UNIVERSITÄT DORTMUND

Fakultät für Elektrotechnik

Studienarbeit S4 - 2002

Automatisierte Inhaltserkennung
durch Auswertung statistischer Bildeigenschaften

Axel Siebert
Matr.-Nr. 75346

Lehrstuhl für Kommunikationstechnik
Prof. Dr.-Ing. Rüdiger Kays

Kurzfassung

Unterzieht man Bildinhalte einer digitalen Signalverarbeitung, so benötigt man für eine optimale Verarbeitung der unterschiedlichen Inhaltstypen, wie natürliche Bildinhalte, grafische Inhalte oder Textelemente, jeweils unterschiedliche Algorithmen. Insbesondere bei einer örtlichen Abtastraten-Konversion weisen synthetisch generierte Bildinhalte andere Qualitätsansprüche auf als natürliche Bildinhalte.

In dieser Arbeit wurden verschiedene Bildinhalte auf ihre unterschiedlichen Eigenschaften hin untersucht, um sie hinsichtlich einer möglichen adaptiven örtlichen Abtastratenkonversion automatisiert klassifizieren zu können. Weiterhin wurde ein Segmentierungsalgorithmus implementiert, um regional unterschiedliche Inhaltstypen in gemischten Bildinhalten zu detektieren.

Inhaltsverzeichnis

Kapitel 1

Einleitung

Die Zahl verschiedener Anwendungsgebiete für die Darstellung von Bildinhalten mittels verschiedenster Displays wächst stetig. Zunehmend liegen die Daten in digitaler Form vor und werden auf Displays mit Matrixaufbau wiedergegeben. Damit wachsen auch die Ansprüche an Verfahren zur örtlichen Abtastratenkonversion.

So soll beispielsweise eine Webseite auf einem PC-Bildschirm, einem Fernsehgerät, oder auch auf einem PDA betrachtet werden können, wozu eine Anpassung an die spezifischen Gegebenheiten des jeweiligen Darstellungsmediums notwendig ist.

Nun gibt es aber verschiedene Bildinhaltstypen, die für eine optimale Verarbeitung jeweils unterschiedliche Algorithmen erfordern. So haben natürliche Bildinhalte, wie z.B. Photos, ganz andere Qualitätsansprüche bei einer Abtastratenkonversion als synthetisch generierte wie z.B. Text, Tabellen und Grafiken wie Cliparts und Diagramme. Es wird also eine Inhaltserkennung benötigt, die anhand bestimmter Eigenschaften der verschiedenen Inhaltstypen eine automatische Klassifizierung vornehmen kann.

In vielen Anwendungen, beispielsweise der Darstellung von Dokumenten und Webseiten sowie grafischen Benutzeroberflächen, vermischen sich jedoch häufig Bildinhalte unterschiedlichen Typs, weshalb auch eine Segmentierung hinsichtlich regional unterschiedlicher Verarbeitung vorgenommen werden muß.

Im Folgenden wird zunächst eine Übersicht über bestehende Skalierungsalgorithmen gegeben (Kap. 2) sowie in verschiedenen Gebieten bereits angewandte Klassifizierungen von Bildinhalten und deren Entscheidungskriterien erläutert (Kap. 3). Anhand eines Testdatensatzes von Bildern wird daraufhin eine Auswahl dieser Kriterien ausgewertet und ein darauf aufbauendes Verfahren zur regionalen Erkennung verschiedener Inhaltstypen vorgestellt und implementiert (Kap. 4). Abschließend werden die Ergebnisse zusammengefaßt und ein Ausblick auf mögliche Verbesserungen und Erweiterungen des Verfahrens gegeben (Kap. 5).

Kapitel 2

Anforderungen an Bildskalierungsalgorithmen

Jede Auflösungsreduktion ist immer mit einem gewissen Informationsverlust verbunden. Aufgabe verschiedener Bildskalierungsalgorithmen ist nun, diesen je nach Inhalt so zu gestalten, daß die für den Betrachter wichtigsten Informationen erhalten werden und ein optimaler visueller Gesamteindruck erreicht wird.

Eine Aufösungserhöhung ist weniger kritisch, da sie nicht mit einem Informationsverlust verbunden ist, aber auch hier gilt es, durch Auswahl eines entsprechenden Skalierungsalgorithmus den besten Gesamteindruck zu erreichen.

Nimmt man eine Grobeinteilung in Inhaltstypen mit grundsätzlichen anderen Eigenschaften und Anforderungen vor, so ergeben sich die Haupttypen

- Natürliche Bilder und Photos
- Synthetisch generierte Inhalte wie Texte, Strichgrafiken und Diagramme

Natürliche Bilder weisen ein zu hohen Frequenzen stark abfallendes Spektrum sowie einen sehr geringen Anteil an Aliaskomponenten auf, ein geeignetes Skalierungsverfahren ist beispielsweise die *Bilineare Interpolation*.

Synthetisch generierte Inhalte weisen dagegen einen großen Anteil hoher Frequenzanteile im Spektrum auf, bedingt durch scharfe Kanten und hohe Kontraste. Bei der Generierung wird zur Erreichung größerer Schärfe und Lesbarkeit auf eine Vorfilterung verzichtet, beispielsweise beim Zeichnen von Linien oder Buchstaben. Dadurch entstehen deutliche Aliasanteile, die bei der Skalierung zu einer Verfälschung führen. Um Unschärfe zu vermeiden, wird oft die *Nearest-Neighbour-Interpolation* angewendet, die aber andere Nachteile mit sich bringt.

Eine lineare Auflösungsreduktion läßt sich allgemein als Anti-Alias-Vorfilterung mit anschließender Dezimation der Abtastwerte um den entsprechenden Faktor beschreiben. Im Idealfall würde die Vorfilterung alle Anteile, die bei der Dezimation zu einem Alias führen würden, eliminieren und das übrige Nutzspektrum vollständig ausnutzen. Entsprechende Filter hätten aber eine unendlich lange Impulsantwort und sind somit nicht realisierbar, außerdem führt dieses Verfahren bei Signalen, die schon durch die Generierung mit Alias behaftet sind, nicht zu optimalen Ergebnissen. Daher werden bislang meistens die bereits genannten und im Folgenden näher erläuterten Verfahren angewendet.

2.1 Nearest-Neighbour-Interpolation

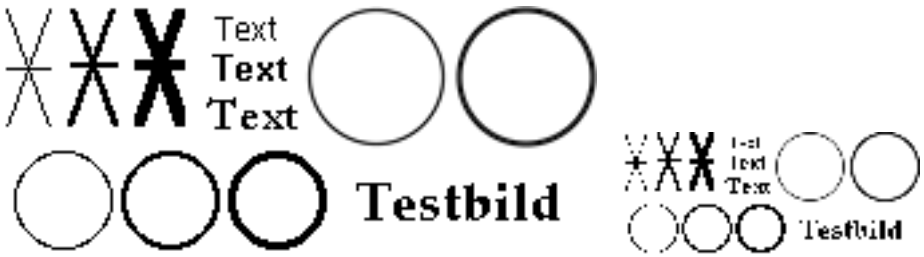
Diese Variante ist sehr weit verbreitet, die Einfachheit der Implementierung (siehe D.3) stellt den größten Vorteil des Verfahrens dar. Dabei wird einem Bildpunkt immer der Wert zugewiesen, den der im Ursprungsbild geometrisch nächstliegende Punkt besitzt. Die Abtastwerte werden also durch eine rechteckförmige Impulsantwort interpoliert und die resultierende Funktion neu abgetastet. Reduziert auf eine Dimension läßt sich die Impulsantwort folgendermaßen beschreiben [Ben92]:

$$h(x) = \begin{cases} 1 & |x| \leq \frac{T}{2} \\ 0 & |x| > \frac{T}{2} \end{cases}$$

Ermittelt man mittels der Fourier-Transformation die Auswirkungen dieser Impulsantwort im Spektrum, so erhält man eine si-Funktion:

$$H(\omega) = \int_{-\infty}^{\infty} h(x) e^{-j\omega x} dx = \int_{-\frac{T}{2}}^{\frac{T}{2}} e^{-j\omega x} dx = T \,\mathrm{si}\left(\frac{\omega T}{2}\right)$$

Dies ist der größte Nachteil des Verfahrens. Durch die si-Funktion im Spektrum verbleiben wesentliche Frequenzanteile oberhalb der neuen Grenzfrequenz im Signal, die bei der Abtastung zu Alias-Anteilen führen (**Abb. 2.2**). Außerdem werden hochfrequente Anteile im gewünschten Bereich des Spektrums abgeschwächt, wenn es sich nicht um eine Verkleinerung mit ganzzahligem Faktor handelt. In dem Fall gleicht nämlich der Alias die Abschwächung aus, im Ortsbereich findet dann lediglich eine Dezimation ohne jede Filterung statt.



(a) Testbild (b) Nearest Neighbour

Abbildung 2.1: Auflösungsreduktion um den Faktor 2

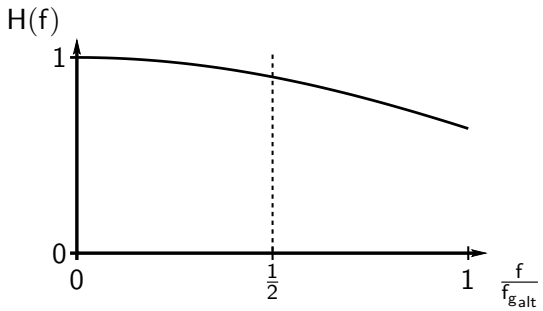


Abbildung 2.2: Auswirkung der Filterung im Spektrum

2.2 Bilineare Interpolation

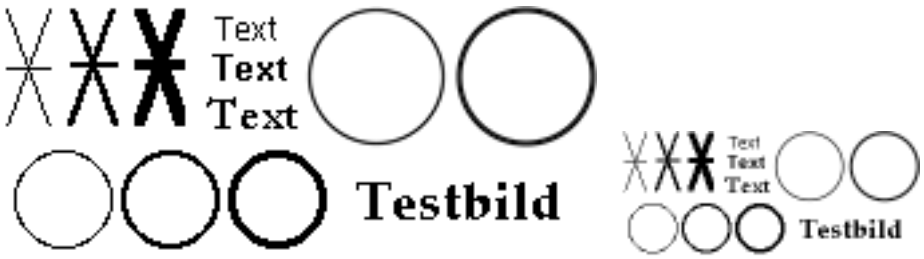
Bei diesem Verfahren erhält der Bildpunkt im Ausgangsbild bei einer Vergrößerung den Wert, der sich aus einer linearen Gewichtung der benachbarten Ursprungsbildpunkte ergibt. Reduziert auf eine Dimension läßt sich dies durch eine dreieckförmige Impulsantwort beschreiben [Ben92]:

$$h(x) = \begin{cases} 1 - \frac{|x|}{T} & |x| \leq T \\ 0 & |x| > T \end{cases}$$

Da dies einer Faltung der Nearest-Neighbour-Impulsantwort mit sich selbst darstellt, kann das Spektrum durch Multiplikation mit sich selbst ermittelt werden, also ist

$$H(\omega) = T\,\mathrm{si}^2\left(\frac{\omega T}{2}\right)$$

Die hochfrequenten Anteile, die bei der erneuten Abtastung zu Alias-Anteilen führen, werden deutlich stärker abgeschwächt. Leider gilt dies auch für das nutzbare Spektrum, was zu Unschärfe führt (**Abb. 2.4**). Beim Verkleinern verwenden die meisten Grafikprogramme[1] aus Implementationsgründen das *Averaging*, bei dem aus mehreren Bildpunkten das arithmetische Mittel gebildet wird. Dies entspricht wie die Nearest-Neighbour-Interpolation einer rechteckförmigen Impulsantwort, die Breite enspricht aber hier dem Abtastintervall im Zielbereich.



