithmus 3 stark von der Komplexität der lokalen Minimierung in den einzelnen Schritten des Algorithmus ab. Wichtig ist also, eine möglichst effiziente lokale Minimierung zu finden. Die Anzahl aller möglichen Unterteilungen von $\{1, \dots, W\}^2$ durch eine Kante in \mathbb{R}^2 ist nach J. KOPLOWITZ et al. (1990) gegeben durch $3W^4/\pi^2 + O(W^3 \log W)$. Jede dieser Kanten in die Minimierung einzubeziehen würde (nach Lemma 4.27) eine Komplexität von $O(|S|^2)$ bedeuten, sofern man die Werte für jede der Kanten in $O(1)$ berechnen könnte. Hier wird nun Effizienz in zweierlei Weise erreicht: Zum einen wird, wie schon angemerkt, die Anzahl der Winkel und Startpunkte für die Kanten eingeschränkt. Die Winkel werden vor der Berechnung fest vorgegeben. Als Startpunkte für die Kanten werden die Randpunkte auf den Rechtecken gewählt. Zum anderen werden für feste Winkel kumulative Matrizen vortabelliert, die eine Berechnung der benötigten Momente für jede Kante in $O(1)$ ermöglichen. Dies wird nun genauer erläutert.

Es werden Mittelwerte und Summe der Quadrate in Rechtecken des Bildes y benötigt. Im Abschnitt 4.2.5 wurde für die effiziente Berechnung dieser Größen bereits ein Verfahren angegeben. Zusätzlich werden diese Momente aber auch für Dreiecke und aus Dreiecken und Rechtecken zusammengesetzte Figuren benötigt. Inhalt dieses Abschnitts ist nun, den Summierungstrick aus Abschnitt 4.2.5 durch Summierung entlang der vorgegebenen Kanten auf Dreiecke zu übertragen. Im Folgenden wird zur Vereinfachung angenommen, dass für jeden Winkel $\alpha \in \Delta \setminus \{\square\}$ gilt $\alpha \in [0, \frac{\pi}{2}]$, d.h. $\cos \alpha \geq \sin \alpha \geq 0$. Diese Restriktion dient lediglich zur Vereinfachung der Notation und zur Vermeidung vieler Unterscheidungen, die Ergebnisse dieses Abschnitts lassen sich auch auf allgemeine Winkel übertragen, so dass die Anzahl der Schritte von *MinWedge* höchstens vervierfacht wird.

Bevor die Momentenberechnung entlang allgemeiner Kanten und auf Dreiecken eingeführt wird, wird für deren effiziente Berechnung noch etwas Vor-

arbeit benötigt:

Notation 4.28 Für die schnelle Berechnung der Mittelwerte und der quadratischen Abweichung werden die Summen λ der Werte von y und die Summen κ der Quadrate der Werte in Spalten eines Bildes gespeichert:

$$\lambda_{uv} := \sum_{j=1}^v y_{uj}, \quad \kappa_{uv} := \sum_{j=1}^v (y_{uj})^2$$

zusätzlich $\lambda_{i0} := 0$, $\lambda_{0j} := 0$, $\kappa_{i0} := 0$ und $\kappa_{0j} := 0$ für alle $0 \leq u \leq N_1$ und $0 \leq v \leq N_2$.

Mit diesen Summen erhält man die in Abschnitt 4.2.5 schon definierten Matrizen m und s durch

$$m_{u,v} := \sum_{i=1}^u \lambda_{iv}, \quad s_{u,v} := \sum_{i=1}^u \kappa_{iv}.$$

Diese Matrizen beinhalten die Summe über Werte und deren Quadrate links unterhalb vom Punkt (u, v) .

Die Summe entlang der Kanten werden wie folgt definiert:

$$d_{u,v}(\alpha) := \sum_{i=1}^u \lambda_{i, \lfloor v - i \tan \alpha \rfloor}, \quad q_{u,v}(\alpha) := \sum_{i=1}^u \kappa_{i, \lfloor v - i \tan \alpha \rfloor}.$$

Ein dyadisches Rechteck $Q = [a, b] \cap \mathbb{Z} \times [c, d] \cap \mathbb{Z}$ sei nun durch eine Kante $k := k(p, \alpha)$ unterteilt. Die Kante starte in Punkt $P_{Start} = (e, f)$ und verlasse das Rechteck *genau* im Punkt $P_{End} = (g, h)$, d.h. es existiere eine $\lambda \in \mathbb{R}$ mit $(e, f) + \lambda(\cos \alpha, \sin \alpha) = (g, h)$.

Die Summe in einer Dreiecksfläche $\text{sum}(\Delta)$ kann dann einfach berechnet werden: Die aufsummierten Werte links unterhalb der Kante an der Stelle P_{End} werden von dem aufsummierten Wert am Punkt auf der Linie vor dem Startpunkt P_{start} abgezogen, sei $(g', h') := (g - 1, \lfloor h - \tan \alpha + 1/2 \rfloor)$

$$\text{sum}(\Delta) = d_{e,f} - d_{g',h'} + m_{e,h} - m_{g-1,h-1}.$$

Für die Berechnung der Quadratsumme ssq muss man lediglich d durch q und m durch s ersetzen. Die Berechnung von Mittelwert und Abweichungsquadrat kann analog zu Abschnitt 4.2.5 mit

$$\mu(\Delta) = \frac{\text{sum}(\Delta)}{|\Delta|}$$

und

$$\sigma(\Delta) = ssq(\Delta) - \frac{1}{|\Delta|}(\text{sum}(\Delta))^2$$

durchgeführt werden.

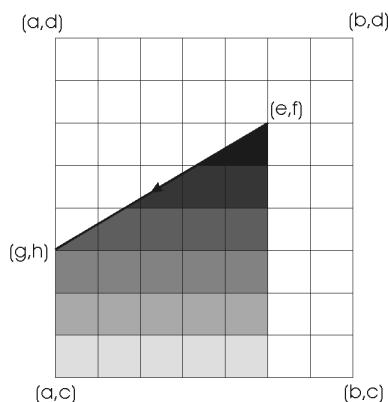


Abb. 12: Die Summen unterhalb einer Kante

Dies wird auch in folgender Abbildung geometrisch deutlich:

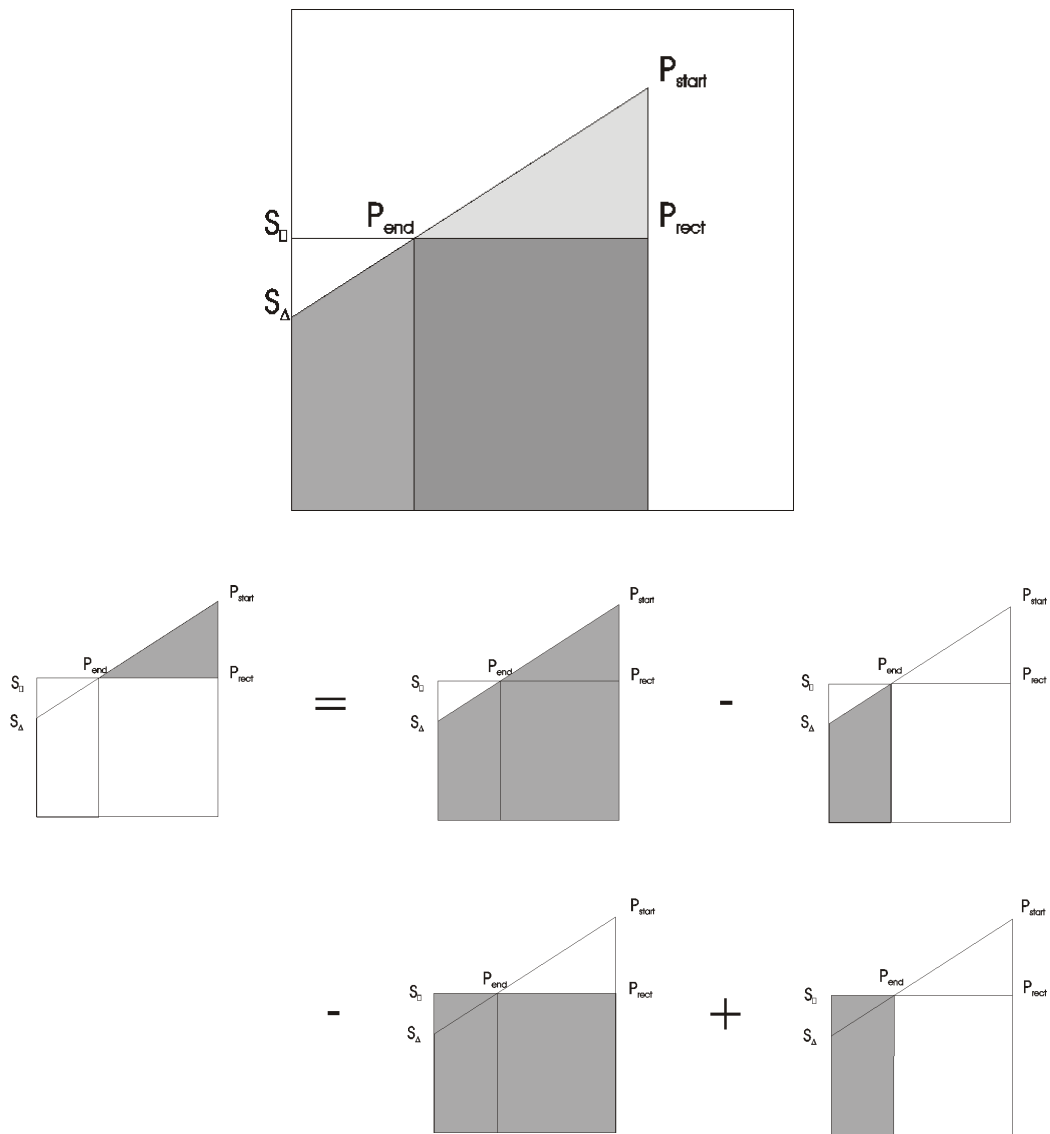


Abb. 13: Berechnung der Summe in einem Dreieck über die aufsummierten Matrizen

Somit ist nach der Tabellierung eine Berechnung der Momente in einem Dreieck in $O(1)$ möglich, sofern die Start- und Endpunkte der Kante **genau** durch den Pixelmittelpunkt gehen. Andernfalls besteht folgendes Problem: Die Kanten durch den Anfangspunkt und durch den Endpunkt mit Winkel α verlaufen nicht parallel und es sind somit Fehler bei der Differenzbildung möglich, dieses Problem wird durch folgende Abbildung verdeutlicht.

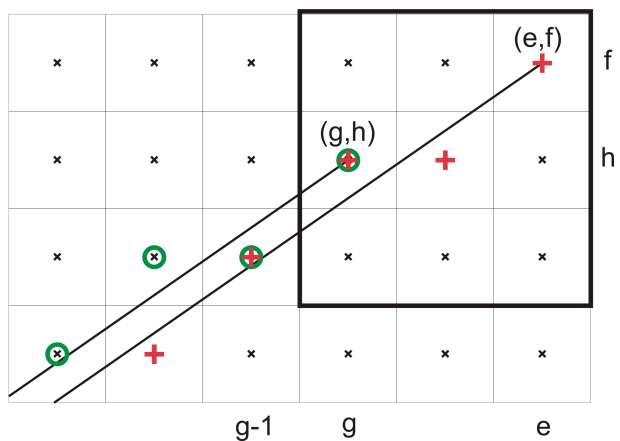


Abb. 14: Fehler bei der Differenzbildung

In den nächsten beiden Abschnitte werden für dieses Problem zwei Lösungen angeboten: Zum einen die Interpolationsmethode, bei der der Fehler durch Interpolation und Verwendung von Teilpixeln verringert wird. Zum anderen eine Methode, bei der Gesamtlinien vorgegeben werden und Partitionen so ausgewählt werden, dass die Eckpunkte garantiert auf gleichen Gesamtlinien liegen.

4.3.5.1 Interpolations-Methode

Um die Summe der Werte unterhalb einer Kante $k := k(P, \alpha)$ mit der Länge $len(k)$ zu bekommen, wird von dem Startpunkt s diese Linie pixelweise nach links unten durchwandert. In jedem Pixel x_i wird die Summe um den unteren Pixelanteil erhöht, der durch Schnitt des Pixels mit der Kante k entsteht:

$$\text{sum}_k(s) := \sum_{s \in K(s)} \text{val}(y_s) \cdot F(s).$$

Dabei ist die Menge $K(s)$ die Menge aller Pixel, die von der Kante k links von $s \in S$ geschnitten werden:

$$K(s) := \{(u, v) \in S : [u - \frac{1}{2}, u + \frac{1}{2}] \times [v - \frac{1}{2}, v + \frac{1}{2}] \cap \{s - \nu(\frac{\cos \alpha}{\sin \alpha}), \nu \geq 0\} \neq \emptyset\}.$$

$F(s)$ bezeichne die Fläche, die unterhalb von k im Pixel s liegt.

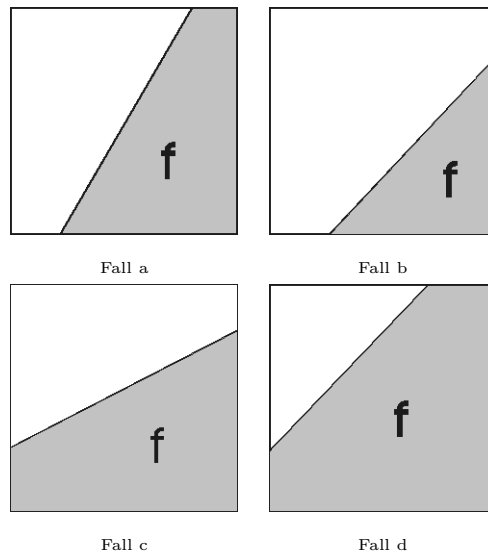


Abb. 15: Alle 4 Fälle für die Unterteilung des Pixels y_s mit einer Kante und die dazugehörige Fläche

Für $\text{val}(y_s)$ im Pixel y_s muss in jedem Pixel unterschieden werden, ob die Linie das Pixel durch seinen y-Achse verlässt (Fall a und b). In diesen Fällen

wird nur der flächenmässige Anteil des Pixels unterhalb der Kante zur Gesamtsumme dazu addiert, d.h. $val(y_s) = y_s$. Bei Fall c und d muss zu dem flächenmässige Anteil des Pixels auch noch die Summe der Spalten unterhalb dieses Pixels aus der Spaltenmatrix m zur Gesamtsumme dazu addiert werden. $val(y_s)$ steht hier für die Summe aller Pixel in der Spalte unterhalb s , d.h. $val(y_s) = \frac{m_s}{F(s)} + y_s$.

Dies muss für jedes Pixel berechnet werden, das die Kante bis zum Schnittpunkt mit dem Bildrand durchläuft. Das Durchlaufen des Bildes von einem Punkt P mit der Kante $k(P, \alpha)$ hat die Komplexität der Länge der Kante, also $O(W)$ für Fall a und b, und $O(H)$ in Fall c und d. Die Vortabellierung der kumulativen Summen benötigt also pro Kante $O(|S|^{(3/2)})$ Schritte. Eine Verwendung des Ergebnisses anderer Pixel bei der Berechnung der Summe für das Pixel s ist im Allgemeinen ausgeschlossen, da Kanten durch verschiedene Pixel im allgemeinen verschieden sind.

Zusätzlich zu diesem relativ hohen Tabellierungsaufwand bringt die Methode aber auch Ungenauigkeiten mit sich. Da auf Teilpixeln gerechnet wird, muss beim Abziehen der unteren Dreiecksfläche und für die Summe des Rechtecks interpoliert werden. Dadurch kommt es zu Fehlern in der Gesamtsumme und zu Fehlern bei der Minimierung von $h_\gamma(\mathbf{P}|y)$.

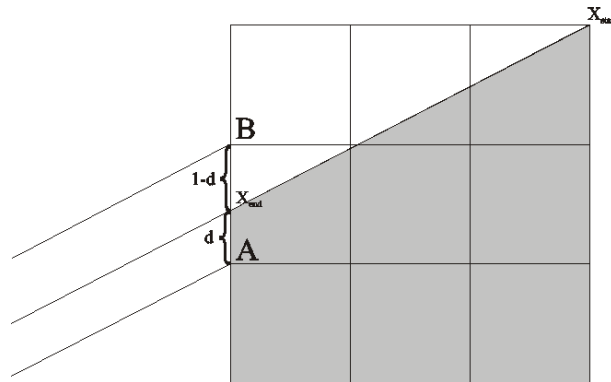


Abb. 16: Beispiel für Fehler bei der Interpolation

Für die Summe der Werte in der Dreiecksfläche wurde folgende lineare Interpolation verwendet: Für den Startpunkt x_{start} wird der reelle Endpunkt x_{end} , der sich auf der Kante k und dem linken Rand des Rechtecks befindet, extrapoliert. Dann wird auf dem Endpixel zwischen den bekannten Summen für A und B gemäß dem Abstand d linear interpoliert. Danach muss auch noch für das Rechteck unterhalb des Dreiecks dieselbe Interpolation durchgeführt werden. Die durch lineare Interpolation entstehenden Ungenauigkeiten können sich selbst über relative weite Entfernungen im Bild zu Fehlern addieren.

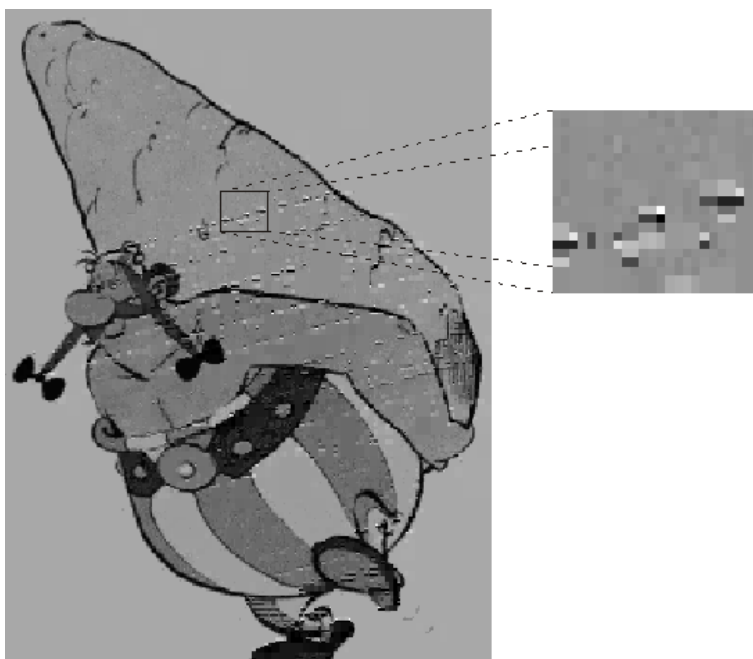


Abb. 17: Beispiel für Fehler mit der Interpolationsmethode bei Bild *Obelix*

Aufgrund der Fehler in dem Resultat und des hohen Tabellierungsaufwandes wurde diese Methode nicht weiter verwendet und stattdessen eine spezielle Methode zur Verrasterung aller Kanten mit pixelgenauen Berechnungen gewählt. Diese Methode wird im nächsten Abschnitt beschrieben.

4.3.5.2 Unterteilung in diskrete Linien Beim Diskretisieren von kontinuierlichen Linien kommt es zu folgendem Phänomen, das für Probleme der Art, wie sie schon oben beschrieben wurden (s. Abb. 14), verantwortlich gemacht werden kann. Ist s ein Pixel auf einer diskreten Linie mit Anfangspunkt a und Endpunkt b , so weicht die diskretisierte Linie von a nach s im Allgemeinen von dem Segment auf der Linie von a nach b ab. Es kann sogar vorkommen, dass die diskrete Linie von a nach b von der von b nach a abweicht. Dieses scheinbare Paradoxon würde sich, wie in folgender Abbildung gezeigt, auf den kontinuierlichen Fall übertragen.

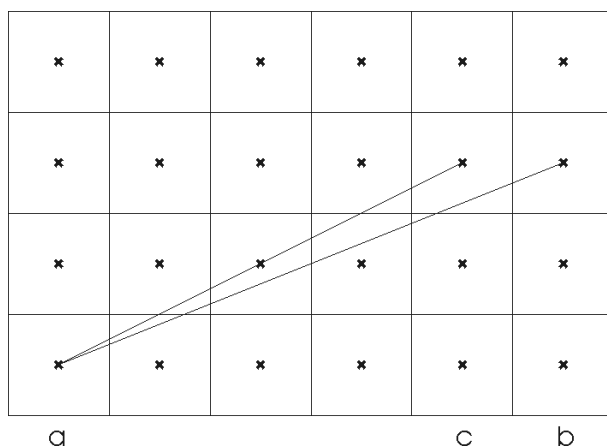


Abb. 18: Fehler zwischen den Kanten $K_{(a,b)}$ und $K_{(a,c)}$, kontinuierlich

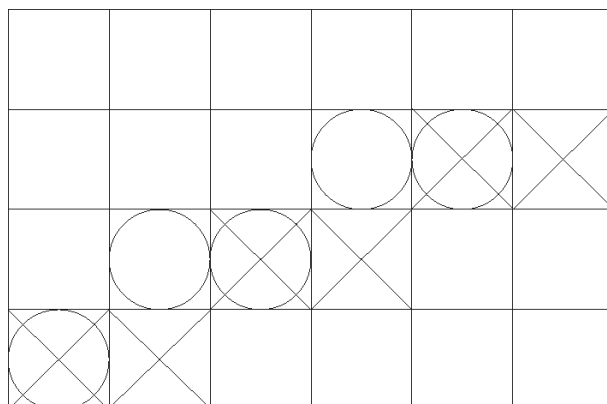


Abb. 19: Fehler in der Differenz durch die verrasterten Kanten, diskret

Dieses Problem lässt sich wie folgt lösen. Um zu gewährleisten, dass Paare von Punkten einer Linie immer genau diese Linie definieren, wird zu jedem angegebenen Winkel $\alpha \in \Delta$ die diskretisierte Kante $K_{\alpha,0}$ mit dem Bresenham Algorithmus berechnet. Durch vertikale (horizontale für $|\sin \alpha| > |\cos \alpha|$) parallele Verschiebung dieser Linie um j ($j \in \mathbb{N}$), erreicht man eine Zerlegung der Indexmenge S in diskrete Linien $K_{\alpha,j}$.

Für die Berechnung der optimalen Kante in einem Rechteck $Q \subset S$ werden nur noch Kanten der Form $K_{\alpha,j} \cap Q$ zugelassen. Zu auf dem Rand von Q liegenden Startpunkten müssen also Endpunkte gefunden werden, die auf den jeweils gleichen Kanten liegen. Jeder verschobenen Kante wird eine Nummer *lineNr* zugeordnet, die ihre Position in dem Bild bestimmt. Dadurch ist für jedes Pixel schnell bekannt, welche weiteren Pixel auf der gleichen Kante liegen.

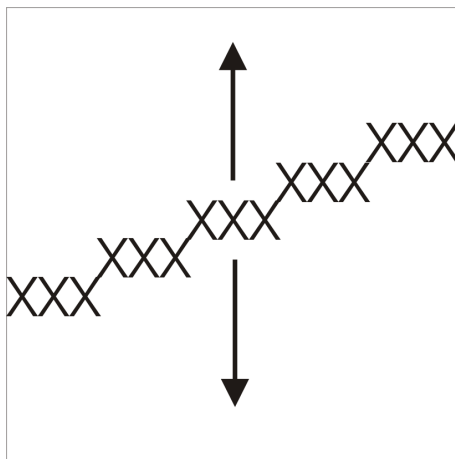


Abb. 20: Verschiebung der Kante durch das Bild

Auch eine Tabellierung kann mit den diskretisierten Kanten schnell durchgeführt werden, da für die kumulativen Summen in s auch der Wert der Summe des Vorgängers von s auf der selben Kante benutzt werden kann. Die Tabellierung der kumulativen Summen kann somit in $|S|$ Schritten durchgeführt werden.

Anschließend wird der Algorithmus beschrieben, der für die Diskretisierung der Kanten benutzt wurde.

Der Bresenham-Algorithmus Der Bresenham Algorithmus dient in der Computergraphik zur Verrasterung von Linien auf pixelorientierte Bildschirme. Die Linien verlaufen durch Start- und Endpunkt und der Wert von jedem Punkt wird durch den vorhergehenden bestimmt, wobei der Fehler zur exakten Linie minimiert wird (siehe auch BRE (8. 6. 2004)). Der Algorithmus beschränkt sich auf Geraden mit Steigung $0 \leq m \leq 1$, auf andere Steigungen erweitern lässt er sich einfach durch Spiegelung an der x- bzw. y-Achse, sowie durch Vertauschen von x - und y - Achse.

Begonnen wird beim Startpunkt (x_{start}, y_{start}) . Da die Steigung der Linie $0 \leq m \leq 1$ ist, wird vom Startpunkt beginnend die x-Achse durchlaufen bis zum Endpunkt (x_{end}, y_{end}) . Für jedes x_{i+1} kommen entweder y_i oder y_{i+1} in Frage. Es wird das Pixel gewählt, das näher beim exakten Wert der Gerade liegt, indem der Fehler der exakten Gerade zu y_i und y_{i+1} berechnet wird. Der Algorithmus kommt nur mit Integer-Arithmetik aus und hat eine Komplexität von $O(x_{end} - x_{start})$. Die Berechnung der Matrix der Dreieckssummen für eine Linie von jedem Punkt der Matrix hat die Komplexität $O(|S|)$.

Algorithmus 4 (Bresenham-Algorithmus für $0 \leq m \leq 1$)

```
 $dx := x_{end} - x_{start}; dy := y_{end} - y_{start}$   
 $incE := 2 * dy; incNE := 2 * (dy - dx)$   
 $x := x_{start}; y := y_{start}$   
 $d := 2 * dy - dx$   
WHILE  $x \leq x_{end}$  DO  
   $SetPixel(x, y);$   
  IF  $d < 0$  THEN  $d := d + incE;$     {East wird gewählt}  
  ELSE  $d := d + incNE; y := y + 1;$   {NE wird gewählt}  
  END;  
   $x := x + 1$   
END;
```

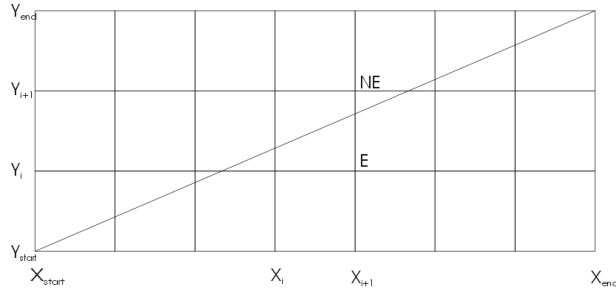


Abb. 21: Bresenham-Algorithmus

4.3.5.3 Komplexität

Mit den letzten beiden Abschnitten ist klar: Die lokale Minimierung benötigt sowohl mit der Interpolationsmethode als auch mit Diskreten Linien $c\sqrt{N}|\Delta|$ Schritte, da lediglich Randpunkte des betrachteten Rechteckes als Startpunkte zugelassen werden und die Momente zu jedem Winkel und jedem Startpunkt in einer festen Anzahl Schritten berechnet werden können.