(a) Testbild (b) Averaging

Abbildung 2.3: Auflösungsreduktion um den Faktor 2

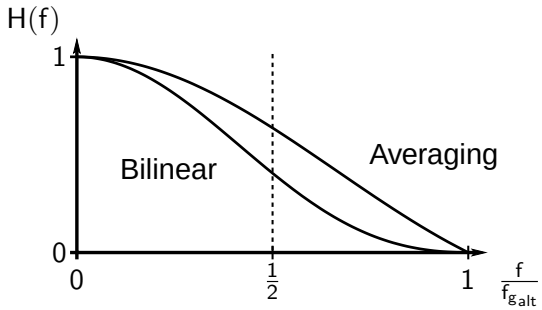


Abbildung 2.4: Auswirkung der Filterung im Spektrum

[1] z.B. Adobe Photoshop, Image Alchemy

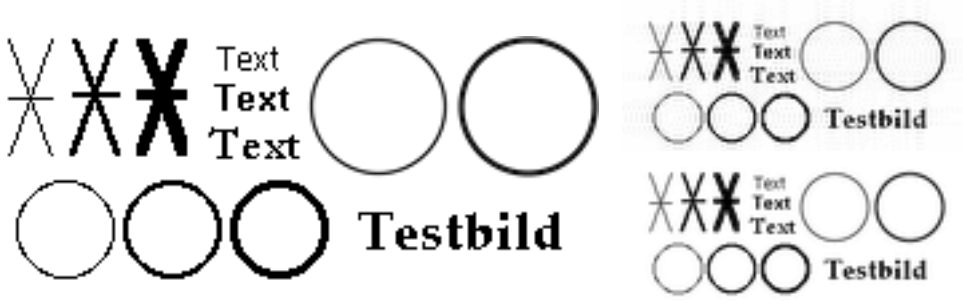
2.3 Filterung durch idealen Tiefpass

Um alle Anteile oberhalb der neuen halben Abtastfrequenz zu entfernen, ohne das nutzbare Spektrum zu beeinflussen, ist ein idealer Tiefpass mit Rechteckcharakteristik im Frequenzbereich nötig. Die entsprechende Impulsantwort im Ortsbereich ist allerdings eine si-Funktion, die aufgrund der unendlichen Ausdehnung nicht realisierbar ist und begrenzt werden muß. Außerdem ist die Rechteck-Funktion an den Flanken nicht stetig, so daß sich aufgrund des *Gibbsschen Phänomens* [SOS] auch bei der Realisierung als Faltung mit sehr großer Ausdehnung immer Überschwinger an den Flanken ergeben, die sich im Ortsbereich als *Ringing* äußern (**Abb. 2.5(b)**). Um dieses zu minimieren und eine Begrenzung der Ausdehnung zu erreichen, wird die Impulsantwort mit einer Fensterfunktion gewichtet. Häufige Verwendung findet dabei die Hann-Funktion h_{Hann}, in der Bildverarbeitung wird aber auch das sehr ähnliche Lanczos-Fenster $h_{Lanczos}$ verwendet [Theu99]:

$$h_{Hann}(x) = \begin{cases} \frac{1}{2}(1+cos(\pi\frac{x}{\tau})) & |x| < \tau \\ 0 & sonst \end{cases}$$

$$h_{Lanczos}(x) = \begin{cases} \mathrm{si}(\pi\frac{x}{\tau}) & |x| < \tau \\ 0 & sonst \end{cases}$$

Als Parameter τ wird in beiden Fällen üblicherweise 3 gewählt.



(a) Testbild

(b) Tiefpass ohne und mit Hann-Fenster

Abbildung 2.5: Auflösungsreduktion um den Faktor 2

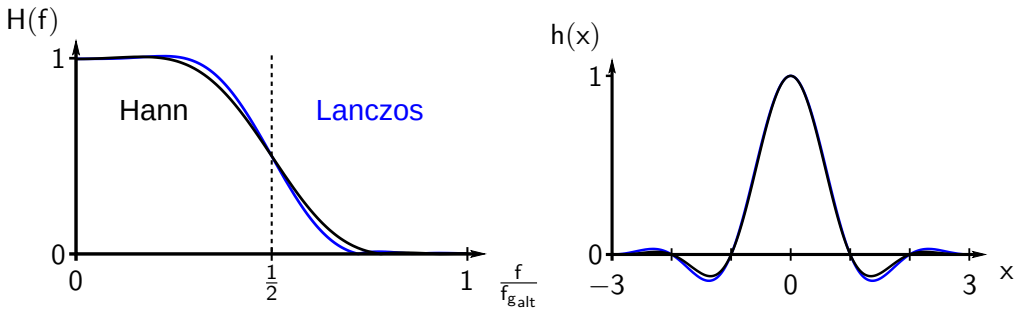


Abbildung 2.6: Auswirkung der Filterung im Spektrum und Impulsantwort

Es existieren weitere Verfahren, etwa die Splines-Interpolation, die den aliasfreien Idealfall annähern, das Problem der Unschärfe bei aliasbehafteten Ausgangsbildern jedoch nicht lösen.

Kapitel 3

Bestehende Klassifizierungen

Eine Klassifizierung von Bildinhalten wird bereits in verschiedenen Bereichen der Bildverarbeitung angewendet. In diesem Kapitel werden mehrere Anwendungsfälle und die dabei eingesetzten Verfahren erläutert, um sie hinsichtlich der Verwendbarkeit für eine inhaltsadaptive Skalierung untersuchen zu können. Zunächst wird die unterschiedliche Behandlung verschiedener Inhaltstypen bei der Erstellung von Webseiten betrachtet (3.1). Während hier die Objekte bereits getrennt vorliegen, müssen sie bei der automatischen Auswertung von Dokumenten zunächst erkannt und entsprechend separiert werden. Daher wird im Folgenden gezeigt, wie eine solche Erkennung bei der automatischen Schrifterkennung (3.2) sowie bei der Bildverarbeitung im Internet (3.3), z.B. zur funktionalen Erweiterung einer Suchmaschine für Bilder, vorgenommen wird. Weiterhin wird auf eine Wavelet-basierte Inhalteseparierung eingegangen (3.4), die für die Optimierung von Druckprozessen und Kompressionsverfahren vorgesehen ist. Abschließend wird noch ein kurzer Überblick über die Bildanalyse bei der Nachbildung des menschlichen Sehvermögens (3.5) gegeben.

3.1 Inhaltstypen in Hyper Text Markup Language

In *Hyper Text Markup Language* (HTML) verfaßte Webseiten sind ein gutes Beispiel für eine bereits bei der Erstellung durch das Format herbeigeführte Inhalteseparierung. Durch die beschränkte Bandbreite ist man besonders darauf angewiesen, für jeden Inhaltstyp eine günstige Darstellungsweise zu finden, also ein Format, welches jeweils den bestmöglichen Kompromiss zwischen Übermittlung der gewünschten Information, Begrenzung der Datenmenge und Flexibilität bei der Wiedergabe darstellt. Es ergibt sich folgende Einteilung:

- **Textelemente**
- **Einfarbige rechteckige Flächen** durch Elemente mit Hintergrundfarbe [W3col]
- **Horizontale und vertikale Linien** durch Umrandungen [W3lin]
- **Rechteckige Strukturen** durch Eingabeelemente [W3frm]
- **Bilder**

Es können sich auch Mischformen ergeben, wenn beispielsweise Textelemente mit einem Hintergrundbild hinterlegt werden.

Bilder werden gesondert im jeweils günstigsten Format im Hinblick auf Datenmenge und Qualitätsverlust gespeichert, wobei die Wahl desselben gewöhnlich anhand von Faustregeln getroffen wird. Eine solche ist

„Viele Farben, JPEG... Wenige Farben oder keine Farbverläufe, GIF. “ [WDG]

Auch detailliertere Anleitungen zur Wahl des Bildformates, die die Art des Bildes einbeziehen, sind verbreitet [LANL]:

- **JPEG:** Photographien, realistische Szenen, Bilder mit kontinuierlicher Farbeinteilung
- **GIF:** Strichgrafiken, Cartoons, Icons, Zeichen, generell palettenbasierte Bilder mit beschränkter Farbanzahl und scharfen Kanten

Diese Kriterien berücksichtigen genau die Stärken der beiden verwendeten Kompressionsverfahren:

Das JPEG-Format basiert auf einer Transformation in den Frequenzbereich und einer nachfolgenden Quantisierung. Dabei werden entsprechend den Eigenschaften des menschlichen Auges höhere Frequenzen gröber quantisiert, es geht also Information verloren [JPEG]. Dieser Verlust ist nur gering, wenn das Bild kaum Anteile hoher Frequenzen hat, was besonders bei Bildern mit oben genannten Eigenschaften gegeben ist.

Das GIF-Format hingegen ist verlustfrei, es wird lediglich Redundanz entfernt [GIF]. Dies ist allerdings nur bei sich zumindest teilweise exakt wiederholenden Strukturen möglich. Je geringer die Anzahl der verwendeten Farben und je größer der Anteil homogener Flächen ist, desto größer ist die Wahrscheinlichkeit solcher Wiederholungen.

Ein Vergleich mit den Vor- und Nachteilen der in Kapitel 2 erläuterten Skalierungsalgorithmen zeigt, daß die obigen Kriterien für die Wahl des Bildformates genau die Eigenschaften eines Bildes einbeziehen, die auch bei der Wahl eines Skalierungsalgorithmus beachtet werden müssen. Sie sind jedoch in der vorliegenden Form nicht für die Implementierung einer automatisierten Klassifizierung geeignet.

3.2 Inhaltstypen bei Optical Character Recognition

Ein Anwendungsgebiet, in dem die Separierung von Bildinhalten schon seit längerem eine wichtige Rolle spielt, ist die Texterkennung, genannt *Optical Character Recognition* (OCR). Dabei müssen Bilddaten, meistens erzeugt durch Scannen einer Vorlage, in Textdokumente umgewandelt werden. Da Teile der Vorlage eventuell keinen Text enthalten, sondern Photos oder Grafiken, müssen diese zunächst vom zu erkennenden Text separiert und dann entweder ignoriert oder im Ausgabedokument als Bilder eingebunden werden.

Existierende Verfahren zur Separierung lassen sich zum Großteil auf die Verfahren **Bottom-Up** [FlKa88] und **Top-Down** [WaSr89] zurückführen, die im Folgenden erläutert werden.

Bei den Bottom-Up-Verfahren werden zunächst *connected components* (**Abb. 3.1**), also Gruppen zusammenhängender Pixel bestimmt, welche dann regelbasiert zu größeren Blöcken zusammengefaßt werden. Diese Regeln setzen meist eine Kenntnis des vorliegenden Dokumenttyps voraus, um bestimmte geometrische Charakteristika und Beziehungen der Blöcke untereinander

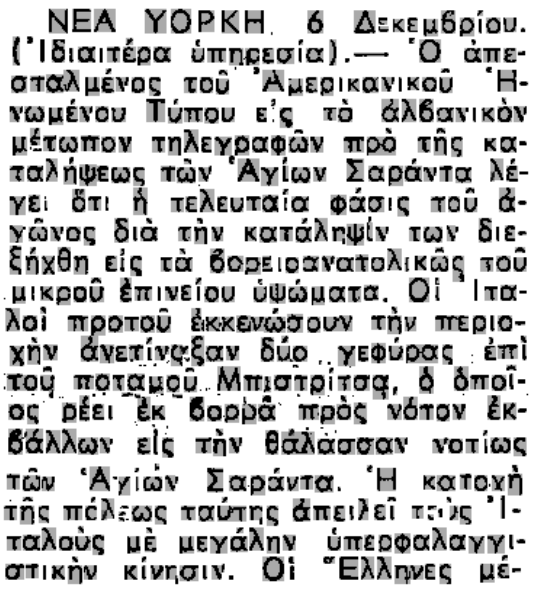

ΝΕΑ ΥΟΡΚΗ 6 Δεκεμβρίου.
('Ιδιαιτέρα ὑπηρεσία).— 'Ο ἀπε-
σταλμένος τοῦ 'Αμερικανικοῦ 'Η-
νωμένου Τύπου εἰς τὸ ἀλβανικὸν
μέτωπον τηλεγραφῶν πρὸ τῆς κα-
ταλήψεως τῶν 'Αγίων Σαράντα λέ-
γει ὅτι ἡ τελευταία φάσις τοῦ ἀ-
γῶνος διὰ τὴν κατάληψίν των διε-
ξήχθη εἰς τὰ βορειοανατολικῶς τοῦ
μικροῦ ἐπινείου ὑψώματα. Οἱ Ἰτα-
λοὶ προτοῦ ἐκκενώσουν τὴν περιο-
χὴν ἀνετίναξαν δύο γεφύρας ἐπὶ
τοῦ ποταμοῦ Μπιστρίτσα, ὁ ὁποῖ-
ος ρέει ἐκ βορρᾶ πρὸς νότον ἐκ-
βάλλων εἰς τὴν θάλασσαν νοτίως
τῶν 'Αγίων Σαράντα. 'Η κατοχὴ
τῆς πόλεως ταύτης ἀπειλεῖ τοὺς 'Ι-
ταλοὺς μὲ μεγάλην ὑπερφαλαγγι-
στικὴν κίνησιν. Οἱ "Ελληνες μέ-

Abbildung 3.1: Connected components (aus [HaHi01])

auszuwerten. So sind beispielsweise bestimmte wissenschaftliche Publikationen immer zweispaltig, während eine Zeitungsseite gewöhnlich völlig anders aufgeteilt ist.

Bei den Top-Down-Verfahren werden von der gesamten Vorlage *projection profiles* (**Abb. 3.2**) angelegt, also die Anzahl schwarzer Pixel in horizontaler und vertikaler Richtung an den Rand projiziert, und anhand dieser Profile horizontale und vertikale Schnitte durchgeführt. Die dabei entstehenden Blöcke werden dann unter Einbeziehung zuvor genannter Regeln ebenfalls zu größeren logischen Einheiten zusammengefügt.