Dies führt gemeinsam mit Lemma 4.27 mit endlicher Menge erlaubter Winkel Δ zu folgendem wichtigem Ergebnis:

Satz 4.29 *Werden in jedem Rechteck die Randpunkte als Startpunkte erlaubter Kanten eingesetzt, so ergibt sich unter Benutzung einer vorberechneten Tabellierung eine Gesamtkomplexität des Algorithmus 3 von $O(|S| \cdot |\Delta|)$. Der Algorithmus hat einen Speicherbedarf von $O(|S| \cdot |\Delta|)$.*

Für die Tabellierung werden bei Einsatz der Interpolationsmethode $O(|S|^{3/2} \cdot |\Delta|)$ und bei Benutzung der Methode der Diskreten Linien $O(|S| \cdot |\Delta|)$ Schritte benötigt.

5 Software

Die Algorithmen zur Minimierung des Funktionals h_γ wurden in der Programmiersprache Oberon verfasst. Benutzt wurden die Softwarepakete ANTS_{InFields} (F. FRIEDRICH (2002), F. FRIEDRICH (2003)) und Voyager (G. SAWITZKI (1996)).

Die Berechnung der optimalen Partition vollzieht sich in drei Schritten, die getrennt implementiert wurden und auch einzeln abrufbar sind.

1. Tabellierung der benötigten Matrizen zur effizienten Momentenberechnung zu vorgegebener Winkelmenge,
2. Erzeugung des gesamten quadrees inclusive Berechnung der optimalen Zerlegung für jeden quadsplit,
3. Auswahl der optimalen Partition zu vorgegebenem $\gamma \geq 0$.

Die Aufteilung in diese drei Schritte hat folgende Gründe: Zum einen ist im Sinne einer objektorientierten Programmierung eine Aufteilung des Codes in inhaltlich wiederverwendbare Komponenten erwünscht. Schon der Aufbau dieser Arbeit zeigt die inhaltliche Trennung der Schritte 1., 2. und 3. Zum anderen hat die Trennung auch praktische Gründe. Bei der Erzeugung des quadrees können Parameter eingehen, wie z.B. maximale Baumtiefe, maximale Wedge-Größe etc., die von den Werten der Momente völlig unabhängig gewählt werden können. Eine wiederholte Erzeugung eines quadrees mit anderen Parametern soll eben keine Neuberechnung der Momente voraussetzen. Gegen die Trennung von 1. und 2. spricht der relativ große Speicherbedarf für die Matrizen und diese müssen bei einer solchen Trennung im Speicher behalten werden. Die Trennung zwischen Schritt 2 und 3 hingegen bringt erhebliche Vorteile: Die Erzeugung des quadrees ist mit erheblichem Rechenaufwand verbunden und stellt somit im Allgemeinen den Schritt mit längster Berechnungsdauer dar. Ist der quadtree einmal komplett vorhanden ist die Adaption an den Parameter γ in sehr kurzer Zeit zu bewerkstelligen. Es müssen dazu nur die Abweichungsquarate an den Knoten gemäß Algorithmus 3 verglichen werden. Tatsächlich ist der Zeitaufwand für Schritt 3

selbst für Bilder der Größen 512×512 auf einer Pentium4 Maschine mit 2.8GHz im Millisekundenbereich, so dass verschiedene γ -Werte auf einem Bild in Echtzeit betrachtet und ausgewählt werden können.

Neben der Implementierung der eigentlichen Algorithmen mussten einige Datenstrukturen bereitgestellt werden:

- Ein abstraktes Linienobjekt zur Partition des Bildes in disjunkte parallelverschobene Linien: `pxlData.Line`,
- Container, die die kumulativen Matrizen und deren Winkeldaten beinhalten: `pxlData.AngleData`,
- die Baumstruktur quadtree: `pxlTree.Tree`,
- ein Datenobjekt zum Einhängen in den quadtree: `pxlTree.Data`.

Zur Visualisierung wurden im wesentlichen die Plots des Softwarepakets `ANTSInFields` und `Voyager` benutzt, zur Visualisierung des quadtrees wurde zusätzlich ein `TreePlot` implementiert, mit dem die eigentliche Partition sichtbar gemacht werden kann.

Es sind folgende Module entstanden:

<code>pxlData.Mod</code>	Beinhaltet Datenstrukturen <code>Line</code> und <code>AngleData</code> . Stellt Berechnung der kumulativen Matrizen und Operationen auf diskreten Linien zur Verfügung
<code>pxlTree.Mod</code>	Beinhaltet die Datenstruktur <code>Tree</code> und <code>Data</code> . Stellt Erzeugung und Berechnung des quadtrees bereit, ausserdem die Konstruktion des Ergebnisbildes aus einem quadtree. Benutzt <code>pxlData</code> .
<code>pxlStatistic.Mod</code>	Stellt Berechnungsprozeduren für die Ermittlung der besten lokalen Zerlegung durch Einziehen einer Kante bereit. Berechnung von Mittelwert und Abweichungsquadrat zu dieser Kante unter Benutzung der kumulativen Summen. Benutzt <code>pxlData</code> .
<code>pxlPrep.Mod</code>	Vorbereitsprozeduren, Einlesen von Parametern etc.

- pxlMin.Mod Die eigentliche Minimierungsprozeduren und quadtree-
Erzeugung. Benutzt pxlTree, pxlStatistic und pxlData.
- pxlTreePlot.Mod Graphische Darstellung des quadtree. Benutzt pxlTree
und pxlData.
- pxlWedget.Mod Basismodul. Stellt Interaktion mit dem Benutzer bereit
und beinhaltet zusätzliche kleine Tools zur Erstellung
von PSNR Daten etc., benutzt alle anderen Module.

Neben der Erstellung der eigentlichen Algorithmen und Datenstrukturen wurde ein graphisches Benutzerinterface bereitgestellt. Dazu wurde in Oberon ein so genanntes Panel erstellt, das sich im Betriebssystem Windows wie ein gewöhnliches Fenster präsentiert. Mit diesem Panel können Bilddaten geladen, der Algorithmus mit verschiedenen Parametern angewendet, das Ergebnis betrachtet und auch gespeichert werden.

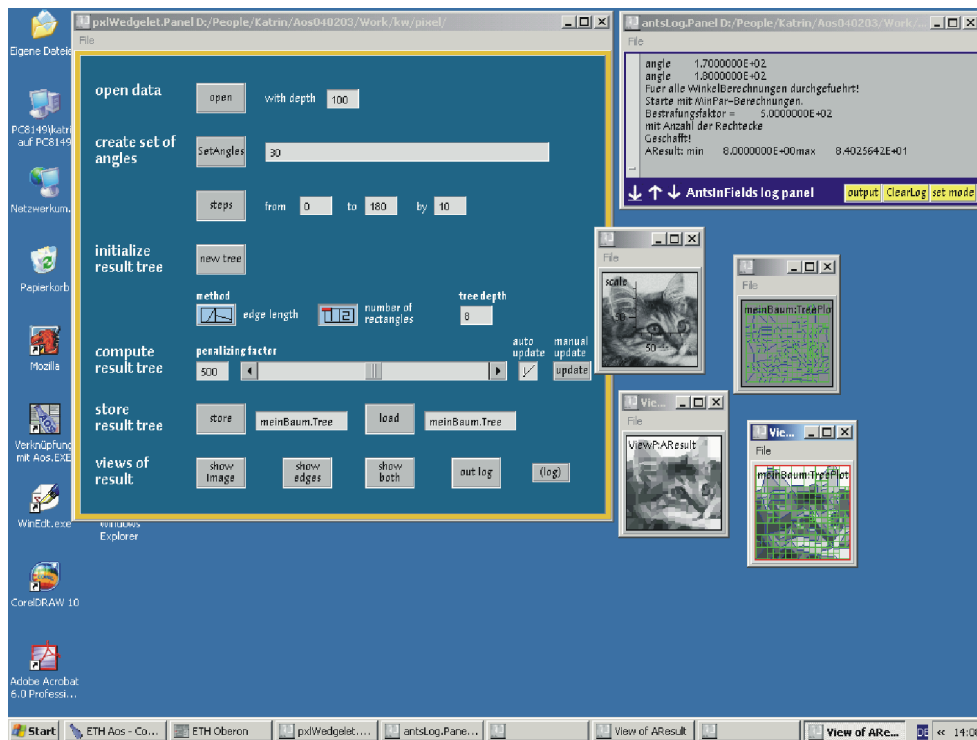


Abb. 22: Oberfläche

Es folgt eine Kurzbeschreibung zur Benutzung des Panels für das zweidimensionale Wedgelet-Modell.

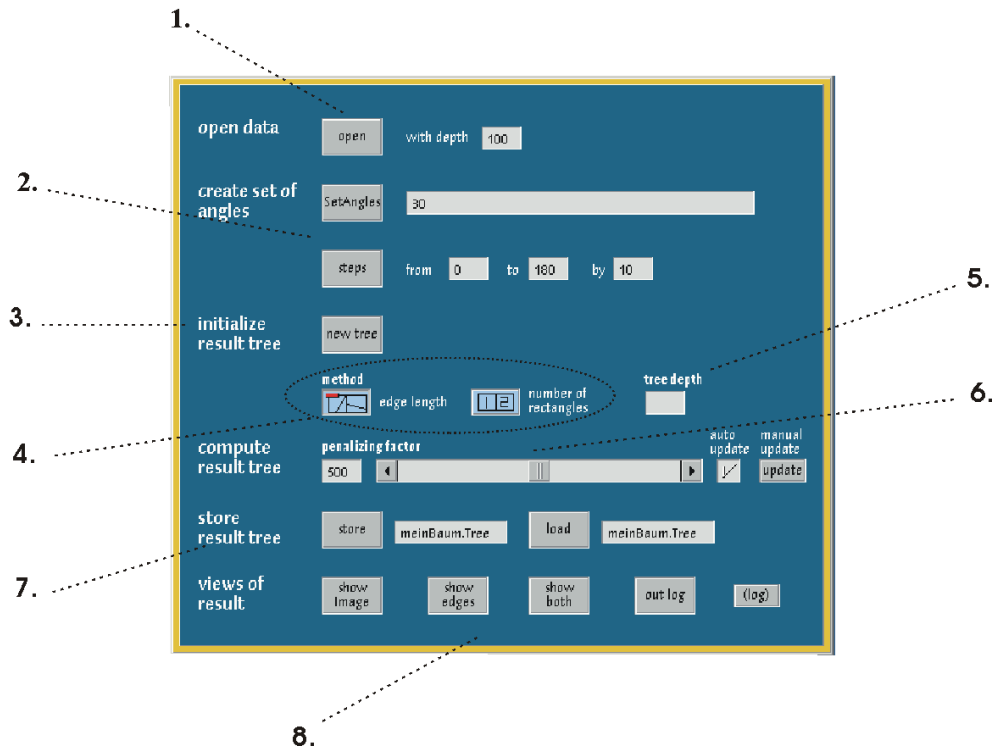


Abb. 23: Panel

Die Daten (Bilder), auf die der Algorithmus angewendet werden soll, werden unter **1.** geladen. Betätigen der Maustaste auf dem Knopf *open* öffnet ein Fenster, in dem ein Bild gesucht und geöffnet werden kann. Bei dem Eingabefeld rechts neben *open* lässt sich zuvor noch die gewünschte Graustufenanzahl für das Datenbild festlegen (in diesem Beispiel ist die Graustufenanzahl 100). Als nächstes werden unter **2.** die kumulierten Matrizen (Zeilen- und Dreiecksmatrizen) für die verschiedenen Winkel berechnet (vgl. Kapitel 4.3.5). Für die Eingabe der Winkel gibt es zwei Möglichkeiten: Zum Einen kann der Benutzer die gewünschten Winkel per Hand in beliebiger Reihenfolge und durch ein Leerzeichen getrennt in die Eingabezeile schreiben und schließlich mit *SetAngles* die Berechnungen starten. Zum anderen gibt es die Möglichkeit, für alle Winkel innerhalb der Grenzen *from* bis *to* mit der Schrittweite *by*

durch den Knopf *steps* die Berechnungen durchzuführen. Bei **3.** bietet sich die Option, einen neuen Baum zu initialisieren, der die für die Partition notwendigen Daten abspeichert (vgl. Definition 4.19). Dadurch kann ein Vergleich mit Ergebnissen erfolgen, die mit anderen Einstellungen entstanden. Wird kein neuer Baum initialisiert, so wird ein zuvor angelegter Baum benutzt, die Ergebnisse aus diesem Baum werden überschrieben. Bei **4.** kann die Methode ausgewählt werden, die für den Regulierungsterm angewendet werden soll. (Wahl zwischen Kantenlänge und der Anzahl der Rechtecke der Partition, was in den Symbolen angedeutet ist (vgl. Kapitel 4.1).) Als nächstes lässt sich bei **5.** die Anzahl der maximalen Unterteilungen (quad-splits) angeben. Wird dieses Feld freigelassen, so erscheint nach dem Durchführen des Algorithmus die Anzahl der tatsächlich angewendeten quad-splits.

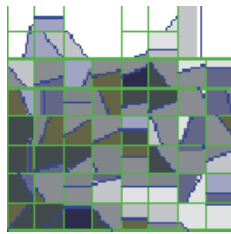


Abb. 24: Beispiel für maximal 4 quad-splits

Schliesslich wird bei **6.** der Bestrafungs-Parameter γ entweder in dem Eingabefeld oder in der verschiebbaren Bildlaufleiste eingestellt. Wird für die Eingabe das Eingabefeld benutzt, so muss danach für die Berechnung der optimalen Partition noch der Knopf *update* gedrückt werden. Ist der Knopf *autoupdate* gewählt und erfolgt die Eingabe des Parameters über die Bildlaufleiste, so wird automatisch der Algorithmus gestartet. Ist hingegen *autoupdate* nicht gewählt, so muss die Berechnung für die optimale Partition immer noch über *update* erfolgen. Für einen Vergleich der beiden Methoden wurde die Bestrafungsparameter für die unterschiedlichen Methoden angeglichen : $\gamma_{len(edge)} := \frac{\gamma_{\#Rechtecke}}{\sqrt{w+h}}$. Weiter gibt es bei **7.** die Möglichkeit, den entstandenen Baum unter einem Namen in dem Feld *meinBaum.Tree* abzuspeichern, oder einen bereits gespeicherten Baum über seinen Namen zu laden. Eine Darstellung des (entweder des berechneten oder des geladenen) Baumes wird schließlich bei **8.** ermöglicht. Hier kann unter *showimage* das Resultat der Partition angezeigt werden,



Abb. 25: *showimage*
für die eingestellten
Parameter

unter *showedges* wird die Partition nur durch die entstandenen dyadischen Quadrate und die eingefügten Kanten angezeigt

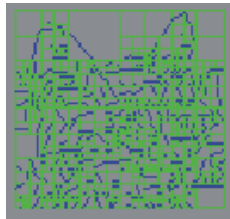


Abb. 26: *showedges*

und unter *showboth* werden das Resultat der Daten zusammen mit den Unterteilungen und den eingefügten Kanten dargestellt.

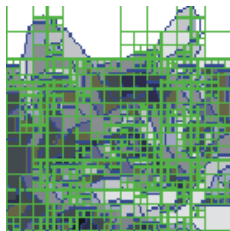


Abb. 27: *showboth*

Rechts daneben befindet sich noch ein *outLog*-Knopf, der ein einfaches Fenster öffnet, in dem die vorher beschriebenen Darstellungen in beliebiger Reihenfolge zusammenkopiert und gespeichert werden können. Der *(log)*-Knopf

rechts unten bietet die Möglichkeit, ein Log-Fenster zu öffnen, um während der Berechnungen die entstehenden Ausgaben mitverfolgen zu können.



Abb. 28: (log)

6 Anwendungen

6.1 Kompression

Um das in dem vorherigen Abschnitt beschriebene Verfahren hinsichtlich seiner Verwendung zur Bildkompression zu beurteilen, wurde der Algorithmus an einigen Beispielen mit einer modifizierten Standardauswertung untersucht.

Für eine Auswertung müssen folgende Größen bestimmt werden:

Mit einem fest vorgegebenem γ wird der Algorithmus auf ein Bild x der Größe $w \cdot h$ und einem Graustufenvorrat d (*depth*) angewendet. Dabei wird die optimale Segmentation P und die Anzahl der benötigten Komponenten n in der dazugehörigen Rekonstruktion \hat{x}_P bestimmt. Ein Quadrat mit zusätzlicher Unterteilung durch eine Winkelgerade besteht aus zwei Komponenten, ein Quadrat ohne eine weitere Winkelunterteilung besteht aus einer Komponente.

Ein oft verwendetes objektives Qualitätsmaß für die Güte von komprimierten Bildern ist der *PSNR* Wert (*Peak Signal to Noise Ratio*), gemessen in Dezibel dB (siehe L. DEMARET et al. (2004)).

Der *PSNR* Wert berechnet sich mit

$$PSNR = 10 * \log_{10} \left(\frac{d^2}{MSE} \right),$$

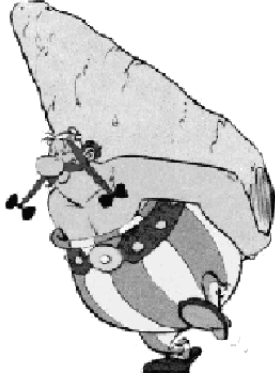
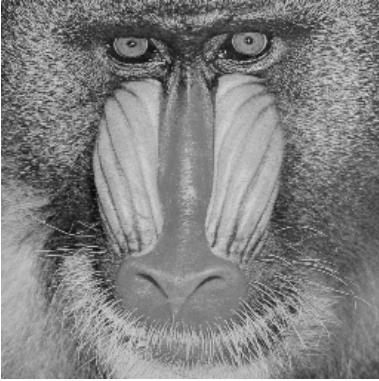
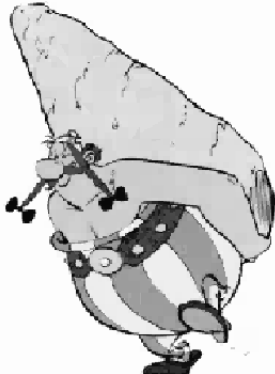
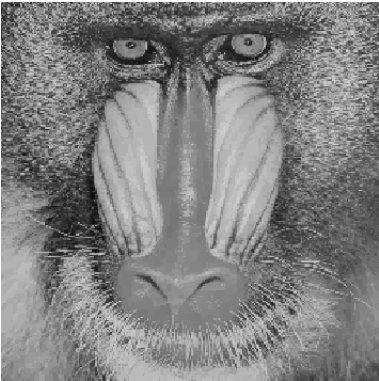

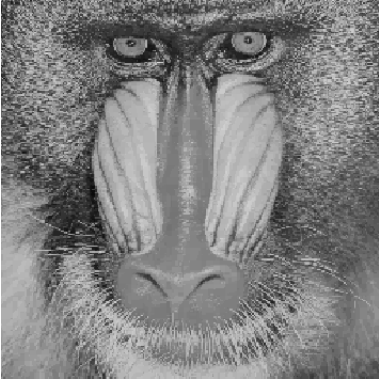
wobei *MSE* den mittleren quadratischen Fehler (*Mean Square Error*)

$$MSE = \frac{ssq(\mathbf{P})}{wh} = \frac{\|x - \hat{x}_P\|_2^2}{wh}$$

für die Segmentation \mathbf{P} bezeichnet.

Die berechneten *PSNR* Werte lassen sich grob in Bereiche unterteilen, die für die Qualität der komprimierten Bilder folgende Bedeutung haben:

Eine Rekonstruktion mit *PSNR* Werte Größer als 35 dB gilt als sehr gut und lässt sich kaum vom Originalbild unterscheiden. Bei *PSNR* Werten zwischen 30 dB und 35 dB ist die Rekonstruktion immer noch gut, aber kleinere Schäden sind erkennbar. Rekonstruktionen mit *PSNR* Werten zwischen 25 dB und 30 dB sind noch wiedererkennbar, *PSNR* Werte kleiner als 25 dB weisen im Allgemeinen aber grobe Fehler auf.

PSNR - Bereich	Inter- preta- tion	 <p>Beispiel <i>Obelix</i></p>	 <p>Beispiel <i>Baboon256</i></p>
> 35	sehr gut	 <p><i>PSNR = 35.129990</i></p>	 <p><i>PSNR = 35.192317</i></p>
30 – 35	gut	 <p><i>PSNR = 30.072943</i></p>	 <p><i>PSNR = 30.091885</i></p>

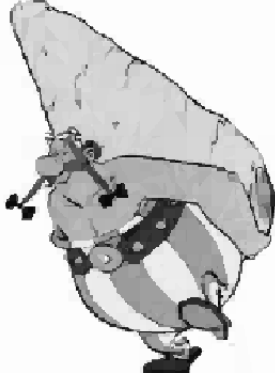
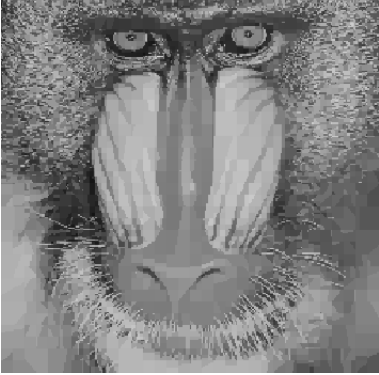
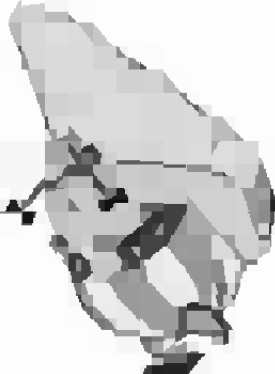

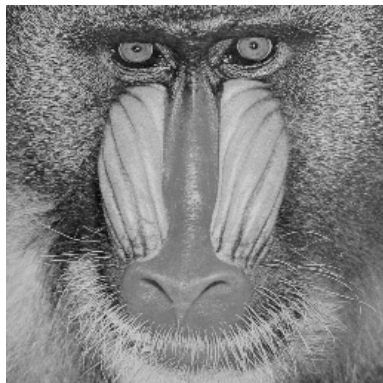
25 – 30	kleinere Fehler	 <p style="text-align: center;"><i>PSNR</i> = 25.020934</p>	 <p style="text-align: center;"><i>PSNR</i> = 25.086845</p>
< 25	Fehler	 <p style="text-align: center;"><i>PSNR</i> = 20.198399</p>	 <p style="text-align: center;"><i>PSNR</i> = 20.006227</p>

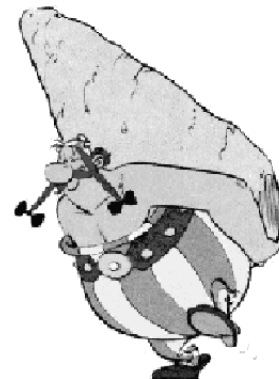
Abb. 29: Die Bereiche der *PSNR* Werte und Beispiele an den Bildern *Obelix* und *Baboon*

Für eine Untersuchung der Auswertungen wurde der Algorithmus auf ein Bild mehrere Male mit verschiedenen Bestrafungsparameter γ und mehreren Winkelmengen angewendet. Bei jeder Anwendung des Algorithmus wurde der *PSNR* Wert und die Anzahl n der verwendeten Komponenten für die Segmentation bestimmt und graphisch dargestellt.

Das beschriebene Verfahren wurde auf folgende Bilder angewendet:
Baboon (Größe 256×256), *Uhr* (512×512), *Peppers* (512×512), *Fruits* (512×512) und *Obelix* (256×256). Alle Bilder besitzen den Graustufenvorrat $d = 256$.



Baboon (Größe 256×256)



Obelix (Größe 256×256)



Peppers (Größe 512×512)











Fruits (Größe 512×512)



Uhr (Größe 512 × 512)

Die verschiedenen Bestrafungsparameter wurden gewählt, indem von $\gamma = 0,25$ startend in 75 Iterationsschritten für den Wert des aktuellen γ das vorherige immer mit dem Faktor 1,3 multipliziert wurde.

Für die Winkelmengen wurden die Winkel zwischen 0° und 180° in jeweils einer festen Schrittweite durchlaufen. Es wurden folgende 8 verschiedene Winkelmengen festgelegt und jeder Winkelmenge wurde jeweils eine Farbe zugeordnet, die die Kurve in den Abbildungen kennzeichnet:

Farben	Winkel
 (türkis)	from 0° to 180° by 1°
 (rot)	from 0° to 180° by 5°
 (blau)	from 0° to 180° by 10°
 (grün)	0° 30° 60° 90° 120° 150° 180°
 (pink)	0° 45° 90° 135° 180°
 (gelb)	0° 60° 120° 180°
 (dunkelgrün)	0° 90° 180°
 (schwarz)	\emptyset

Auf ein Bild wurde für jede der acht verschiedenen Winkelmengen der Algorithmus angewendet. In jedem Iterationsschritt für den Bestrafungsparameter γ wurde jeweils der *PSNR* Wert und die Anzahl der in der Segmentation enthaltenen Komponenten n bestimmt und anschließend wurde die Kurve graphisch aufgetragen.