Diese Verfahren sind jedoch für die in dieser Arbeit angestrebte allgemeine Inhalteseparierung nicht gut geeignet, da die untersuchten Dokumente eine Reihe von Voraussetzungen erfüllen müssen:

- Die Verfahren sind auf Schwarzweiß-Bilder beschränkt. Die gegebenenfalls notwendige Konvertierung kann Probleme aufwerfen, falls das Ursprungsbild eine hohe Farbenanzahl aufweist, stark gestört ist oder eine unregelmäßige Helligkeitsverteilung hat.
- Es wird ein sogenanntes Manhattan-Layout [WuMa97] vorausgesetzt, also eine Vorlage, die sich in ausschließlich rechteckige Blöcke einteilen läßt, weshalb die meisten Implementierungen noch einen Schritt zur Ermittlung und Korrektur einer eventuellen Schräglage vorschalten.
- Die Regeln zur Zusammenfassung kleiner Blöcke oder Komponenten zu größeren Einheiten setzen wie zuvor erläutert eine zumindest grobe Kenntnis des vorliegenden Dokumenttyps voraus.
- Eine Separierung, die lediglich Text von allgemeinen anderen Elementen unterscheidet, ist für die Realisierung einer inhaltsadaptiven Skalierung zu grob, da die anderen Elemente nicht weiter unterteilt werden, z.B. in Grafiken und Photos. Lediglich der bei den Bottom-Up-Verfahren verwendete Ansatz der Gruppierung kleiner Elemente zu verbundenen Komponenten könnte für eine objektbasierte Formatkonversion von Interesse sein, auf eine nähere Betrachtung wurde jedoch im Hinblick auf Umfang und Thema dieser Arbeit zugunsten von den in Kap. 3.3 und 3.4 betrachteten Verfahren verzichtet.

Abbildung 3.2: Segmentierung anhand von projection profiles (siehe D.4)

3.3 Inhaltserkennung für die Bildverarbeitung

Soll der Inhalt eines Bildes als Ganzes klassifiziert werden, um eine Bildverarbeitung zu optimieren, so bietet sich das Auswerten statistischer Bildeigenschaften an, welches bereits in unterschiedlichen Zusammenhängen erforscht wurde:

- Unterscheidung von Grafiken und Photos im Internet zur funktionalen Erweiterung einer Suchmaschine [AtSw97, SmCh97]
- Implementierung eines *Image Transcoding Proxy* zum Anpassen von Bildern im Internet an verschiedene Zielgeräte mit unterschiedlichen Verfahrensweisen für bestimmte Bildinhalte [SmLi98]

Im Folgenden werden die statistischen Bildeigenschaften näher erläutert, die bei obigen Verfahren als Entscheidungskriterien verwendet wurden. Die Größe des Bildes sei dabei MxN Pixel, die Farbwerte (r,g,b) der Pixel Integerwerte zwischen 0 und 255.

3.3.1 Durchschnittliche Sättigung

Die Sättigung ohne Helligkeitskompensation S_h ist die Differenz zwischen dem Maximum und dem Minimum aus (r,g,b). Es wird der Durchschnitt $\bar{S_h}$ für alle Pixel des Bildes gebildet, dieser ist bei Grafiken höher als bei Photos, da hochsaturierte Farben in Grafiken häufiger auftreten [SmLi98].

$$S_h = \max(r,g,b) - \min(r,g,b)$$
$$\bar{S_h} = \frac{1}{MN} \sum_{m,n} S_h[m,n]$$

3.3.2 Anteil schwarzer, weißer und grauer Pixel

Die Anteile A_x schwarzer, weißer und grauer Pixel ermöglichen eine Entscheidung zwischen Farb-, Graustufen- und Schwarz-Weiß-Bildern, ein hoher Weißanteil deutet zusätzlich auf Grafiken hin [SmCh97].

$$A_x = \frac{1}{MN} \sum_{m,n} \begin{cases} 1 & \text{Pixel } p[m,n] \text{ hat die Farbe } x \\ 0 & \text{sonst} \end{cases}$$

3.3.3 Anteil voll- und halbgesättigter Pixel

Die Anteile der Pixel mit voller und halber Sättigung S ermöglichen eine Unterscheidung von Photos und Grafiken, wobei letztere noch in komplexe und einfache Grafiken unterteilt werden können. Grafiken haben einen höheren Anteil A_v an vollgesättigten Pixeln, während besonders komplexe Grafiken sich durch einen hohen Anteil A_h halbgesättigter Pixel auszeichnen [SmCh97, AtSw97].

$$A_h = \frac{1}{MN} \sum_{m,n} \begin{cases} 1 & 1 > S[m,n] \geq \frac{1}{2} \\ 0 & \text{sonst} \end{cases}, \; A_v = \frac{1}{MN} \sum_{m,n} \begin{cases} 1 & S[m,n] = 1 \\ 0 & \text{sonst} \end{cases}$$

3.3.4 Anzahl der verwendeten Helligkeits-, Sättigungs- und Farbtonstufen

Die Anzahl der verwendeten Helligkeits-, Sättigungs- und Farbtonstufen wird ebenfalls zur Unterscheidung von Photos, einfachen und komplexen Grafiken verwendet. In einfachen Grafiken finden sich mit Abstand am wenigsten Abstufungen in allen drei Bereichen, komplexe Grafiken unterscheiden sich von Photos durch eine größere Anzahl Farbtöne, aber weniger Sättigungs- und Helligkeitsstufen [SmCh97].

3.3.5 Anzahl verwendeter Farben im quantisierten HSV-166-Farbraum

Die Anzahl verwendeter Farben im quantisierten HSV-166-Farbraum wird ebenfalls zur Unterscheidung von Photos, einfachen und komplexen Grafiken verwendet. Komplexe Grafiken weisen eine deutlich größere Anzahl als Photos auf, diese wiederum eine größere Anzahl als einfache Grafiken [SmCh97, AtSw97].
Zunächst wird eine Transformation T des RGB-Farbraums in den HSV-Farbraum vorgenommen (Hue, Saturation, Value = Farbton, Sättigung, Helligkeit), in dem sich bestimmte Eigenschaften der Wahrnehmung besser modellieren lassen [SmCh96].

$$v = \max(r,g,b),\ d = v - \min(r,g,b)$$

$$s = \frac{d}{v}$$

$$r' = \frac{v-r}{d},\ g' = \frac{v-g}{d},\ b' = \frac{v-b}{d}$$

$$h = \begin{cases} 5+b' & r = v \text{ und } g = m \\ 1-g' & r = v \text{ und } g \neq m \\ 1+r' & g = v \text{ und } b = m \\ 3-b' & g = v \text{ und } b \neq m \\ 3+g' & b = v \text{ und } r = m \\ 5-r' & \text{sonst} \end{cases}$$

Nach der Transformation wird eine Quantisierung vorgenommen. Da bei der Betrachtung verwendeter Farben der Farbton die wichtigste Komponente darstellt, erhält dieser mit 18 Stufen die feinste Quantisierung. Sättigung und Helligkeit werden jeweils auf 4 Stufen quantisiert. Der HSV-Farbraum ist zylindrisch (**Abb. 3.3**), bei einer Sättigung von null werden also keine Farbtöne unterschieden, sondern nur 4 Graustufen mit verschiedenen Helligkeiten. Insgesamt ergeben sich also $18 \cdot 3 \cdot 3 + 4 = 166$ HSV-Werte.

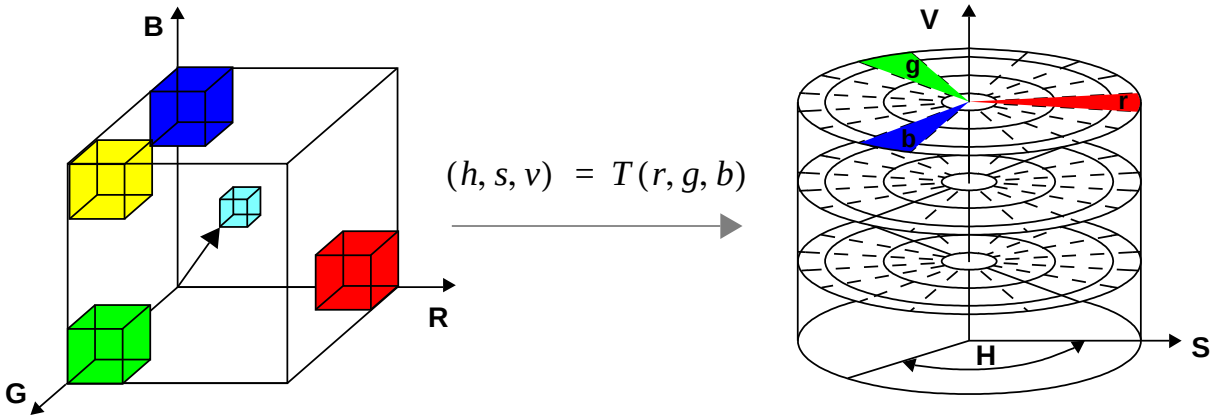


Abbildung 3.3: HSV-Raum

3.3.6 Varianz und Entropie der Intensität

Varianz und Entropie der Intensität dienen der Unterscheidung von Graustufen- und Schwarz-Weiß-Bildern. Letztere haben eine geringe Entropie und große Varianz, während dies für Graustufen-Bilder umgekehrt ist. Die Entropie kann auch zur Unterscheidung von Graustufen-Photos und -Grafiken herangezogen werden, bei letzteren ist der Wert geringer [SmLi98].

Bei der Berechnung der Intensität I wird die jeweilige Empfindlichkeit des menschlichen Auges für die drei Farbkomponenten Rot, Grün und Blau berücksichtigt:

$$I = 0.3x_r + 0.6x_g + 0.1x_b$$

Nach Bestimmung des Mittelwerts

$$\bar{I} = \frac{1}{MN} \sum_{m,n} I[m,n]$$

wird die Varianz σ_I^2 ermittelt:

$$\sigma_I^2 = \frac{1}{MN} \sum_{m,n} (I[m,n] - \bar{I})^2$$

Für die Berechnung der Entropie wird zunächst die Verteilung $p[k]$ der Intensitäten ermittelt:

$$p[k] = \frac{1}{MN} \sum_{m,n} \begin{cases} 1 & k = I[m,n] \\ 0 & \text{sonst} \end{cases}$$

Die Entropie E_I ist dann

$$E_I = -\sum_{k=0}^{255} p[k] \log_2 p[k]$$

3.3.7 Wechselhäufigkeit der Intensität

Die Wechselhäufigkeit der Intensität ermöglicht eine Unterscheidung von Grafiken und Photos bei Graustufen- und Schwarz-Weiß-Bildern. Bei Grafiken ist die Wechselhäufigkeit deutlich geringer [SmLi98].

Die Wechselhäufigkeit W_h in horizontaler Richtung ist definiert als

$$W_h = \frac{1}{MN} \sum_{m,n} \begin{cases} 1 & I[m-1,n] \neq I[m,n] \\ 0 & \text{sonst} \end{cases}$$

Auf gleiche Weise wird die Wechselhäufigkeit in vertikaler Richtung bestimmt, der Gesamtwert ist dann $W = \min(W_h, W_v)$.

Auch bei farbigen Bildern können Wechselhäufigkeit und Entropie zur Unterscheidung von Photos und Grafiken herangezogen werden, dann wird statt der Intensität der HSV-166-Farbraum verwendet.

3.4 Wavelet-basierte Inhalteseparierung

Zur Optimierung von Druckprozessen und Kompressionsverfahren oder der effizienten Datengewinnung aus Bilddatenbanken existiert ein Verfahren zur Segmentierung von Bildinhalten in die Klassen Photo, Grafik und Text, welches auf der Analyse der Koeffizienten-Verteilung in den hochfrequenten Bändern nach einer Wavelet-Transformation beruht [LiGr97, LiGr00].

3.4.1 Die Wavelet-Transformation

Die Wavelet-Transformation Ψ ist die Dekomposition einer Funktion in Basisfunktionen, welche aus einem Mutter-Wavelet ψ durch eine Skalierung s, also Streckung oder Stauchung, und Verschiebung τ erzeugt werden [RiVe91, Robi]. Im Folgenden wird eine eindimensionale, zeitabhängige Funktion $x(t)$ vorausgesetzt, durch separate Transformationen für jede Dimension ist aber eine Anwendung bei Bildern gleichermaßen möglich. Die kontinuierliche Wavelet-Transformation (CWT) ist gegeben durch:

$$CWT_x^{\Psi}(\tau,s) = \Psi_x^{\Psi}(\tau,s) = \frac{1}{\sqrt{|s|}} \int x(t) \psi^* \left(\frac{t-\tau}{s} \right) dt$$

Wird ψ als Bandpass-Filter aufgefaßt, so ergibt sich nach der Skalierung wieder ein Bandpass-Filter mit derselben relativen Bandbreite, das Verhältnis von zentraler Frequenz und Bandbreite ist also konstant. Eine Vergrößerung des Skalierfaktors s bedeutet eine Streckung des Wavelets mit der daraus folgenden kleineren zentralen Frequenz und entsprechend kleinerer Bandbreite des Bandpasses.

Ersetzt man den Begriff der Skalierung durch den Begriff der Frequenz, so wird die Verwandtschaft mit der Kurzzeit-Fouriertransformation (STFT) offensichtlich. Der wesentliche Unterschied ist jedoch, daß bei der STFT die von der Wahl der Fensterfunktion abhängige Zeit- und Frequenzauflösung über alle Frequenzen gleich ist, während bei einer Wavelet-Transformation für

höhere Frequenzen die Zeitauflösung besser wird, für niedrigere Frequenzen dagegen die Frequenzauflösung. Damit kann die menschliche Wahrnehmung von Signalen modelliert werden, beispielsweise entsprechen höhere Oktaven bei der akustischen Wahrnehmung jeweils einer Verdoppelung der Frequenz mit gleichzeitig abnehmender Frequenzauflösung.