Als nächstes folgen die Abbildungen der Auswertungen zu den 5 Beispielen unter den angegebenen Bestrafungsparametern und den festgelegten Winkelmengen, sowie eine Abbildung der Kurven ohne Winkelunterteilung, in 10° - und in 30° - Abständen von allen Bildern:

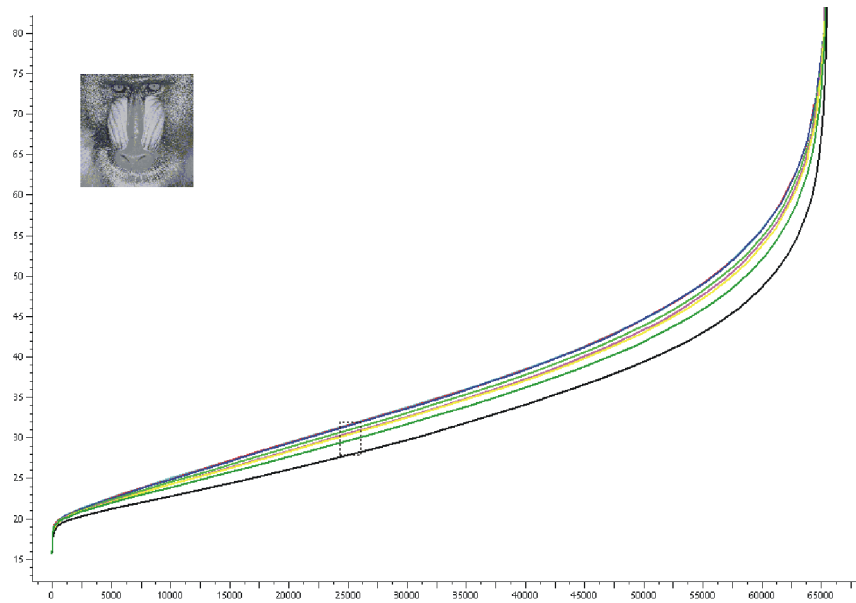


Abb. 30: *PSNR* Wert gegen Anzahl der Komponenten für Bild *Baboon*

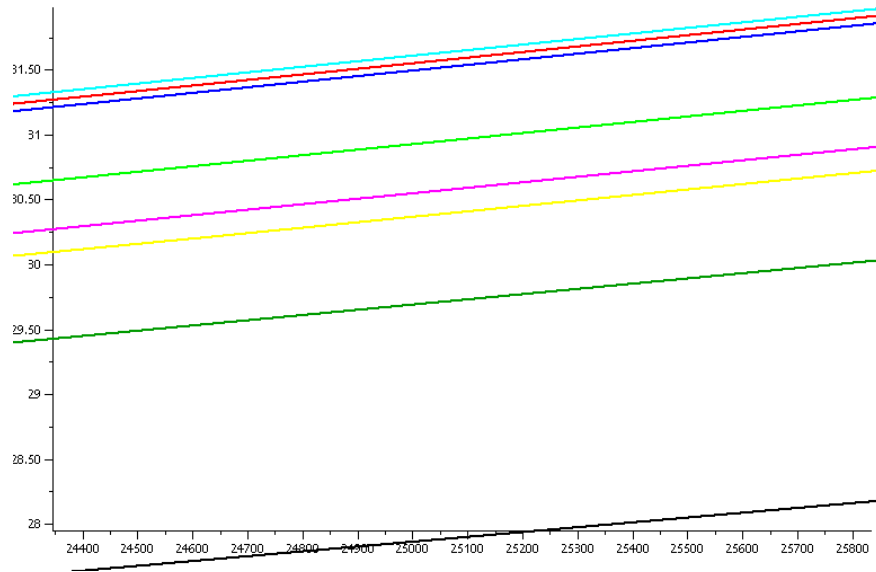


Abb. 31: Ausschnitt aus der Auswertung zu *Baboon*

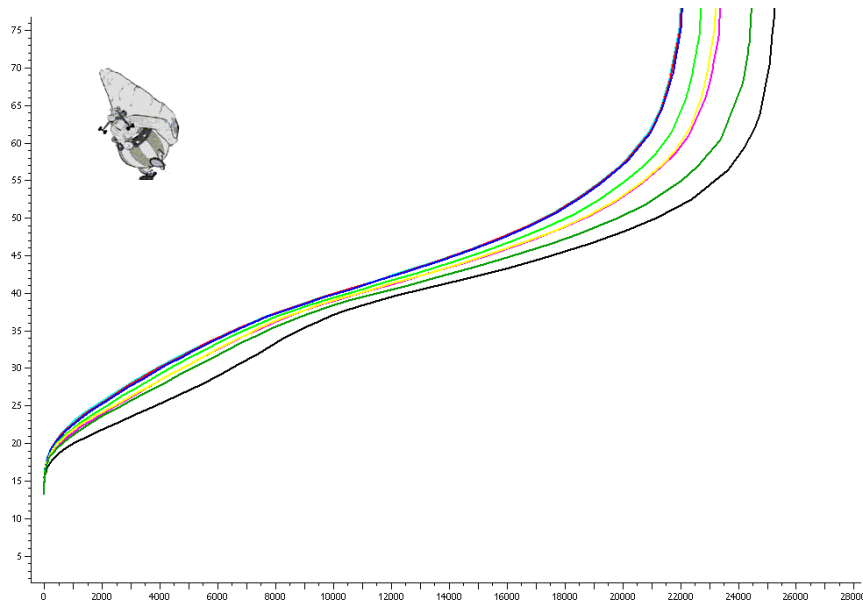


Abb. 32: *PSNR* Wert gegen Anzahl der Komponenten für Bild *Obelisk*

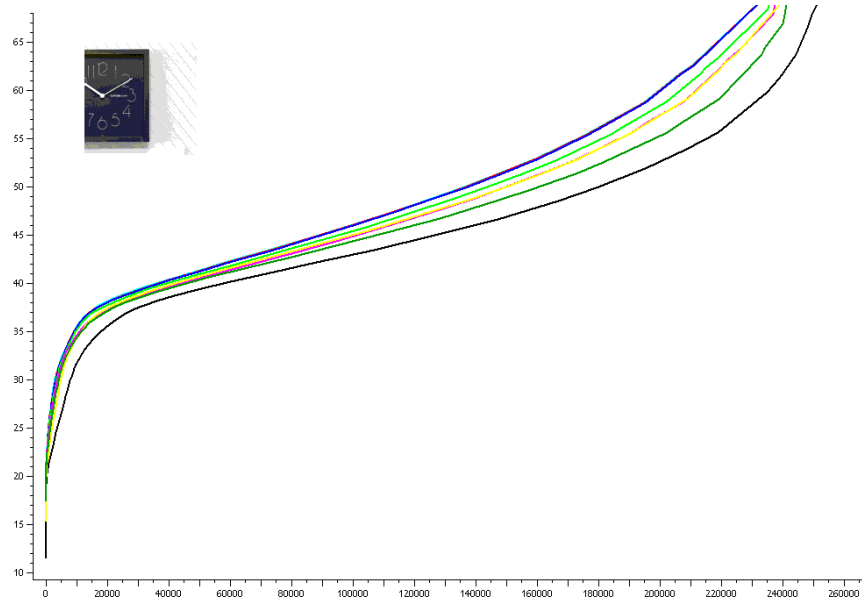


Abb. 33: *PSNR* Wert gegen Anzahl der Komponenten für Bild *Uhr*

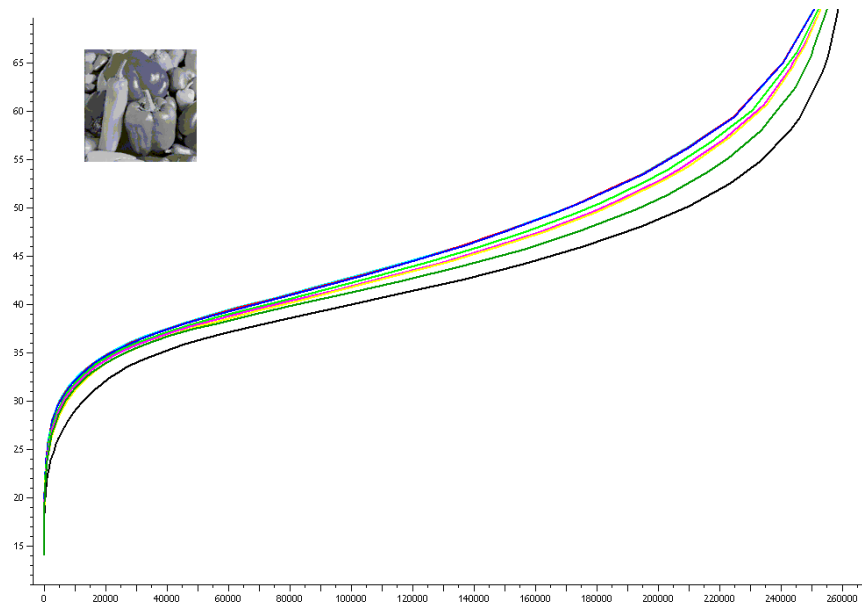


Abb. 34: *PSNR* Wert gegen Anzahl der Komponenten für Bild *Peppers*

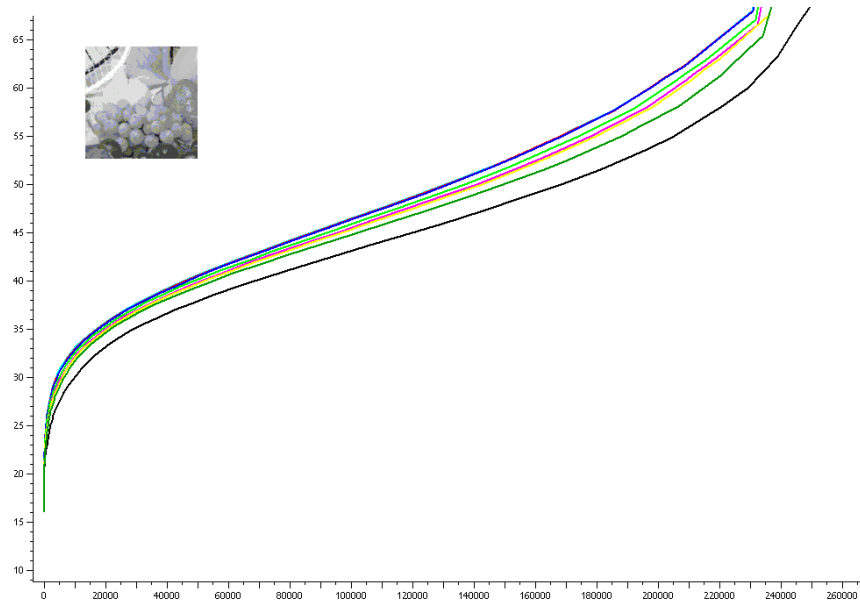


Abb. 35: *PSNR* Wert gegen Anzahl der Komponenten für Bild *Fruits*

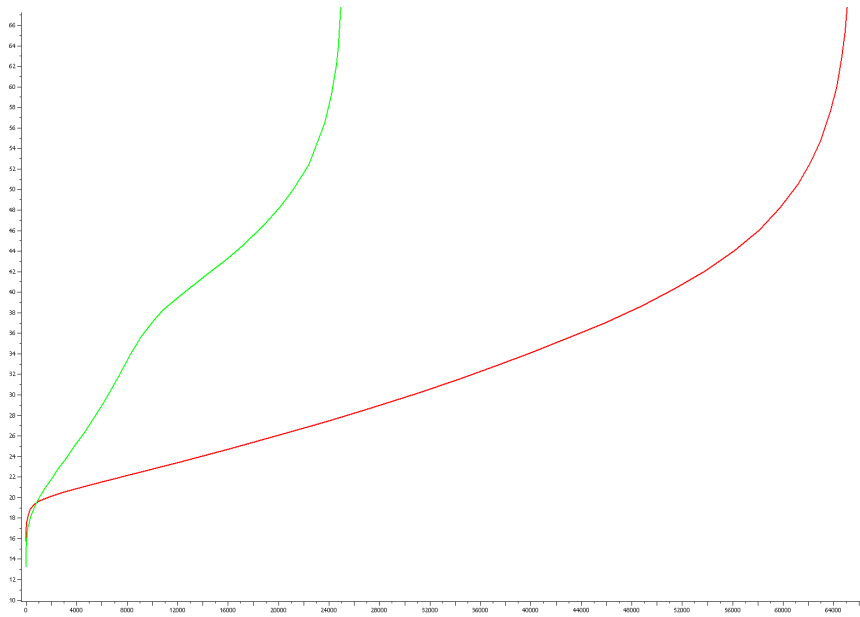


Abb. 36: Kurven ohne Winkelunterteilung für die Bilder der Größe 256×256 *Baboon* (rot) und *Obelix* (grün)

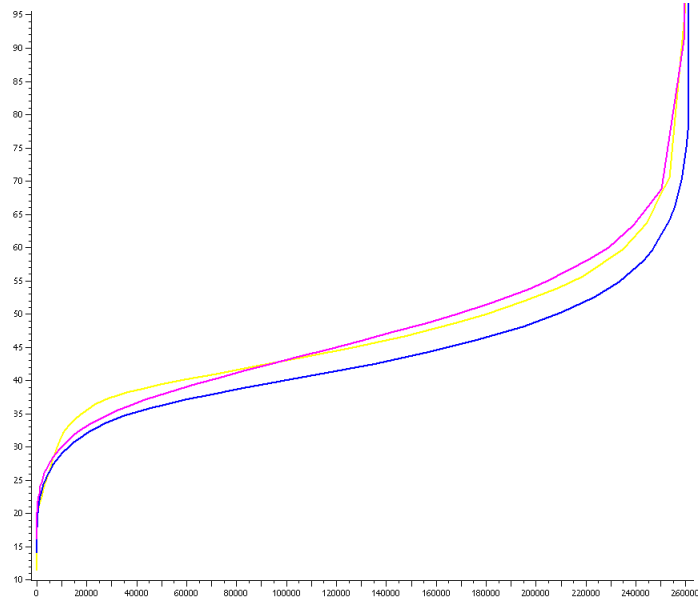


Abb. 37: Kurven ohne Winkelunterteilung für die Bilder der Größe 512×512 *Uhr* (gelb), *Peppers* (blau) und *Fruits* (pink)