Die Transformation liefert aufgrund beliebiger Werte für s und τ sehr redundante Informationen, man ist also daran interessiert, s und τ so zu diskretisieren, daß sich eine Zerlegung in eine minimale Anzahl von skalierten Wavelets ergibt. Dies führt bei der Wavelet-Transformation in diskreter Form (DWT), die zunehmend Anwendung in der Signalverarbeitung und -kodierung findet, zu einer Realisierung mittels digitaler Filter und einer rekursiven Frequenzband-Halbierung (**Abb. 3.4**). Das Signal wird mit einem Tiefpass und einem Hochpass in zwei Halbbänder aufgeteilt, die dann aufgrund der halbierten Grenzfrequenz um den Faktor 2 unterabgetastet werden können. Hochpass- und Tiefpass-Funktion mit der geraden Länge L hängen über folgende Modulation zusammen:

$$h(L-1-n) = (-1)^n g(n)$$

Diese Aufteilung kann für das untere Halbband solange wiederholt werden, bis nur noch ein Sample übrigbleibt, wobei die Unterabtastung jeweils der Skalierung des Wavelets entspricht.

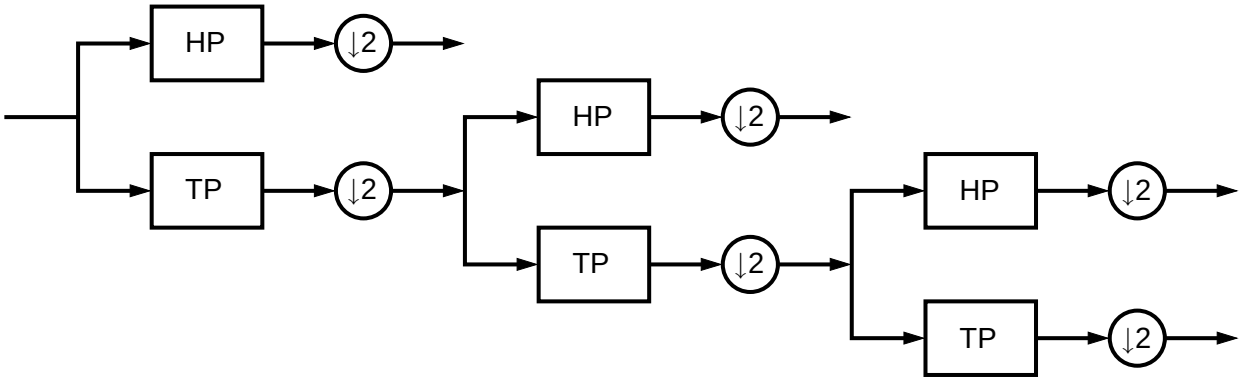


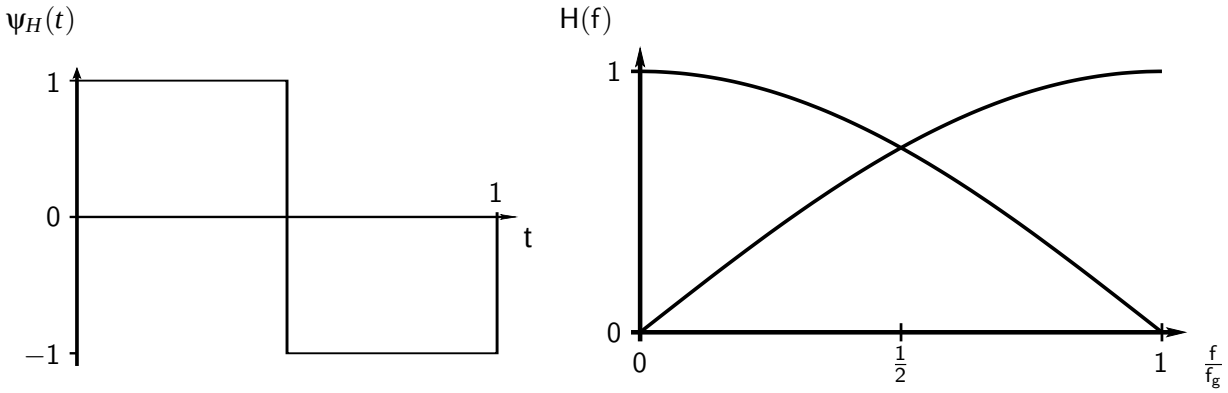
Abbildung 3.4: Halbband-Filterung

3.4.2 Das Verfahren

Es hat sich gezeigt, daß die Koeffizienten in den hochfrequenten Bändern bei Photos zu einer Laplace-Verteilung neigen [SmRo96]. Weiterhin läßt sich im Histogramm der Koeffizienten im Falle von Grafik und Text eine Konzentration bei einzelnen Werten feststellen, bei Photos nicht. Da Text gewöhnlich einfarbig ist, kann mithilfe der Anzahl der Helligkeitswerte zwischen Text und Grafik unterschieden werden.

Zunächst wird die Wavelet-Transformation der Helligkeitswerte der Bildpixel unter Verwendung des Haar-Wavelets ψ_H (**Abb. 3.5**) berechnet, da dieses aufgrund des extrem kurzen Filters die beste Ortsgenauigkeit ergibt, die für eine Einteilung in Blöcke wichtig ist, außerdem ist aus demselben Grund der Rechenaufwand am geringsten.

$$\psi_H(t) = \begin{cases} 1 & 0 \leq t \leq \frac{1}{2} \\ -1 & \frac{1}{2} < t \leq 1 \\ 0 & \text{sonst} \end{cases}$$



(a) Wavelet-Funktion (b) Tiefpass- und Hochpass-Spektrum

Abbildung 3.5: Haar-Wavelet

Es wird nur die erste Iteration der Transformation durchgeführt, da diese in den drei hochfrequenten Bändern bereits genügend Merkmale zur Unterscheidung liefert. Ein Transformationsschritt besteht aus jeweils einer Filterung in horizontaler und vertikaler Richtung, weshalb sich vier Bänder ergeben (**Abb. 3.6**).

Danach wird das Bild in quadratische Blöcke eingeteilt und diese jeweils anhand der nachfolgend beschriebenen zwei Kriterien klassifiziert. Kann ein Block keinem Inhaltstyp eindeutig zugeordnet werden, wird er in vier kleinere Blöcke unterteilt, welche dann jeweils einzeln untersucht werden.

LL LH

HL HH

Abbildung 3.6: Ergebnis der ersten Iteration

Grad der Übereinstimmung mit einer Laplace-Verteilung

Die LH-, HL- und HH-Koeffizienten des zu untersuchenden Blocks werden in einem Histogramm zusammengefaßt und dessen Grad der Übereinstimmung mit einer Laplace-Verteilung ermittelt. Diese ist gegeben durch

$$p(x) = \frac{\lambda}{2} e^{-\lambda|x|}$$

Der Parameter λ muß nun entsprechend der vorliegenden Verteilung gewählt werden, eine einfache Möglichkeit ist die Anpassung der Varianz. Dies entspricht zwar nicht der besten Näherung im Sinne einer möglichst geringen Abweichung, ist aber für das zu bestimmende Maß ausreichend [SmRo96] (**Abb. 3.7**). Für $p(x)$ mit $E\{p(x)\} = 0$ ist die Varianz

$$\begin{aligned}
\sigma^2 &= \int_{-\infty}^{\infty} x^2 p(x) dx \\
&= 2 \int_0^{\infty} x^2 \frac{\lambda}{2} e^{-\lambda x} dx \\
&= 2 \left(-e^{-\lambda x} \left(\frac{1}{2} x^2 + \frac{x}{\lambda} + \frac{1}{\lambda^2} \right) \right) \Bigg|_0^{\infty} \\
&= \frac{2}{\lambda^2}
\end{aligned}$$

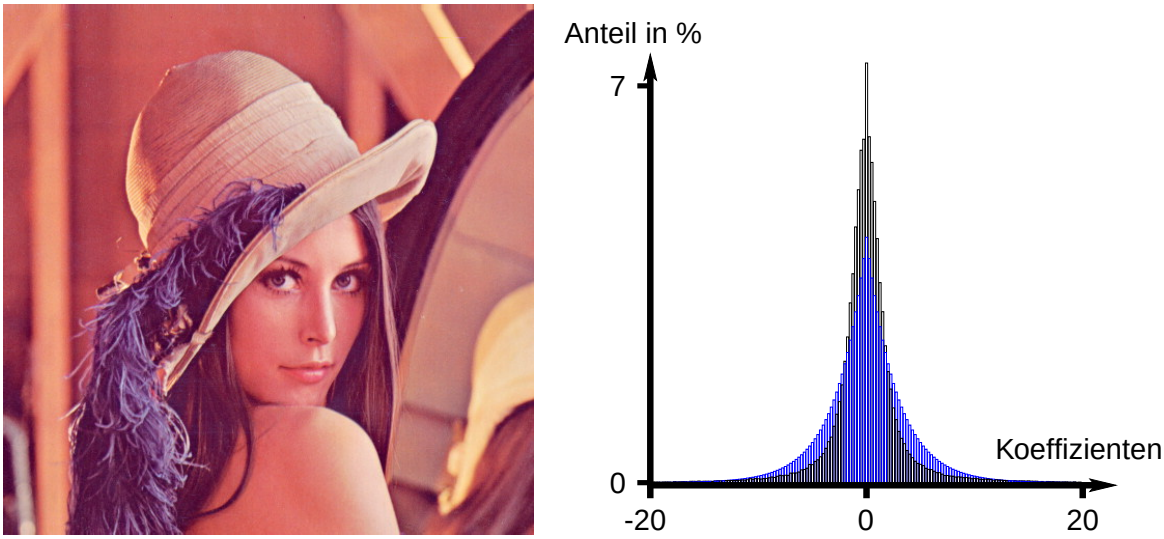
Also wird $\lambda = \sqrt{\frac{2}{\sigma^2}}$ gewählt. Die vorliegende Verteilung wird durch Auszählung der vorkommenden Koeffizienten ermittelt und verläuft somit sprunghaft, stellt also im Gegensatz zur angenommenen Laplace-Verteilung keine kontinuierliche Funktion dar. Um die Abweichung bestimmen zu können, muß daher der Wertebereich der Koeffizienten in K diskrete Bereiche $\alpha_k < x < \alpha_{k+1}$ aufgeteilt und jeweils die Gesamthäufigkeit der Koeffizienten in den Bereichen betrachtet werden. K muß groß genug sein, damit die Laplace-Verteilung ausreichend genau angenähert wird, allerdings auch klein genug, um die Sprünge in der ermittelten Verteilung auszugleichen. f_k sei der Anteil der Koeffizienten im Bereich k der vorliegenden Verteilung, F_k der entsprechende Anteil bei der gewählten Laplace-Verteilung mit

$$F_k = \int_{\alpha_k}^{\alpha_k + 1} p(x) dx$$

Das Maß $\overline{\chi}^2$ für die Übereinstimmung zwischen Koeffizienten-Histogramm und Laplace-Verteilung ist dann

$$\overline{\chi}^2 = \sum_{k=1}^{K} \frac{(f_k - F_k)^2}{F_k}$$

Es läßt sich ein oberer Grenzwert für $\overline{\chi}^2$ ermitteln, unterhalb dessen ein Block zuverlässig in die Klasse Photo eingeordnet werden kann.



(a) Testbild [Lena]

(b) Gemessene und berechnete Verteilung

Abbildung 3.7: Näherung durch Laplace-Verteilung gleicher Varianz

Grad der Konzentration bei einzelnen Werten

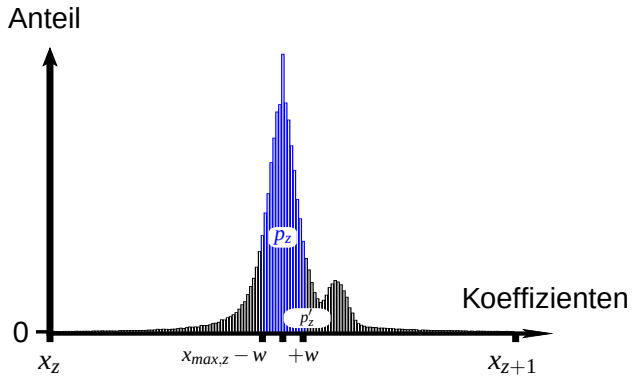
Ein Block kann zuverlässig als Grafik oder Text klassifiziert werden, wenn im Histogramm der Koeffizienten eine Konzentration bei einzelnen Werten auftritt. Um festzustellen, in welchem Maße dies der Fall ist, wird das Histogramm zunächst in Zonen eingeteilt. Jede Zone umfaßt eine Ansammlung der Häufigkeit um einen lokalen Maximalwert der Verteilung, die Werte an den Grenzen der Zone sind beide jeweils um einen Faktor d kleiner als dieser Maximalwert. Dadurch ist gewährleistet, daß die Ermittlung der Konzentrationszentren unempfindlich gegen Störungen ist, wie sie z.B. bei eingescannten Vorlagen auftreten können. Der Faktor d ist experimentell zu ermitteln, ein größerer Wert erhöht die Toleranz gegenüber störungsbedingten lokalen Fluktuationen, kann aber auch zu einer unerwünschten Zusammenlegung eigentlich unabhängiger Zonen führen.

Der in **Abb. 3.9** dargestellte Algorithmus arbeitet das Histogramm linear vom kleinsten bis zum größten Koeffizienten ab und ermittelt die Maximalwerte der Verteilung sowie die Grenzen der zugehörigen Zonen (**Abb. 3.8**).