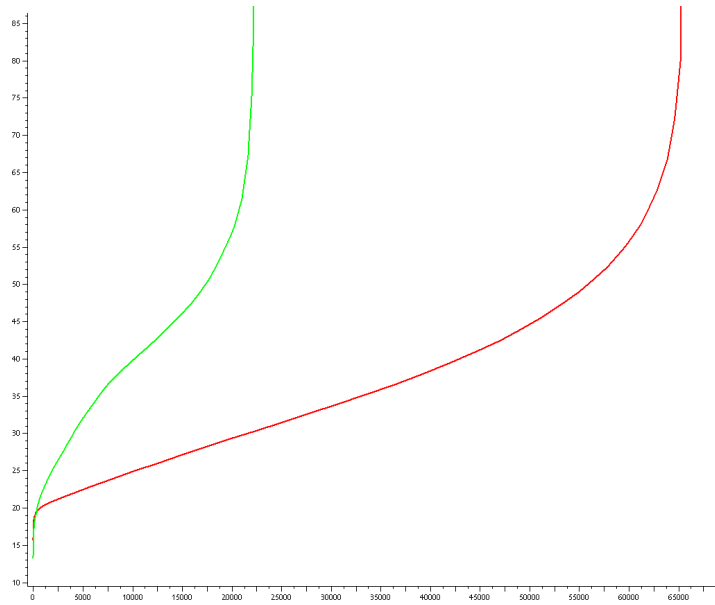


Abb. 38: Kurven der Winkel in 10° Abständen für die Bilder der Größe 256×256 *Baboon* (rot) und *Obelix* (grün)

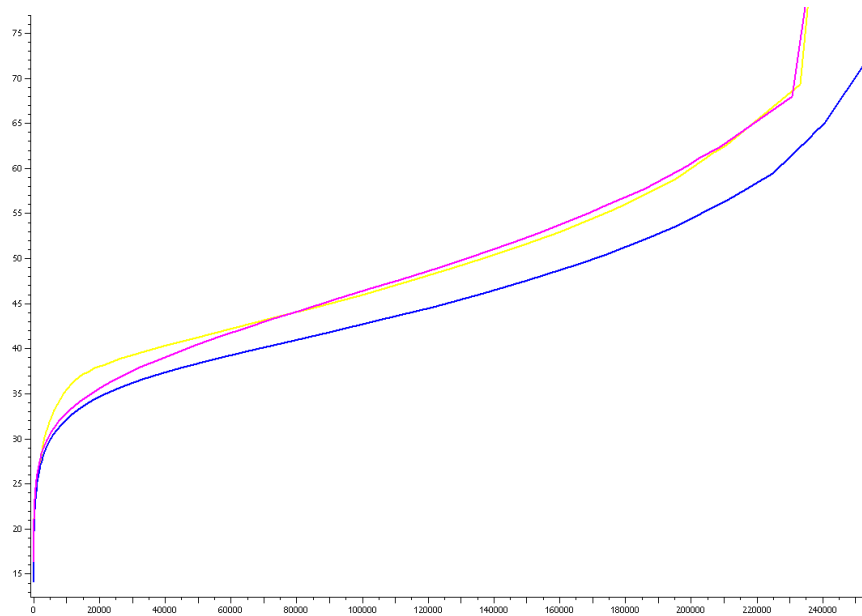


Abb. 39: Kurven der Winkel in 10° Abständen für die Bilder der Größe 512×512 *Uhr* (gelb), *Peppers* (blau) und *Fruits* (pink)

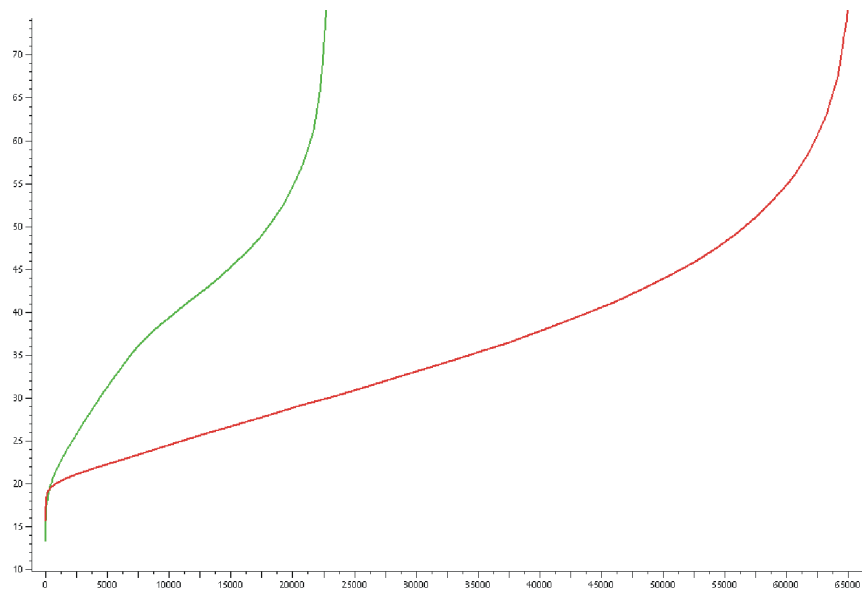


Abb. 40: Kurven der Winkel in 30° Abständen für die Bilder der Größe 256×256 *Baboon* (rot) und *Obelix* (grün)

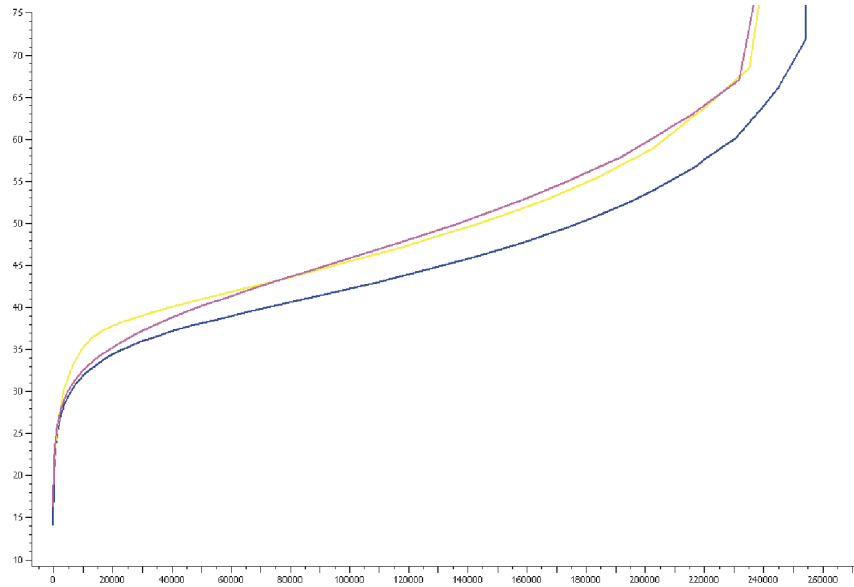


Abb. 41: Kurven der Winkel in 30° Abständen für die Bilder der Größe 512×512 *Uhr* (gelb), *Peppers* (blau) und *Fruits* (pink)

Für eine Beurteilung der Abbildungen lässt sich feststellen, dass sich die Graphiken für jedes Bild in ihrer Form ähneln. Alle Kurven der verschiedenen Abbildungen starten bei $PSNR$ Werten zwischen 10 dB und 15 dB für eine Komponente. Dann lassen sich die Kurven in drei Abschnitte unterteilen. Im ersten Abschnitt besitzen alle Kurven einen steilen Anstieg. Dieser kommt dadurch zustande, dass für eine kleine Anzahl von Komponenten die Bildqualität schnell stark zunimmt, wenn wenige weitere Unterteilungen hinzukommen. Diese schnelle Verbesserung wird gefolgt von einer längeren flachen Steigung, in der die Bildqualität durch weitere Unterteilungen nur langsam zunimmt. Nach diesem langen mittleren flachen Teil folgt ein steiler asymptotischer Anstieg. Diese letzte Steigung zeigt, dass ab einer gewissen Anzahl von Komponenten die Bildqualität nicht mehr verbesserbar ist.

Bei den Bildern *Fruits* und *Uhr* ist bei einigen Kurven (z.B. *dunkelgrün* bei beiden Bildern) im letzten Abschnitt ein Sprung zu unendlichen $PSNR$ Werten, also zu Rekonstruktionen ohne Fehler, zu beobachten. Wird durch eine weitere Unterteilung in einem Bild der Größe 256×256 mit 256 Graustufen nur ein Pixel um eine Graustufe variiert, so ergibt sich für den MSE :

$$MSE = \frac{1}{wh}$$

und somit für den *PSNR* Wert:

$$\begin{aligned}PSNR &= 10 \log_{10} d^2 wh \\ &= 10 \cdot 4 \cdot \log_{10} 256 \approx 100\end{aligned}$$

Daher kann durch Veränderung von nur einem Pixel ein sprunghafter Anstieg des *PSNR* Wertes (von 0 zu ∞) zustande kommen.

Weiter besitzen die Kurven für die unterschiedlichen Winkelmengen immer entsprechend der Inklusionsbeziehung der Winkelmengen dieselbe Reihenfolge. Die Kurve für die Winkel in 10° Abständen (*blau*) ist bei allen Bildern fast identisch mit den Kurven in 1° (*türkis*) und in 5° Abständen (*rot*). Das zeigt, dass eine feinere Unterteilung als die 18 Winkel für die 10° Abstände keine deutliche Verbesserung für die Qualität der komprimierten Bilder mit sich bringt.

Die Kurve für die Segmentationen ohne weitere Winkelunterteilung ist bei allen Beispielbildern die schlechteste Variante und liegt jeweils etwa 2 dB unterhalb der Kurven mit Winkelunterteilung.

Die Graphiken unterscheiden sich für Bilder mit großen konstanten Flächen (*Obelix*) bei denen jede Kurve eine eigene Asymptote besitzt, von den Bildern mit hauptsächlich unterschiedlichen Pixeln (*Baboon*) bei denen alle Kurven bei derselben Anzahl von Komponenten eine senkrechte Asymptote besitzen, unabhängig von der Feinheit der benutzten Winkel. Dies liegt daran, dass bei glatten Bildern mit großen konstanten Flächen und mit vielen Winkeln wesentlich weniger Komponenten benötigt werden für ein gutes oder gar perfektes Ergebnis und früher ein höherer *PSNR* Wert erreicht wird. Für weniger Winkel sind dagegen feinerer Unterteilungen nötig und dadurch erhöht sich die Anzahl der benötigten Komponenten. Bilder mit vielen Texturen (*Baboon*) und ohne größere einheitlichen Flächen müssen dagegen unabhängig von den vorhandenen Winkeln nahezu bis auf Pixelebene unterteilt werden um Rekonstruktionen ohne Fehler zu erreichen. Dies zeigt sich auch in der Anzahl der Komponenten. Obwohl beide Bilder dieselbe Größe besitzen (256×256), wird bei *Baboon* in $65536 = 256^2$ Komponenten unterteilt, bei *Obelix* nur - abhängig von der Winkelmenge - in 22000 bis 25000 Komponenten.

6.2 Laufzeitvergleich

Es wurde ein Laufzeitvergleich des Wedgelet-Algorithmus mit *BeamLab* (s. BEA (8. 6. 2004)) durchgeführt. Die Messungen wurden mit Bildern der Größe 16×16 , 32×32 , 64×64 , 128×128 , 256×256 und 512×512 durchgeführt. Für die Laufzeiten ist nur die Größe entscheidend, um was für Bilder es sich handelt spielt hier keine Rolle.

Die Messungen wurden auf einer Pentium4 Maschine mit 2.8GHz durchgeführt, *BeamLab* wurde in der Matlab Version 6 benutzt.