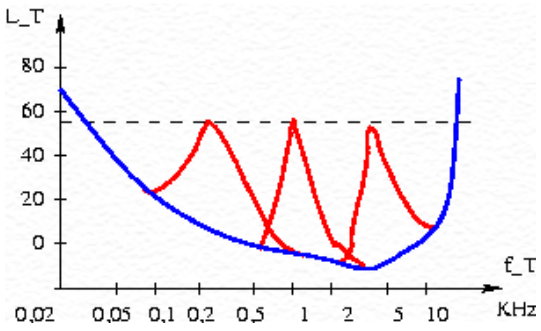
In jeder Zone z wird nun das Maß β_z für den Grad der Konzentration bei dem jeweiligen Maximalwert der Verteilung berechnet. Um den Koeffizienten $x_{max,z}$ herum, bei dem die Verteilung diesen Maximalwert annimmt, wird eine Nachbarschaft von Koeffizienten bestimmt, für die gilt: $|x - x_{max,z}| \leq w$. Es sei p_z die Gesamthäufigkeit der Koeffizienten in dieser Nachbarschaft, p'_z die Gesamthäufigkeit in der ganzen Zone (**Abb. 3.8(a)**). β_z ergibt sich daraus zu p_z/p'_z. Je größer der Grenzwert w gewählt wird, desto eher bescheinigt das Maß eine hohe Konzentration der Verteilung, durch die Festlegung von w läßt sich also die Empfindlichkeit des Maßes einstellen. Das Gesamtmaß L für alle Z Zonen ist

$$L = \sum_{z=1}^{Z} p_z \beta_z$$

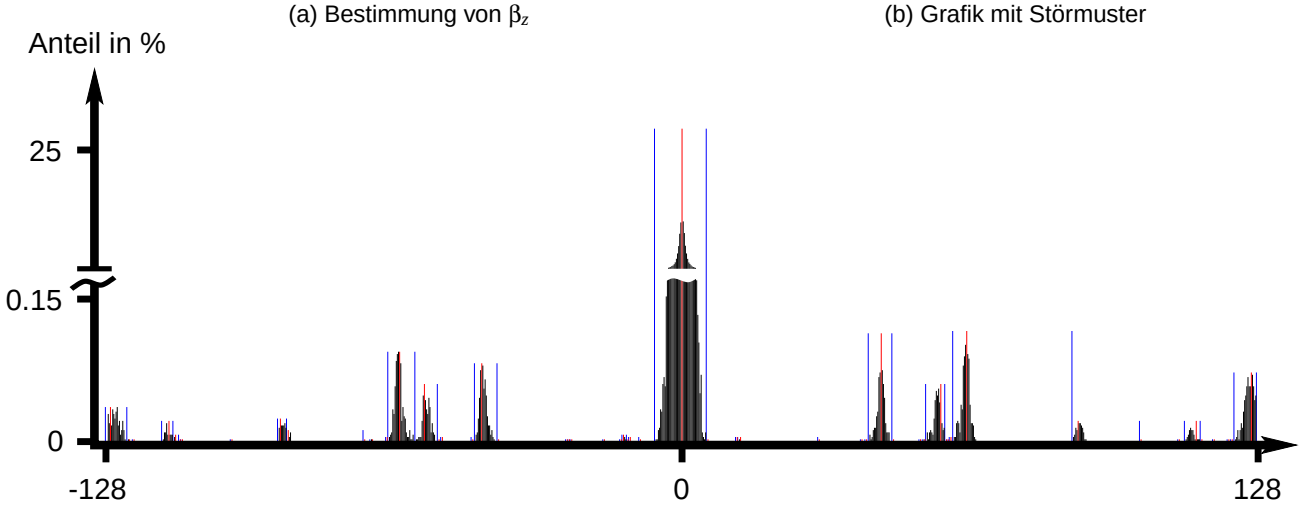
Es gilt $0 < L \leq 1$. Es läßt sich ein unterer Grenzwert für L ermitteln, oberhalb dessen ein Block zuverlässig in die Klasse Grafik oder Text eingeordnet werden kann. Ist $L = 1$ und die Helligkeit der Pixel nimmt nur zwei verschiedene Werte an, wird der Block als Text klassifiziert. Diese einfache Bestimmung der Unterklasse Text nach [LiGr97, LiGr00] funktioniert jedoch nur unter der Annahme, daß Grafiken niemals einfarbig sind, was beispielsweise einfache Strichgrafiken ausschließt. Die Textklassifizierung wurde deshalb durch eine weiterführende Auswertung der Koeffizientenverteilung ersetzt, welche im Zuge der Implementierung (Kap. 4.2) vorgestellt wird.



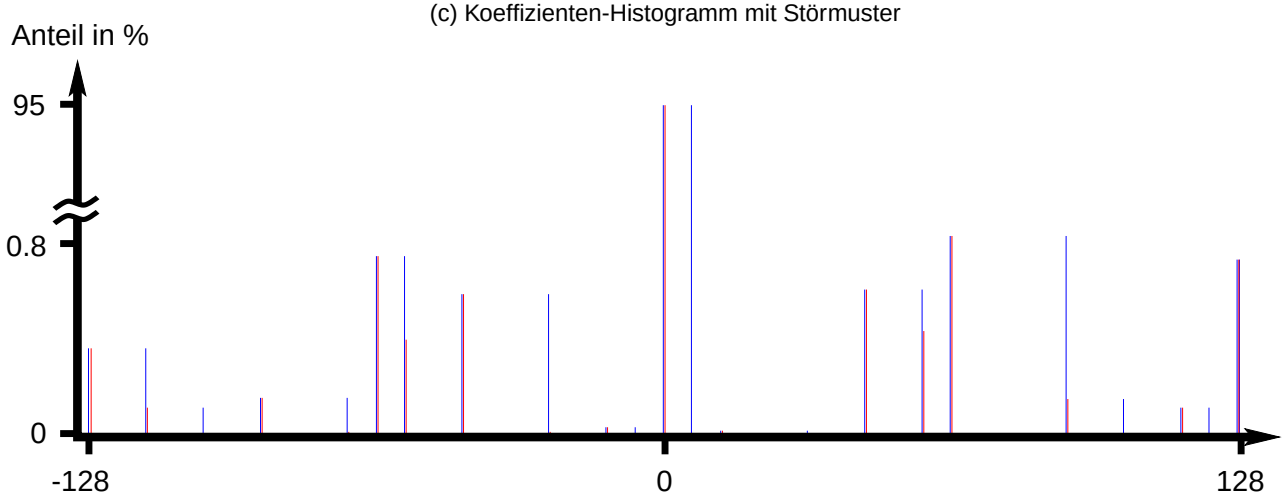
(a) Bestimmung von β_z



(b) Grafik mit Störmuster



(c) Koeffizienten-Histogramm mit Störmuster



(d) Koeffizienten-Histogramm ohne Störmuster

Abbildung 3.8: Zoneneinteilung

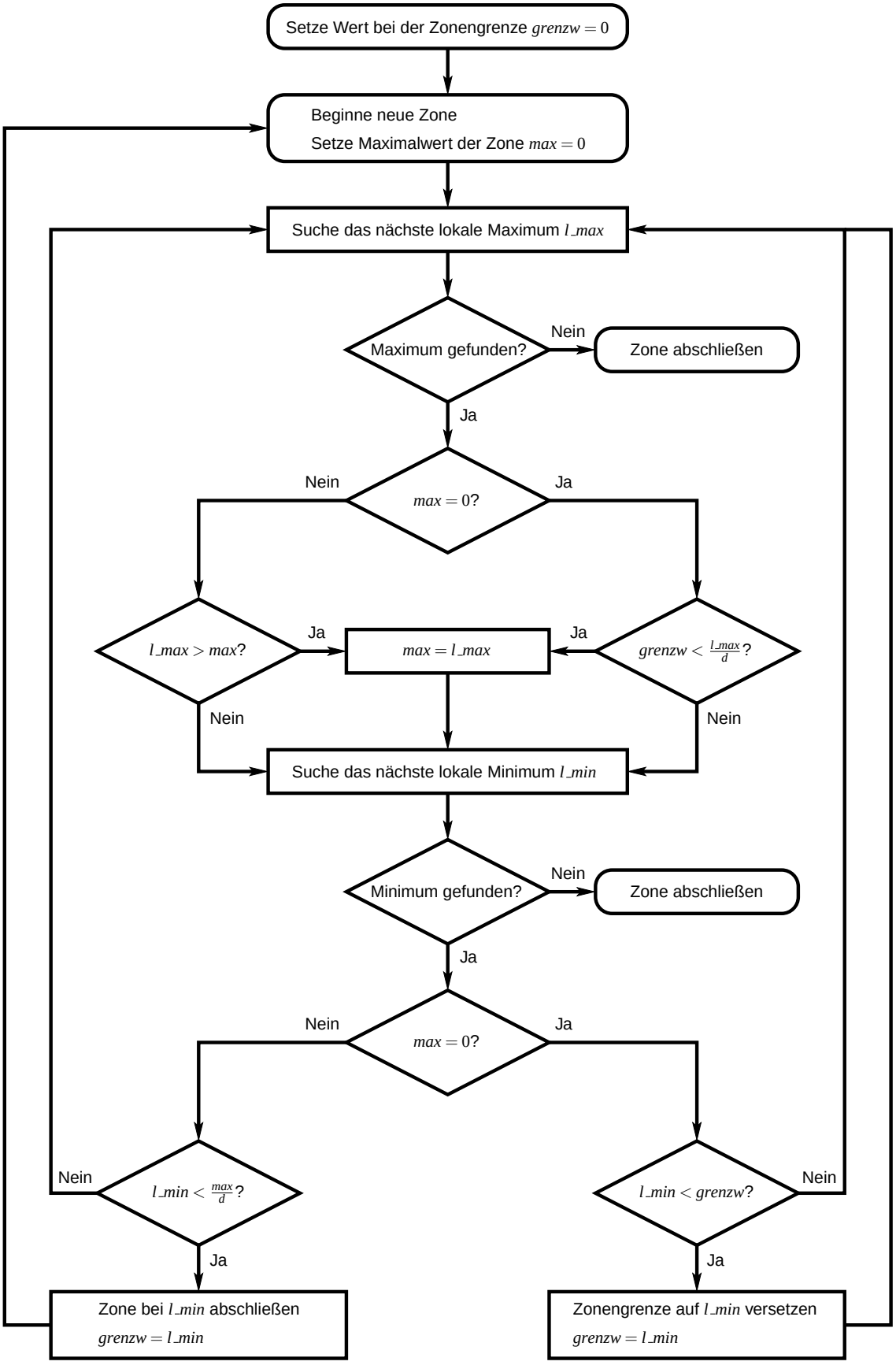


Abbildung 3.9: Algorithmus für die Zoneneinteilung

3.5 Texturanalyse für den Bereich Machine Vision

Die computergestützte Nachbildung des menschlichen Sehvermögens, genannt *Machine Vision*, ist ein weiteres Anwendungsgebiet einer Inhalteseparierung. Hierbei kommt es darauf an, ein Bild in Objekte aufzuteilen und diese ggf. zu erkennen, deren Eigenschaften oder auch Position im Raum zu ermitteln.

Dabei kommt der Texturanalyse eine besondere Bedeutung zu, denn oft sind verschiedene Objekte nicht klar genug voneinander getrennt, um sie beispielsweise mit einfachen Kantendetektionsverfahren zu separieren, obwohl ein Mensch sie problemlos anhand von Unterschieden in der Textur trennen kann. Weiterhin können Informationen aus der Textur selbst gewonnen werden [TuJa98]:

- Defekte in lackierten Oberflächen oder Textilien
- Krankhafte Veränderungen in Röntgen- oder oder Ultraschallbildern.
- Landschaftstypen in Satellitenbildern und deren Übergänge wie etwa Küstenlinien [ChHu99]
- Lage und Ausrichtung von größeren Flächen mit gleichmäßiger Textur

Die Beispiele zeigen, daß anhand der Textur eher eine semantische Analyse des Bildinhaltes vorgenommen werden kann, für eine Separierung im Hinblick auf eine Skalierung ist die Texturanalyse also wenig geeignet. Von Interesse sind allerdings die verschiedenen grundlegenden Methoden, die bei der Texturanalyse Verwendung finden [TuJa98]:

- Statistisch: Beispielsweise Co-Occurrence-Matrizen oder Autokorrelation
- Geometrisch: Eine Textur wird als Zusammensetzung kleiner Grundelemente betrachtet und deren Verteilung analysiert.
- Modellbasiert: Die Textur wird mittels eines Modells, zum Beispiel eines Fraktals, dargestellt und dessen Parameter analysiert.
- Signalverarbeitung: Es werden Techniken aus der Signalverarbeitung wie Filter oder Transformationen angewendet und deren Resultate untersucht.

Auf den letzen Punkt wird im Kapitel 3.4 genauer eingegangen, die anderen seien nur der Vollständigkeit halber erwähnt.