Der Algorithmus des Wedgelet-Algorithmus unterteilt sich in folgende 3 Schritte (vgl. Abschnitt 4.3.4):

- Tabellierung der Matrizen
- Berechnung der optimalen Partition
- Rekonstruktion aus dem quadtree

Die Zeiten für die Rekonstruktion aus dem quadtree sind vernachlässigbar klein und spielen im weiteren für die Gesamtlaufzeit keine Rolle. Für die Tabellierung der Matrizen bei dem Wedgelet-Modell für unterschiedliche Winkelmengen wurden bei den verschiedenen Bildgrößen folgende Zeiten benötigt:

Bildgröße	ohne Winkel	3 Winkel	5 Winkel	10 Winkel	181 Winkel
16×16	$< 0.01s$	$< 0.01s$	$< 0.01s$	$< 0.01s$	$< 0.01s$
32×32	$< 0.01s$	$< 0.01s$	$< 0.01s$	$0.02s$	$0.1s$
64×64	$0.04s$	$0.01s$	$0.02s$	$0.03s$	$0.3s$
128×128	$0.3s$	$0.1s$	$0.1s$	$0.4s$	$4.2s$
256×256	$0.1s$	$1.9s$	$2.0s$	$5.1s$	$8.7s$
512×512	$4.0s$	$7.7s$	$9.9s$	$21.3s$	$148s$

Die Zeiten für die Berechnungen des Algorithmus *BeamLab* und des Wedgelet Modells für Bilder der Größe (16×16) , (32×32) , (64×64) , (128×128) , (256×256) und (512×512) wurden in folgender Tabelle aufgetragen:

Bild der Größe	16×16	32×32	64×64	128×128	256×256	512×512
<i>BeamLab</i>	6.72s	45.34s	330.41s $\approx 5.5min$	2676.3s $\approx 44.6min$	27918s $\approx 7h45min$	> 12h
<i>Wedgelet-Modell</i> 0° to 180° by 1°	0.016s	0.4s	1.7s	10.5s	36.4s	353.3s
<i>Wedgelet-Modell</i> 0° to 180° by 10°	< 0.01s	< 0.01s	0.16s	1.2s	13.5s	129.4s
<i>Wedgelet-Modell</i> 0° to 180° by 45°	< 0.01s	< 0.01s	0.05s	0.4s	7.9s	113.6s
<i>Wedgelet-Modell</i> 0° to 180° by 90°	< 0.01s	< 0.01s	0.02s	0.5s	8.6s	107s
<i>Wedgelet-Modell</i> ohne Winkel	< 0.01s	< 0.01s	0.02s	0.8s	2.6s	100.1s

Die Berechnungen mit *BeamLab* für das Bild der Größe 512×512 wurden nach 12 Stunden abgebrochen.

Nach der theoretischen Herleitung der Komplexität des Algorithmus ($O(|S|)$), müssten sich die Werte in den Spalten für doppelt so grosse Bilder um den Faktor 4 unterscheiden. Die Tabelle zeigt, dass die Allokation der grösseren aufsummierten Matrizen auch entscheidend in die Laufzeit miteinflusst.

Als nächstes folgt eine Graphik, in der die Laufzeiten der verschiedenen Berechnungen für die unterschiedlichen Bildgrößen aufgetragen wurden.

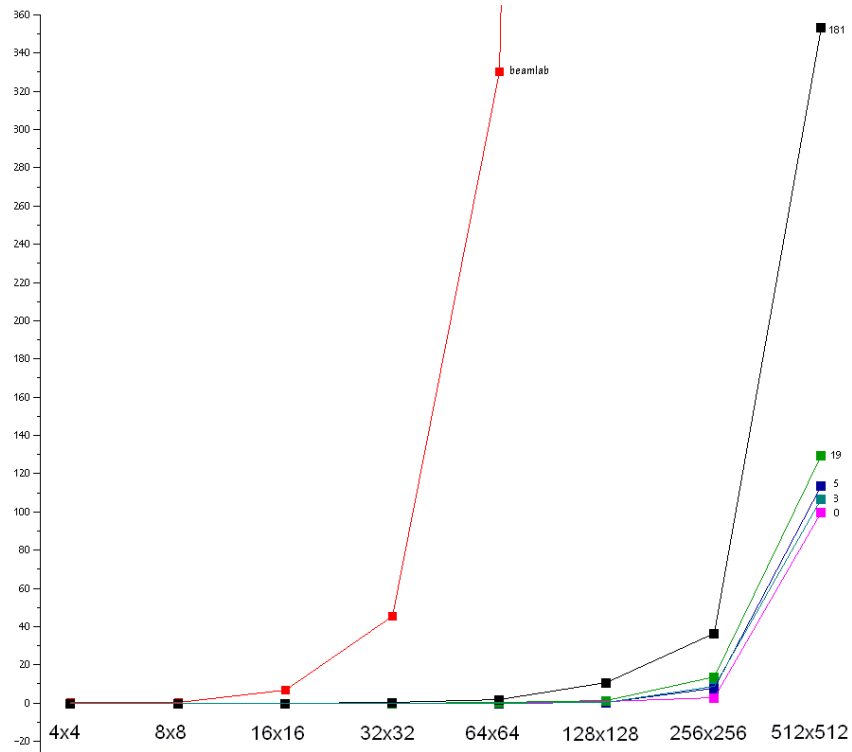


Abb. 42: Laufzeiten

6.3 Skalierung des Parameters γ

Neben der Wahl eines zur jeweiligen Problemstellung geeigneten Parameters γ (s. Abschnitt 6.4), können an die Form des zu minimierenden Funktionals noch einige Anforderungen gestellt werden. Für die Vergleichbarkeit der Anwendung auf Datensätze verschiedener Auflösung in Größe oder Graustufenskala sollten Minimierer des Funktionals bis auf Skalierung möglichst ähnlich sein. Tatsächlich lassen sich diese Anforderungen durch Größen- und Graustufen- abhängige Skalierung des Parameters γ selbst schon erreichen.

Sei die Indexmenge $S = \{1, \dots, W\} \times \{1, \dots, H\}$ gegeben. Zu Daten $y \in \mathbb{R}^S$ über S wird nun die um $w \times h$ ($w, h \in \mathbb{N}$) skalierte Version $y' \in \mathbb{R}^{S'}$ über $S' = \{1, \dots, w \cdot W\} \times \{1, \dots, h \cdot H\}$ betrachtet, d.h.

$$y'_{wi+m, hj+n} = y_{i,j}$$

für alle $i, j \in S$ und $0 \leq m < w$ und $0 \leq n < h$.

Sei $\mathbf{P} = \{(I_1, \mu_1) \dots, (I_n, \mu_n)\}$ ($n \in \mathbb{N}$) eine Segmentation über S und $\mathbf{P}' = \{(I'_1, \mu_1), \dots, (I'_n, \mu_n)\}$ die um $w \times h$ skalierte Version von \mathbf{P} über S' , d.h. es gelte

$$I'_i = \{(wk + u, hl + v) : (k, l) \in I_i, 0 \leq u < w, 0 \leq v < h\}$$

für alle $1 \leq i \leq n$.

Dann gilt

$$\begin{aligned} d(\mathbf{P}'|y') &= \sum_{i=1}^n \sum_{(s,t) \in I'_i} (y'_{s,t} - \mu_i)^2 \\ &= \sum_{i=1}^n \sum_{(s,t) \in I_i} \sum_{\substack{0 \leq u < w \\ 0 \leq v < h}} (y_{ws+u, ht+v} - \mu_i)^2 \\ &= \sum_{i=1}^n \sum_{(s,t) \in I_i} wh \cdot (y_{s,t} - \mu_i)^2 \\ &= wh \cdot d(\mathbf{P}|y). \end{aligned}$$


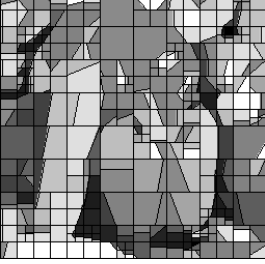
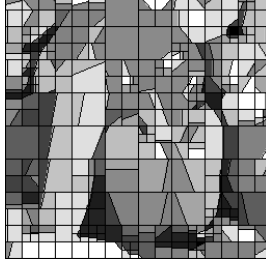

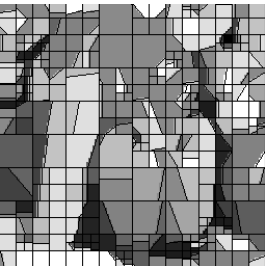
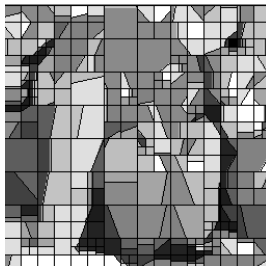

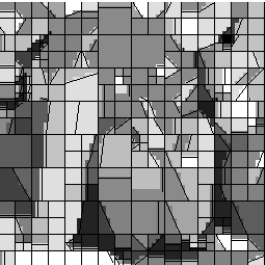
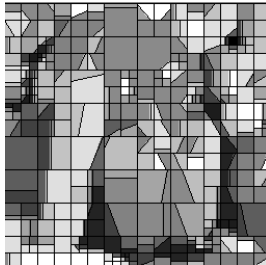

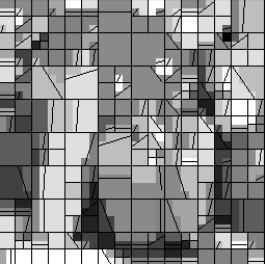
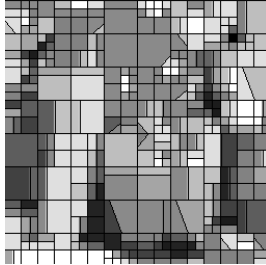
Damit gilt für das Funktional h_γ

$$h_{wh \cdot \gamma}(\mathbf{P}'|y') = wh \cdot g_\gamma(P) + wh \cdot d(\mathbf{P}|y) = wh \cdot h_\gamma(\mathbf{P}|y).$$

Das bedeutet, γ muss durch $|S| \cdot \gamma$ ersetzt werden, wenn die Minimierung mit der Skalierung der Daten die skalierte Partition liefern soll.

Die folgende Tabelle zeigt Ergebnisse der Minimierung von $h_{|S| \cdot \gamma}$ mit $\gamma = 0.5$. Dazu wurde das Bild *Peppers* der Größe 256×256 mit 256 Graustufen auf die Größen 128×128 , 64×64 und 32×32 verkleinert. Auf die verkleinerten Varianten wurde eine Segmentierung mit 19 Winkeln angewendet, zusätzlich wurden die Bilder wieder gemäß obigem Verfahren auf die ursprüngliche Größe 256×256 gebracht und auch segmentiert. Die Anzahl der Komponenten der Partition bleibt grob gleich.

Das betrachtete Funktional ist $h_{\gamma'}$ mit $\gamma' := |S| \cdot \gamma$.

 <p>Original (256x256)</p>	 <p>Segmentierung ($\gamma = 0.5$) $P = 459$</p>	 <p>Segmentierung ($\gamma = 0.5$), $P = 459$</p>
 <p>Original (128x128)</p>	 <p>Segmentierung ($\gamma = 0.5$) $P = 456$</p>	 <p>$128 \times 128 \rightarrow 256 \times 256$, $P = 433$</p>
 <p>Original (64x64)</p>	 <p>Segmentierung ($\gamma = 0.5$) $P = 406$</p>	 <p>$64 \times 64 \rightarrow 256 \times 256$, $P = 425$</p>
 <p>Original (32x32)</p>	 <p>Segmentierung ($\gamma = 0.5$) $P = 331$</p>	 <p>$32 \times 32 \rightarrow 256 \times 256$, $P = 376$</p>

Für eine Skalierung der Graustufen $y \mapsto r \cdot y$, $r \in \mathbb{R}$ erhält man

$$\begin{aligned}
 d(\mathbf{P}|r \cdot y) &= \sum_{i=1}^n \sum_{s \in I_i} ((r \cdot y)_s - \mu_I(r \cdot y))^2 \\
 &= \sum_{i=1}^n \sum_{s \in I_i} (ry_s - r\mu_I(y))^2 \\
 &= \sum_{i=1}^n \sum_{s \in I_i} r^2 (y_s - \mu_I(y))^2 \\
 &= r^2 \cdot d(\mathbf{P}|y).
 \end{aligned}$$

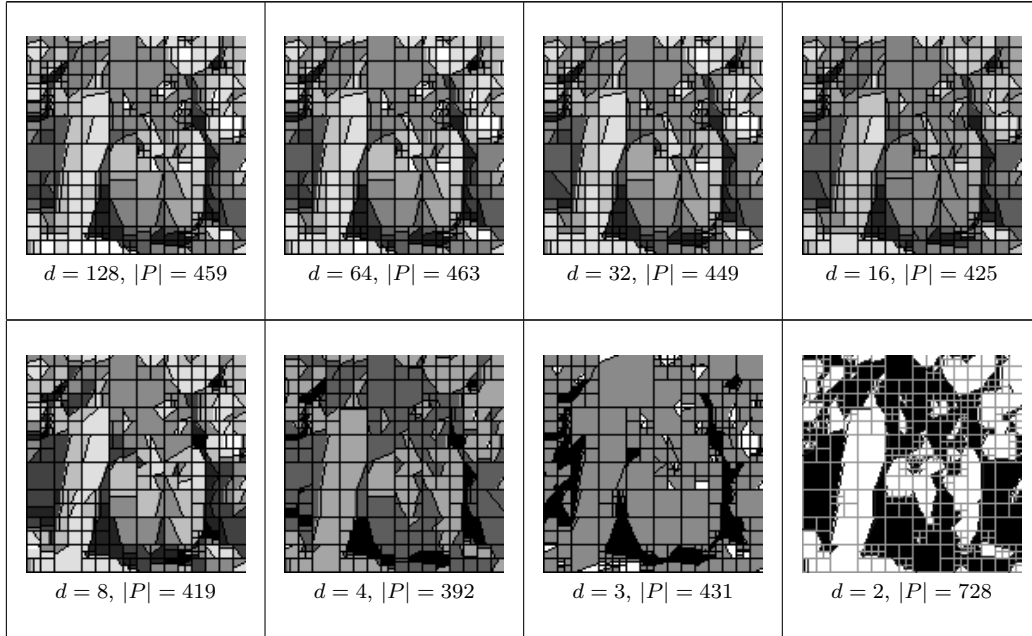
Damit gilt für das Funktional h_γ

$$h_{r^2 \cdot \gamma}(\mathbf{P}|r \cdot y) = r^2 \cdot g_\gamma(P) + r^2 \cdot d(\mathbf{P}|y) = r^2 \cdot h_\gamma(\mathbf{P}|y).$$

Das bedeutet, γ muss durch $d^2 \cdot \gamma$ ($d \in \mathbb{N}$, d ist die Anzahl der Graustufen) ersetzt werden, wenn die Minimierung mit der Graustufenskalierung das entsprechend skalierte Ergebnis liefern soll.

Die folgende Tabelle zeigt Ergebnisse der Minimierung mit 19 Winkeln zu verschiedenen Graustufenanzahlen.

Das betrachtete Funktional ist $h_{\gamma'}$ mit $\gamma' := |S| \cdot \frac{d^2}{256^2} \cdot \gamma$.



Insgesamt kann in Anwendungen also das Funktional $h_{d^2 \cdot |S| \cdot \gamma}$ betrachtet werden, um bei gleichem Parameter γ für verschiedene Auflösungen und verschiedene Graustufen-Quantisierungen ähnliche Ergebnisse zu erzielen.

6.4 Wahl des Parameters γ

Die Segmentierung des Bildes wurde für einen festen reellen Bestrafungsparameter $\gamma > 0$ durchgeführt, der die einander widerstrebenden Forderungen nach Datentreue und Regularität gegeneinander bewertet. Da die so entstehende Rekonstruktion entscheidend von diesem Parameter abhängt, stellt sich die Frage, wie - angepasst an die entsprechende Problemstellung - der Parameter γ möglichst optimal gewählt werden kann. Dabei wird ein objektives Verfahren favorisiert, das unabhängig vom jeweiligen Betrachter, γ alleine aus den Daten und der entsprechenden Fragestellung bestimmt. Insbesondere wenn gleichzeitig viele verschiedene Bilder nach denselben Gesichtspunkten untersucht werden sollen (z. B. bei eindimensionalen Bildern (Zeitreihen) in der Microarray-Analys), wird ein solch automatisierbares Verfahren benötigt

(für bis zu 20000 Zeitreihen ist es nicht möglich alle zu inspizieren und dann das beste Ergebnis herauszusuchen).

Die Möglichkeit, zunächst zu allen möglichen Parametern die Rekonstruktion zu berechnen und dann zu entscheiden, welcher Parameter die beste Lösung liefert, kann ebenfalls nur in Ausnahmefällen genutzt werden (s. A. KEMPE (2004)).

Für den eindimensionalen Fall des Potts-Funktionalen wurden in A. KEMPE (2004) die Eigenschaften verschiedenster Methoden der Parameterwahl diskutiert. Es zeigt sich hierbei, dass es nur wenige allgemeingültige Bedingungen gibt, die eine objektive Parameterwahl erfüllen sollten (Äquivalenz). Verschiedene Parameterwahlen führen genauso zu unterschiedlichen Problemlösungen, wie die Festlegung anderer Penalierungs- oder Datentreueparameter. D. DONOHO (1999) schlägt vor, anstelle die Anzahl der Komponenten einer Partition unabhängig von ihrer Größe zu bestrafen dass für eine dyadische Partition die Größe der dyadischen Rechtecke in der Bestrafung mitspielt und sehr kleine Rechtecke stärker bestraft werden als größere.

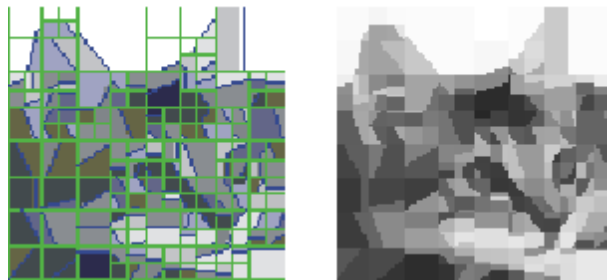
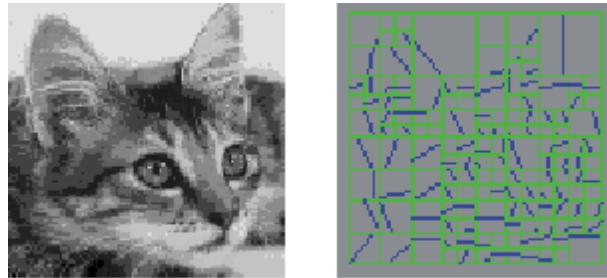
Einen weiteren Ansatz bietet D. DONOHO (1999) in *Theorem 7.1*, wo dem Bestrafungsparameter Folgendes zugewiesen wird:

$$\gamma = (\xi \cdot \sigma \cdot (1 + \sqrt{2 \log |\Delta|})),$$

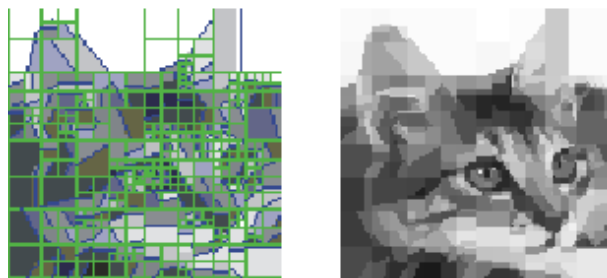
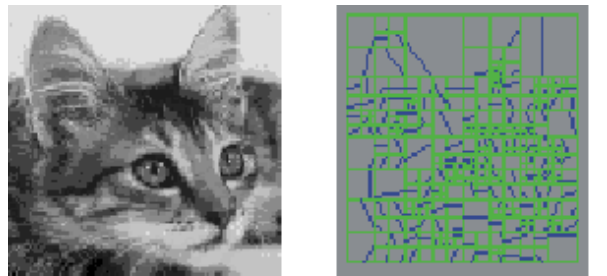
wobei $\xi > 8$, σ die Standardabweichung des Rauschens und $|\Delta|$ die Anzahl der Wedges im Minimierungsprozess ist. D. DONOHO (1999) gibt Abschätzungen des Approximationsfehlers relativ zu dem eines optimalen Schätzers. Eine sorgfältige Auswertung der Auswirkungen verschiedener Parameterwahlen gehört somit zum Kern des Problems genauso wie die Auswahl und die Untersuchung des Algorithmus.

Eine detaillierte Untersuchung zu dem beschriebenen Beispiel steht aber noch aus.

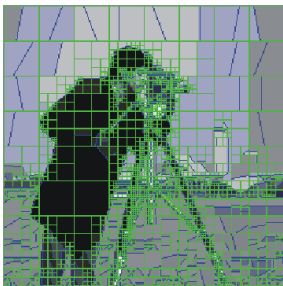
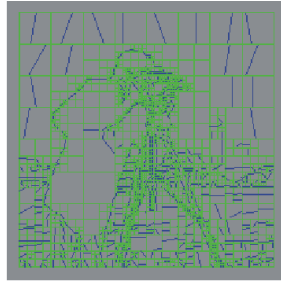
6.5 Beispiele



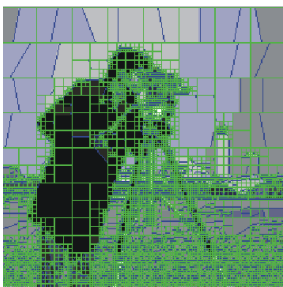
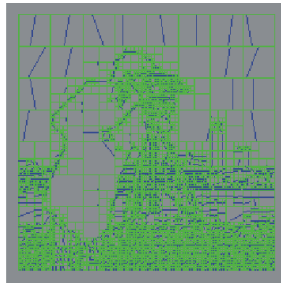
Cat 100×100 , Rechtecke, $\gamma = 1$



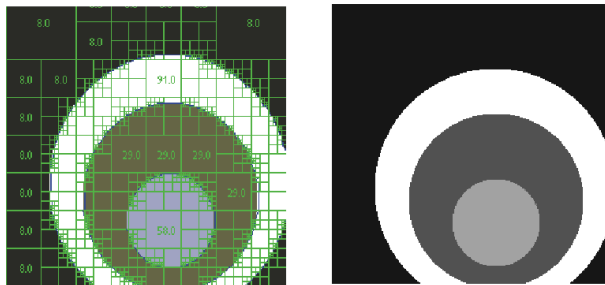
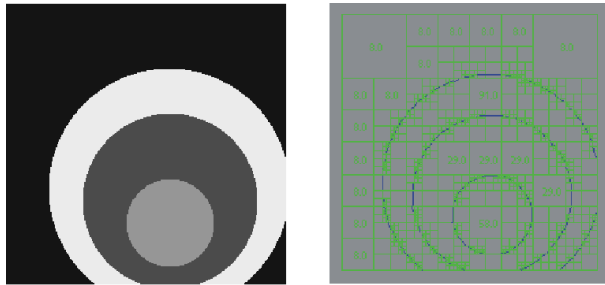
Cat 100×100 , Kantenlänge, $\gamma = 1$



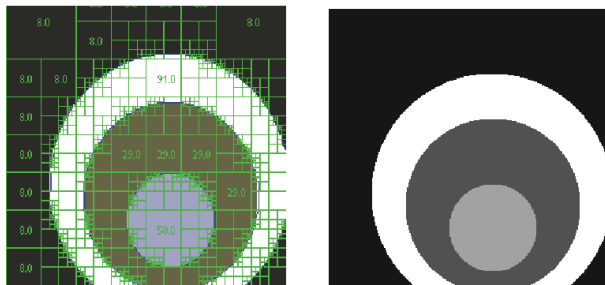
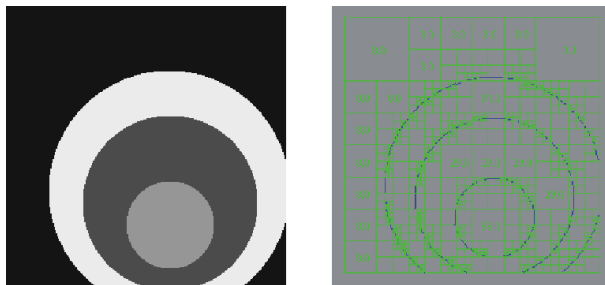
Camera 256×256 , Rechtecke, $\gamma = 0,2$



Camera 256×256 , Kantenlänge, $\gamma = 0,2$



Kreise 256 × 256, Rechtecke, $\gamma = 0,9$



Kreise 256 × 256, Kantenlänge, $\gamma = 0,9$

Literatur

- [1] <http://linux.isoe.ch/algorithm/iteration/bresenham/>. 8. 6. 2004.
- [2] <http://www-stat.stanford.edu/beamlab/>. 8. 6. 2004.
- [3] L. Demaret, N. Dyn und A. Iske. Image Compression by Linear Splines over Adaptive Triangulations. 2004.
- [4] D. Donoho. Wedgelets: Nearly-Minimax Estimation of Edges. *Annals of Stat.*, 27:859–897, 1999.
- [5] F. Friedrich. ANTS_{InFields}: A Software Package for Random Field Models and Imaging. Gsf-bericht, Institut für Biomathematik und Biometrie, GSF Forschungszentrum für Umwelt und Gesundheit, München-Neuherberg, 2002.
- [6] F. Friedrich. ANTS_{InFields}: A Software Package for Random Field Models and Imaging. <http://www.antsinfields.de>, 2003.
- [7] J. Hartigan. Estimation of a Convex Density Contour in Two Dimensions. *J. of the American Statistical Association*, 82(397):267–270, 1987.
- [8] M. Hirsch. *Differential Topology*. Springer Verlag, Berlin, Heidelberg, New York, 1976.
- [9] A. Kempe. *Statistical Analysis of the Potts Model and Applications in Medical Imaging*. PhD thesis, Institut für Biomathematik und Biometrie, GSF Forschungszentrum für Umwelt und Gesundheit, München, 2004.
- [10] J. Koplowitz, M. Lindenbaum und A. Bruckstein. The number of digital straight lines on an $n \times n$ grid. *IEEE Trans. Inform. Theory*, 36(1):192–197, 1990.
- [11] J. Romberg, M. Wakin, H. Choi und R. Baraniuk. *Geometric Tools for Image Compression*. Asilomar Conference on Signals, Systems and Computers, 2002a.

- [12] J. Romberg, M. Wakin und R. Baraniuk. Multiscale Wedgelet Image Analysis: Fast Decompositions and Modeling. 2002b.
- [13] G. Sawitzki. Extensible Statistical Software: On a Voyage to Oberon. *J. Computational and Graphical Statist.*, 5(3):263–283, 1996.
- [14] B. v. Querenburg. *Mengentheoretische Topologie*. Springer Verlag, Berlin, Heidelberg, New York, 1976.
- [15] M. Wakin, J. Romberg, H. Choi und R. Baraniuk. Rate Distortion Optimized Image Compression using Wedgelets. 2002.
- [16] K. Wicker. Das Potts-Modell zum Entrauschen digitaler Mammographien. Projektbericht, Institut für Biomathematik und Biometrie, GSF Forschungszentrum für Umwelt und Gesundheit, München-Neuherberg, 2002.
- [17] G. Winkler. *Image Analysis, Random Fields and Dynamic Monte Carlo Methods*, volume 27 of *Applications of Mathematics*. Springer Verlag, Berlin, Heidelberg, New York, second edition edition, 2002.
- [18] G. Winkler und V. Liebscher. Smoothers for Discontinuous Signals. *J. Nonpar. Statist.*, 14(1-2):203–222, 2002.