Eine Wavelet-Transformation kann beispielsweise als Hilfsmittel verwendet werden, um Bilder zu erkennen, die als Ganzes eine Textur darstellen [LiWa00]. Das Bild wird in quadratische Blöcke mit 4 Pixeln Kantenlänge unterteilt, und in diesen aus dem durchschnittlichen Farbton sowie den hochfrequenten Koeffizienten nach einer Wavelet-Tranformation ein Kennwert gebildet. Dieser wird verwendet, um ähnliche Blöcke zu Regionen zusammenzufassen, und aus der Verteilung der Regionen im Bild wird schließlich auf die Art des Bildes, Textur oder nicht, geschlossen.

Auch bei der Separierung mehrerer Texturen innerhalb eines Bildes mit teilweise sehr ähnlichen Eigenschaften hat sich die Wavelet-Transformation bewährt [Uns95].

Kapitel 4

Implementierung und Auswertung

Die in Kapitel 3.3 und 3.4 erläuterten Verfahren zur Bildinhaltserkennung anhand statistischer Bildeigenschaften und zur Wavelet-basierten Inhalteseparierung wurden in C implementiert und mittels eines Testdatensatzes von 115 Bildern (siehe Anhang A) aus verschiedenen Kategorien untersucht.

4.1 Statistische Bildeigenschaften

Um die Aussagekraft der in Kapitel 3.3 erläuterten Bildeigenschaften zu untersuchen, wurde der Testdatensatz manuell in die Kategorien Farbbild, Graustufenbild, Schwarz/Weiß-Bild, einfache Grafik, komplexe Grafik, Photo und Text unterteilt. Diese Einteilung sowie die Messergebnisse sind in Anhang B aufgeführt. In Abbildung **Abb. 4.1** sind die Messergebnisse für alle Kategorien gruppiert nach Bildeigenschaften dargestellt, graue Meßstriche bei den Kategorien Grafik und Photo weisen auf Graustufen- bzw. Schwarz/Weiß-Bilder hin. Im Einzelnen ergibt sich folgende Auswertung:

4.1.1 Durchschnittliche Sättigung

Entgegen den Angaben in der Literatur konnte eine Entscheidung zwischen Photos und Grafiken anhand der durchschnittlichen Sättigung nicht nachvollzogen werden. Allerdings kann sie mit großer Sicherheit zur Unterscheidung von Farbbildern und Graustufen- bzw. Schwarz/Weiß-Bildern herangezogen werden.

4.1.2 Anteil schwarzer, weißer und grauer Pixel

Bei der Implementierung wurden Pixel mit einem Helligkeitswert kleiner als 25 als schwarz angenommen, sonst mit einem Sättigungswert kleiner als 25 als grau bzw. bei einem zusätzlichen Helligkeitswert größer als 230 als weiß. Der Anteil schwarzer Pixel zeigte keine verwertbaren Unterschiede zwischen den Kategorien, während der Anteil grauer und weißer Pixel bedingt zur Unterscheidung von Farbbildern und Graustufen- bzw. Schwarz/Weiß-Bildern herangezogen werden kann.

4.1.3 Anteil voll- und halbgesättigter Pixel

Bei der Implementierung wurden Pixel mit einem Sättigungswert größer als 230 als vollgesättigt angenommen, sonst mit einem Sättigungswert größer als 127 als halbgesättigt. Die in der Literatur anhand der Anteile voll- und halbgesättigter Pixel vorgenommene Unterscheidung von Photos, einfachen und komplexen Grafiken konnte in der Mehrheit der Fälle nicht bestätigt werden.

4.1.4 Anzahl verwendeter Helligkeits-, Sättigungs- und Farbtonstufen

Die Erkennung einfacher Grafiken anhand einer deutlich geringeren Anzahl von Helligkeits-, Sättigungs- und Farbtonstufen konnte nachvollzogen werden.

Die geringere Anzahl von Sättigungs- und Helligkeitsstufen bei komplexen Grafiken im Vergleich zu Photos konnte in den meisten Fällen ebenfalls nachvollzogen werden, nicht jedoch eine entsprechend größere Anzahl von Farbtonstufen, hier ergab sich das genaue Gegenteil.

4.1.5 Anzahl verwendeter Farben im quantisierten HSV-166-Farbraum

Eine deutlich geringere Anzahl verwendeter Farben im quantisierten HSV-166-Farbraum bei einfachen Grafiken im Vergleich zu komplexen Grafiken und Photos konnte bestätigt werden, zwischen Photos und komplexen Grafiken konnte jedoch kein deutlicher Unterschied festgestellt werden.

4.1.6 Varianz und Entropie der Intensität

Die Varianz der Intensität ergab entgegen den Angaben in der Literatur keinerlei Anhaltspunkte. Eine Unterscheidung zwischen Photos und Grafiken anhand deutlich geringerer Entropie bei letzteren konnte jedoch bestätigt werden. Auch lieferte die Entropie der Intensität auch bei Farbbildern ein deutlicheres Ergebnis als im HSV-166-Raum.

4.1.7 Wechselhäufigkeit der Intensität

Eine deutlich geringere Wechselhäufigkeit der Intensität bei Grafiken im Vergleich zu Photos konnte bestätigt werden, im HSV-166-Raum ließen sich bei der Wechselhäufigkeit jedoch keine Unterschiede feststellen.

Zusammenfassend läßt sich sagen, daß mithilfe der genannten Bildeigenschaften eine Unterscheidung von Farbbildern und Graustufen- bzw. Schwarz/Weiß-Bildern sehr gut möglich ist, eine Unterscheidung von Grafiken und Photos ebenfalls. Allerdings ist nur letztere im Rahmen dieser Arbeit von Interesse, für die zusätzliche Kategorie Text, die in dieser Arbeit von Bedeutung ist, konnten jedoch keine aussagekräftigen Bildeigenschaften festgestellt werden. Darüberhinaus sind die statistischen Bildeigenschaften nicht zur Separierung von Bildinhalten geeignet, da sie in kleinen Bereichen ohne ausreichende Durchschnittsbildung nicht mehr aussagekräftig sind. Bei der Implementierung der Separierung wurde daher von einer Verwendung dieser Bildeigenschaften abgesehen.

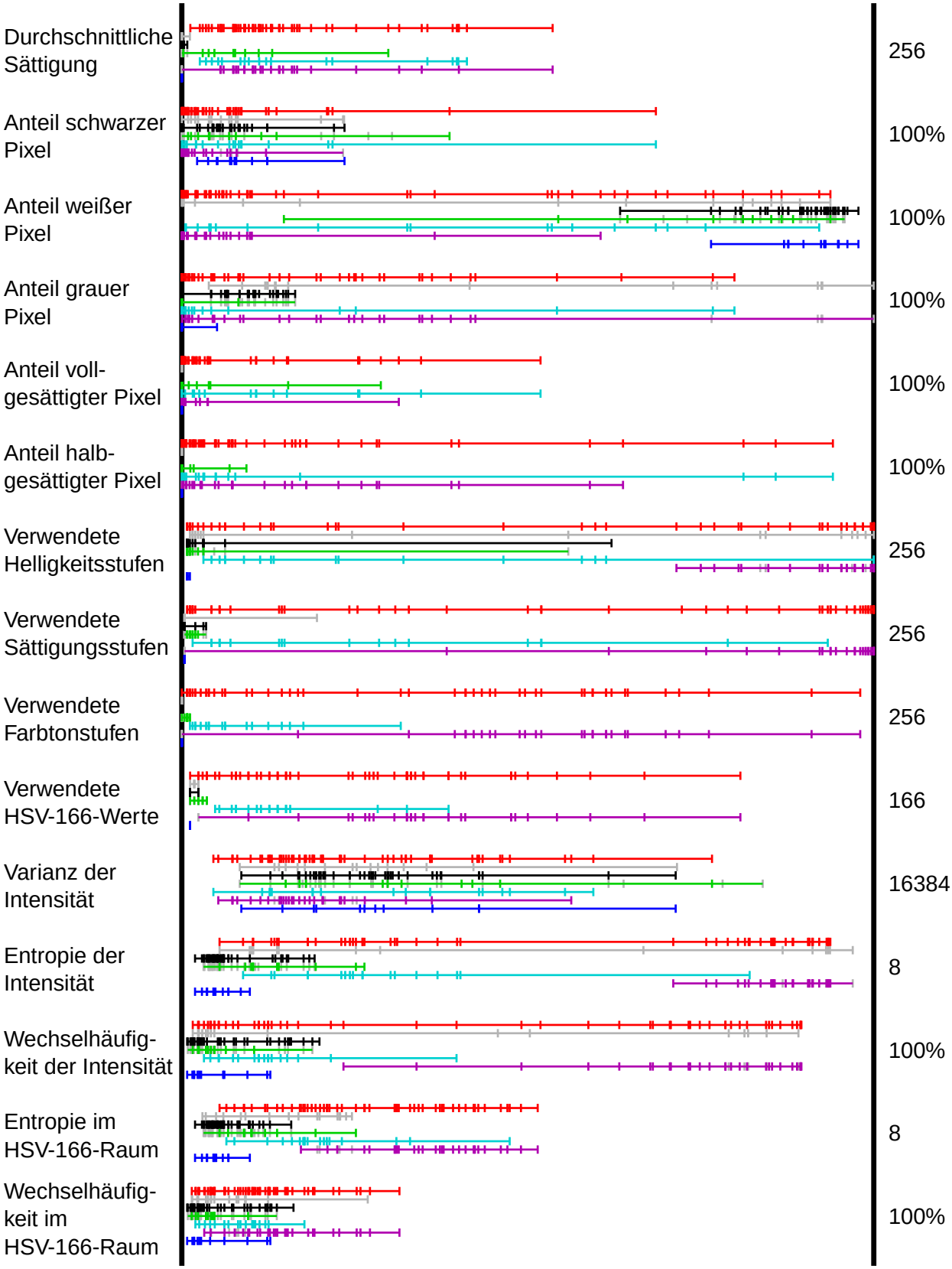


Abbildung 4.1: Messergebnisse der statistischen Bildeigenschaften

4.2 Wavelet-basierte Separierung

Bei der Implementierung (siehe Anhang D.6) des in der Literatur [LiGr97, LiGr00] beschriebenen Verfahrens zur Wavelet-basierten Separierung unterschiedlicher Bildinhalte mußten viele nur oberflächlich erläuterte Vorgehensweisen neu erarbeitet werden. Details der Implementierung und unbekannte Parameter und Grenzwerte mußten anhand des Testdatensatzes an die gewünschte Verwendung für eine inhaltsadaptive Skalierung angepaßt werden. Diese Anpassungen werden im Folgenden näher betrachtet.

Zur Realisierung der Filter des Haar-Wavelets (siehe 3.4.2) muß lediglich aus jeweils einem Wertepaar der Mittelwert gebildet werden, für den Hochpass wird zuvor noch das Vorzeichen des zweiten Wertes umgekehrt. Daher konnte auf die vollständige Realisierung einer Faltung (siehe z.B. [TargJr]) verzichtet werden.

Es wird nur ein Transformationsschritt angewendet, also je eine Mittelwert- bzw. Differenzbildung für die zwei Dimensionen des Bildes. Bei dem verwendeten Bildformat des Testdatensatzes sind die Helligkeitswerte der Pixel Integerwerte zwischen 0 bis 255, also liegen die resultierenden Koeffizienten der hochfrequenten Bänder in Schritten von 0.25 zwischen -128 und 128 vor, die Gesamtzahl der möglichen Koeffizienten beträgt also lediglich 1024. Für die in 3.4.2 im Hinblick auf beliebige Koeffizientenverteilungen beschriebene Bereichseinteilung des Histogramms konnte die Anzahl der Bereiche in diesem Fall also einfach auf 1024 festgelegt werden.

Die Laplace-Verteilung nimmt mit wachsender Entfernung vom Zentrum schnell sehr kleine Werte an. Da das in 3.4.2 beschriebene Maß $\overline{\chi^2}$ auf die erwartete Koeffizientenhäufigkeit normiert wird, entstehen durch sehr kleine Werte derselben extreme Verfälschungen. Die äußeren Bereiche mit einer erwarteten Anzahl kleiner als 0.1 werden daher nicht berücksichtigt.

Als oberer Grenzwert von $\overline{\chi^2}$ für die mögliche Klassifizierung als Photo wurde experimentell (siehe 4.2.1) der Wert 35 ermittelt.

Bei der Zoneneinteilung (siehe 3.4.2) wurde für den Faktor d zwischen Maximalwert und Randwert einer Zone aus der Literatur der Wert 20 übernommen. Der Wert w, der die Ausdehnung des Haupt-Bereiches einer Zone um das Maximum herum angibt, wurde auf einen niedrigen Wert von 0.25 festgelegt, um nur eindeutig als Grafik erkennbare Blöcke als solche zu klassifizieren.

Der untere Grenzwert des Konzentrationsmaßes L für die mögliche Klassifizierung als Grafik ist abhängig von der Blockgröße, da kleinere Blöcke allein durch die geringere Gesamtzahl möglicher Koeffizienten schon eine höhere Konzentration aufweisen. Für Blöcke von 16x16 Pixeln wurde experimentell (siehe 4.2.1) der Wert 0.94 bestimmt, für 8x8 Pixel der Wert 0.97 und für noch kleinere Blöcke der Wert 0.99. Da an den Bildrändern auch rechteckige Blöcke mit anderen Kantenlängen auftreten können, wurden alle Blöcke mit einer Gesamtzahl von mindestens 100 Pixeln der Größe 16x16 zugeordnet, mit mindestens 64 Pixeln der Größe 8x8.

Bei der endgültigen Festlegung des Blocktyps wird Grafik gegenüber Photo bevorzugt. Text wird als bevorzugte Unterklasse von Grafik behandelt, für Textblöcke gilt allerdings eine Mindestgröße von 8x8 Pixeln. Diese Prioritisierung ergab bei dem Testdatensatz die beste Übereinstimmung mit den manuell eingeteilten Kategorien.

Die Blöckgröße beginnt bei 16x16 Pixeln, da größere Blöcke bei den im Testdatensatz durchschnittlich verwendeten Bildgrößen zunehmend mehrere Inhaltstypen umfassen und daher zu Fehlklassifizierungen führen. Kann ein Block nicht sicher genug klassifiziert werden, wird die Klassifizierung rekursiv mit jeweils vier Unterblöcken durchgeführt, bis eine Entscheidung möglich ist.

Das in der Literatur angegebene Kriterium für eine Klassifizierung als Text, ein Konzentrationsmaß von 1 und nur zwei Helligkeitswerte der Pixel, hat sich als völlig unzureichend erwiesen, da dies auch beispielsweise für einfache Strichgrafiken zutrifft. Daher wurde experimentell auf Basis des Testdatensatzes ein völlig neues Kriterium entwickelt, welches direkt das bei der Bestimmung des Konzentrationsmaßes ermittelte Koeffizienten-Histogramm auswertet:

- Für den negativen und positiven Bereich der Koeffizienten werden jeweils die beiden Zonen mit der größten und zweitgrößten Gesamtzahl von Koeffizienten ermittelt, genannt $Z_{-gr}, Z_{-2gr}, Z_{+gr}$ und Z_{+2gr}. Um einen ausreichenden Textkontrast sicherzustellen und Einflüsse des Texthintergrundes zu vermeiden, werden nur Koeffizienten mit einem Betrag größer als 25 berücksichtigt, und die Maxima der Zonen Z_{gr} und Z_{2gr} müssen jeweils eine Distanz von mindestens 25 aufweisen.
- Für eine mögliche Klassifizierung als Text darf die Gesamthäufigkeit der Koeffizienten in diesen beiden Zonen einen blockgrößenabhängigen Grenzwert n nicht unterschreiten und muß um den Faktor 15 größer sein als die Gesamtanzahl im restlichen Koeffizientenbereich gleichen Vorzeichens, um eine fehlerhafte Klassifikation als Text durch zufällige Schwankungen der Verteilung zu vermeiden. Für Blöcke mit einer Kantenlänge $l \geq 10$ wurde experimentell für n der Wert 44, sonst 18 bestimmt.
- Wenn nun eine der beiden Zonen Z_{2gr} betragsgrößere Koeffizienten umfaßt als die zugehörige Zone Z_{gr}, also im Histogramm weiter außen zu finden ist, ist eine Klassifizierung als Text möglich. Ist dagegen die jeweilige Zone Z_{gr} weiter außen, darf deren Koeffizientenanzahl maximal um $\frac{1}{6}$ größer sein als in der zugehörigen Zone Z_{gr2}.

Abb. 4.2 zeigt beispielhaft eine entsprechende Verteilung.

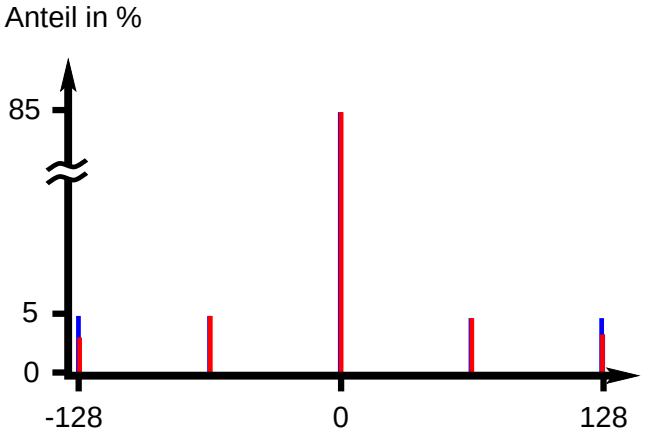


Abbildung 4.2: Koeffizienten-Histogramm von „Textblock“ (siehe Anhang A)

4.2.1 Ergebnisse bei kompletten Bildern

Die Implementierung der Wavelet-basierten Inhalteseparierung wurde zunächst für Bilder im Ganzen vorgenommen. Die Ergebnisse für den Testdatensatz sind in Anhang C aufgeführt. Es läßt sich deutlich erkennen, wie zuverlässig anhand der in 3.4.2 erläuterten Maße Laplace-Match $\overline{\chi}^2$ und Konzentrationsmaß L reine Photos und Grafiken unterschieden werden können.

$\overline{\chi}^2$ ist bei fast allen natürlichen Bildern deutlich kleiner als 10, oft sogar kleiner als 1. Einen Grenzfall stellt beispielsweise „Flughafen3“ dar, welches synthetisch generiert wurde und stellenweise eher grafische Elemente enthält. Bei Grafiken und Text ist dieses Maß dagegen gewöhnlich deutlich größer als 50, bis hin zu Werten größer als 300. In dieser Kategorie erweisen sich die Cliparts wie „Hase“als Grenzfälle, da sie teilweise natürlichen Bildern sehr nahe kommen.

Das Konzentrationsmaß L liefert ähnlich eindeutige Ergebnisse, bei Grafiken und Text ist der Wert meist größer als 0.94, bei natürlichen Bildern kleiner als 0.5. Die Grenzfälle decken sich dabei überwiegend mit denen des Laplace-Match.

Wenn $\overline{\chi}^2$ größer als 10 ist oder L kleiner als 0.94, muß davon ausgegangen werden, daß das Bild mehrere Inhaltstypen beinhaltet. Diese können durch eine blockweise Analyse separiert werden, die im Folgenden betrachtet wird.

4.2.2 Ergebnisse der Separierung

Anhand einiger Bilder mit gemischten Bildinhalten sollen die Ergebnisse der in 4.2 beschriebenen Realisierung einer Wavelet-basierten Inhalteseparierung näher betrachtet werden.

Abbildung 4.3: Screenshot der Suchmaschine Google

Abb. 4.3 zeigt beispielsweise den Ausschnitt einer Webseite, auf der die drei zu unterscheidenden Inhaltstypen Photo, Grafik und Text vertreten sind. Die als Photo klassifizierten Blöcke sind rot und Textblöcke grün markiert, Blöcke mit grafischem Inhalt ausgespart. Bei der Markierung wurden benachbarte Blöcke gleichen Inhaltstyps zur besseren Erkennbarkeit zusammengefaßt.

Der Inhaltstyp Photo wurde zuverlässig erkannt; es handelt sich zwar um Buchstaben, die aber als räumliche Objekte dargestellt sind und somit beim Skalieren wie ein natürliches Bild behandelt werden müssen.

Auch die gute Trennung der verbleibenden Grafik- und Textelemente ist erkennbar, jedoch auch die Auswirkungen der festen Blockeinteilung. So konnte beispielsweise Grafik und Text bei den zwei Buttons nicht separiert werden, da die Grenze zwischen den zwei Inhaltstypen nicht der Grenze der betroffenen Blöcke entsprach.

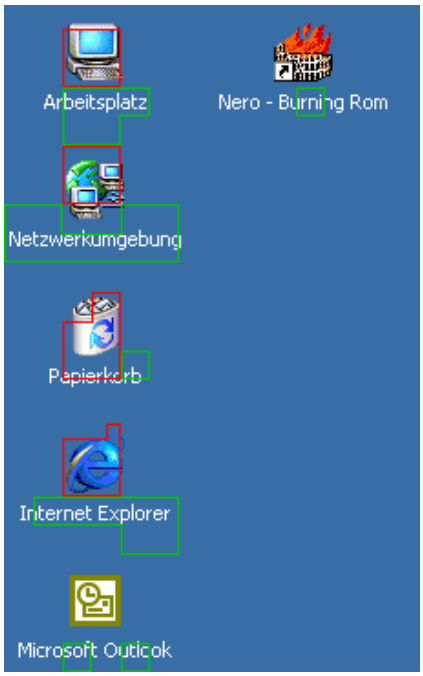


Abbildung 4.4: Screenshot eines Windows-Desktops

Auch in **Abb. 4.4** wird die zuverlässige Erkennung des Inhaltstyps Photo bei Icons, welche Ansichten räumlicher Objekte darstellen, deutlich. Die Beschriftung der Icons ist größtenteils als Text erkannt worden, wobei sich jedoch auch hier teilweise die feste Blockeinteilung ungünstig ausgewirkt hat.

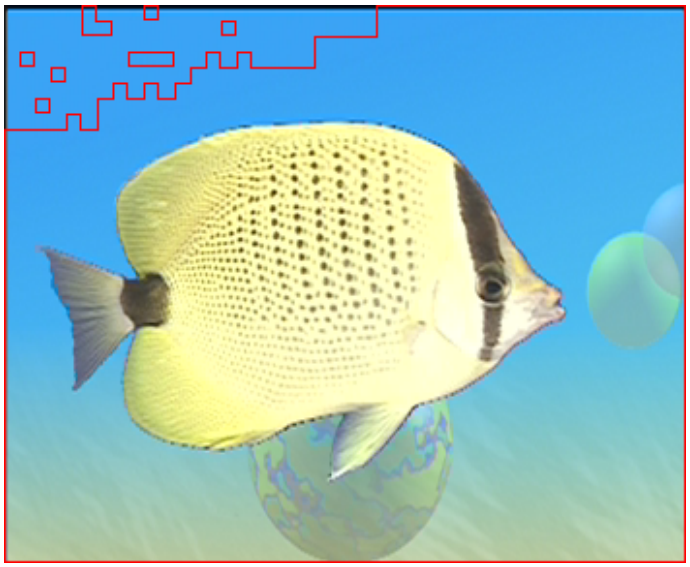


Abbildung 4.5: Natürliches Bild

Abb. 4.5 ist ein natürliches Bild, welches erwartungsgemäß überwiegend als Photo klassifiziert wurde. Wie in den vorausgegangenen Beispielen sind auch hier weitgehend einfarbige Hintergrundflächen davon ausgenommen. Da homogene Flächen bei einer Skalierung unabhängig vom verwendeten Verfahren unkritisch sind, sollte dafür das Verfahren mit dem geringsten Rechenaufwand gewählt werden. Von den erläuterten Verfahren ist dies die Nearest-Neighbour-Interpolation

(siehe 2.1), welche häufig für Grafik verwendet wird, weshalb diese Flächen dem Typ Grafik zugeordnet werden.

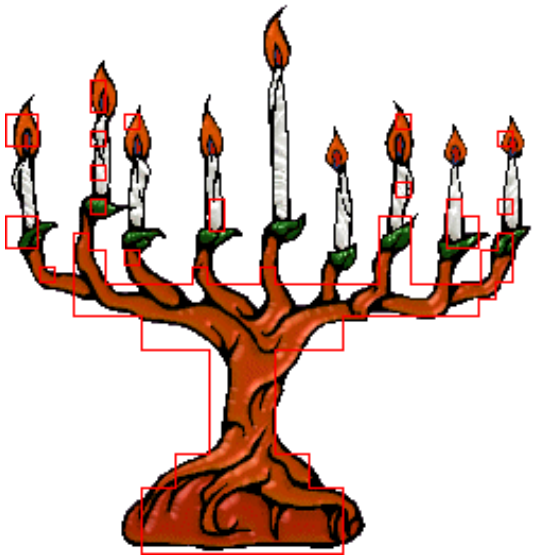


Abbildung 4.6: Zeichnung einer Kerze

Abb. 4.6 zeigt einen Grenzfall zwischen natürlichem Bild und Grafik. Einerseits sind die schwarz umrandeten Konturen der Kerzen bei einer Skalierung anfällig für Unschärfe, andererseits sind diese im Stamm als Teil eines natürlichen Bildes zu betrachten. Entsprechend differenziert ist somit auch die vorgenommene Einteilung, die bei späterer Festgelegung auf bestimmte Skalierungsalgorithmen durch Einstellung der in Kapitel 4.2 beschriebenen Parameter justiert werden kann.

Abbildung 4.7: Eingescanntes Schwarz-Weiß-Bild

Auch **Abb. 4.7** ist ein solcher Grenzfall. Die durch den Scanvorgang entstandene Unschärfe rechtfertigt zwar insbesondere bei feinen Strukturen die Klassifizierung als Photo, im Wesentlichen ist sie jedoch unerwünscht und der Inhalt daher als Grafik zu behandeln.

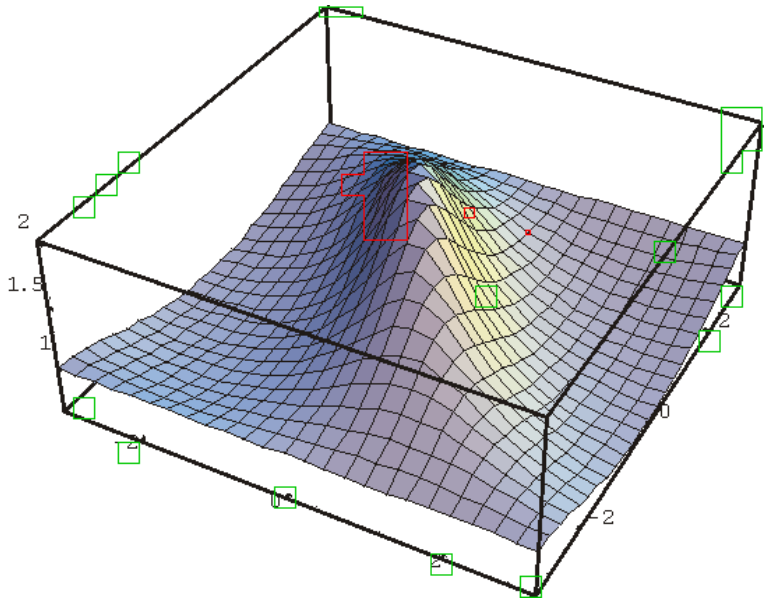


Abbildung 4.8: Komplexe Grafik

Abb. 4.8 ist eine komplexe Grafik, die auch größtenteils als solche erkannt wird. Einige Elemente der Achsenbeschriftung werden sogar korrekt als Text erkannt, allerdings auch fälschlicherweise einige Elemente der Grafik. Dieses Problem ergibt sich aus der getrennten Behandlung der einzelnen Blöcke; betrachtet man die irrtümlich als Text klassifizierten Blöcke losgelöst vom Rest des Bildes, kann man sie sich durchaus als Teil eines größeren Textblockes vorstellen. Auffällig ist der als Photo klassifizierte Bereich, wo das farbig gefüllte Gitternetz am dichtesten und der Informationsgehalt dementsprechend hoch ist. Bei Verwendung einer für natürliche Bildinhalte optimierten Skalierung in diesem Bereich ist daher trotz entstehender Unschärfe ein besserer visueller Eindruck zu erwarten als bei einer Behandlung als Grafik.

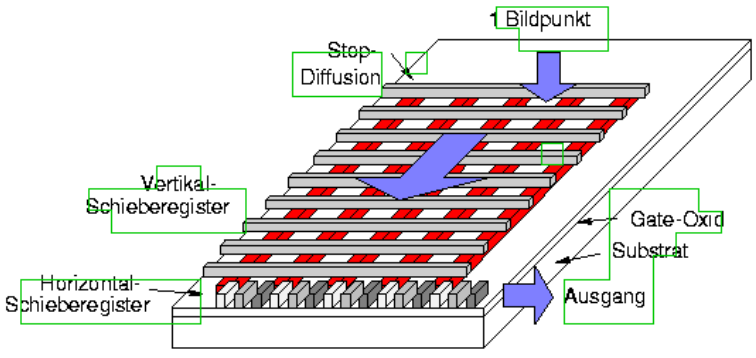


Abbildung 4.9: Grafik mit Textelementen

Abb. 4.9 ist ein typisches Beispiel für eine Grafik mit einzelnen isolierten Textelementen, welche von dem Verfahren zuverlässig erkannt werden. Aus demselben Grund wie im vorhergehenden Beispiel sind jedoch auch hier zwei kleine Blöcke fälschlicherweise als Text klassifiziert worden.

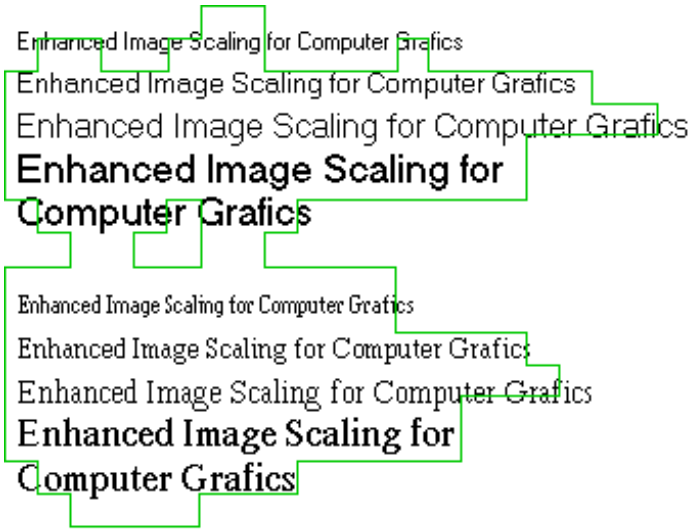


Abbildung 4.10: Text in verschiedenen Schriftarten und -größen

Wie in **Abb. 4.10** zu erkennen, funktioniert die Texterkennung auch bei verschiedenen Schriftarten und -größen recht zuverlässig, aber auch hier ist eine Verschlechterung durch ungünstig platzierte Blockgrenzen deutlich zu erkennen.

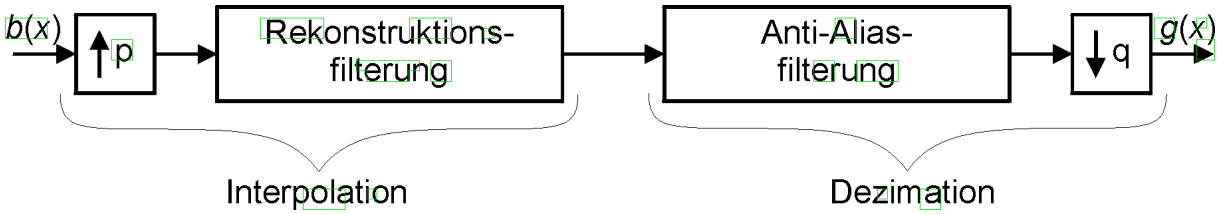


Abbildung 4.11: Komplexe Grafik

Das in **Abb. 4.11** dargestellte Bild hat eine Größe von 1820 x 307 Pixeln. Hier funktioniert die Texterkennung nicht mehr zuverlässig, da die überwiegende Anzahl der Buchstaben für die anfängliche Blockgröße von 16x16 Pixeln zu groß sind. Die Ausgangsgröße des Bildes und die Skalierungsfaktoren bewegen sich jedoch abhänging von dem Umfeld, in dem das Verfahren eingesetzt wird, gewöhnlich in gewissen Grenzen, sodaß das Verfahren durch Einstellung der in Kapitel 4.2 beschriebenen Parameter an die entsprechenden Rahmenbedingungen angepaßt werden kann.

Die nach optimaler Einstellung der Parameter verbleibenden Abweichungen von einer ansonsten zuverlässigen Klassifizierung lassen sich also insgesamt auf die festen Blockgrenzen und die isolierte Betrachtung der einzelnen Blöcke zurückführen. Eine flexiblere Festlegung der Blockgrenzen und eine Einbeziehung bereits klassifizierter benachbarter Blöcke bei der Entscheidung wäre daher wünschenswert, konnte aber im Hinblick auf den Umfang dieser Arbeit nicht realisiert werden.

Kapitel 5

Zusammenfassung und Ausblick

In dieser Arbeit wurden verschiedene Verfahren zur automatisierten Klassifizierung von Bildinhalten untersucht und auf ihre Verwendbarkeit für eine inhaltsadaptive Skalierung hin überprüft. Nach einer Übersicht über bestehende Bildskalierungsalgorithmen und Klassifizierungsverfahren wurden einige zur Klassifizierung verwendete statistische Bildeigenschaften wie z.B. der Anteil schwarzer, weißer und grauer Pixel oder die Varianz und Entropie der Intensität näher betrachtet und ihre Aussagekraft anhand eines Testdatensatzes von 115 Bildern überprüft. Da diese Bildeigenschaften schon bei ganzen Bildern oft kein eindeutiges Ergebnis lieferten, wurde die Verwendung für die gewünschte Separierung gemischter Bildinhalte nicht in Betracht gezogen.

Stattdessen wurde ein Wavelet-basiertes Verfahren untersucht und implementiert, welches sich bei ganzen Bildern als sehr aussagekräftig erwies. Dieses Verfahren wurde durch experimentelle Bestimmung zahlreicher Parameter und Einführung eines völlig neuen Kriteriums zur Texterkennung dahingehend erweitert, daß es in der nun vorliegenden Form zuverlässig Blöcke von 16x16 Pixeln oder kleiner unterscheiden kann, deren Bearbeitung entweder für natürliche Bildinhalte, synthetische Bildinhalte wie Strichgrafiken oder Text optimiert werden sollte.

Diese Einteilung ist noch relativ allgemein, da sich Skalierungsalgorithmen für verschiedene Inhaltstypen noch in der Entwicklung befinden und deren Anforderungen noch nicht klar definiert sind. Es existieren aber zahlreiche Parameter, die variiert werden können, um das Verfahren neuen Anforderungen anzupassen. So kann die Blockgröße auf die erwartete Bildauflösung abgestimmt werden, die Empfindlichkeit gegenüber Störungen oder die Entscheidungsschwellen zwischen den verschiedenen Inhaltstypen eingestellt werden.

Auch Erweiterungen zur Unterscheidung differenzierterer Inhaltstypen, also etwa Unterklassen von synthetischen Bildinhalten, sind durchaus denkbar. Beispielsweise könnten weitere Transformationsschritte oder verschiedene Wavelets zusätzliche Informationen liefern. Eine zuverlässigere Klassifizierung könnte durch flexiblere Festlegung der Blockgrenzen und eine Betrachtung der Abhängigkeiten benachbarter Blöcke untereinander erreicht werden.

Literaturverzeichnis

[Ben92] M.J. Bentum, R. G. J. Arendsen et al., Design and Realization of High Speed Single Exposure Dual Energy Image Processing, *Proceedings of the Fifth Annual IEEE Symposium Computer-based Medical Systems*, Durham, USA, 1992, Seiten 25-34

[Theu99] T. Theußl; On Windowing for Gradient Estimation in Volume Visualization, *Proceedings of the 3rd Central European Seminar on Computer Graphics for students (CESCG '99)*, Budmerice, Slovakia, April 1999

[SOS] Gibbs Phenomenon, S.O.S. MATHematics, http://www.sosmath.com/fourier/fourier3/gibbs.html

[W3col] Cascading Style Sheets level 2 Specification, Colors and Backgrounds, http://www.w3.org/TR/REC-CSS2/colors.html

[W3lin] Cascading Style Sheets level 2 Specification, Border properties, http://www.w3.org/TR/REC-CSS2/box.html#border-properties

[W3frm] HTML 4.01 Specification, Form Controls, http://www.w3.org/TR/REC-html40/interact/forms.html#h-17.2.1

[WDG] Web Design Group - Image Use on the Web, http://www.htmlhelp.com/design/imageuse.htm#jpgvsgif

[LANL] Information Architecture White Paper IA-6801: Electronic Image Formats and Compression Algorithms, Los Alamos National Laboratory, http://preserve.harvard.edu/resources/IA6801.htm

[JPEG] The JPEG tutorial, http://www.ece.purdue.edu/~ace/jpeg-tut/jpegtut1.html

[GIF] The GIF89a Specification, http://morgner.com/Dev_GIF/

[FlKa88] L. A. Fletcher; R. Kasturi; A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Band 10, Ausgabe 6, Seiten 910-918, November 1988

[WaSr89] D. Wang; S. N. Srihari; Classification of Newspaper Image Blocks using Texture Analysis, *Computer Vision, Graphics and Image Processing*, Band 47, Seiten 327-352, 1989

[HaHi01] K. Hadjar; O. Hitz; R. Ingold; Newspaper Page Decomposition using a Split and Merge Approach, *ICDAR 2001: Sixth International Conference on Document Analysis and Recognition*, Seattle, September 2001

[WuMa97] V. Wu; R. Manmatha; E. Riseman; Finding Text in Images, *Proceedings of the 2nd ACM International Conference on Digital Libraries*, Philadaphia, Juli 1997

[TuJa98] M. Tuceryan; A. K. Jain; Texture analysis. In C. H. Chen; L. F. Pau; P. S. P. Wang, editors, *Handbook of Pattern Recognition and Computer Vision*, World Scientific Publishing, 1998

[ChHu99] Li-Yu Chang; A. J. Chen; C. F. Chen; C. M. Huang; A Robust System for Shoreline Detection and its Application to Coastal-zone Monitoring http://www.gisdevelopment.net/aars/acrs/1999/ts8/ts8117.shtml

[LiWa00] Jia Li; J. Z. Wang; G. Wiederhold; Classification of textured and non-textured images using region segmentation, *Proceedings of the International Conference on Image Processing*, Band 3, Seiten 754-757, 2000

[AtSw97] V. Athitsos; M. J. Swain; Distinguishing photographs and graphics on the World Wide Web, *Proceedings IEEE Workshop on Content-Based Access of Image and Video Libraries*, Seiten 10-17, 1997

[SmCh97] J. R. Smith; S.-F. Chang; Multi-stage Classification of Images from Features and Related Text, *Proceedings Fourth DELOS workshop*, Pisa, August 1997

[SmLi98] J. R. Smith; Rakesh Mohan; C.-S. Li; Content based transcoding of images in the internet, *Proceedings of the International Conference on Image Processing*, September 1998

[SmCh96] J. R. Smith; S.-F. Chang; Tools and Techniques for Color Image Retrieval, *Storage and Retrieval for Image and Video Databases IV*, Band 2670, Seiten 426-437, Februar 1996

[LiGr97] Jia Li; R. M. Gray; Text and Picture Segmentation by the Distribution Analysis of Wavelet Coefficients, *Proceedings of the International Conference on Image Processing*, Chicago, Oktober 1998

[LiGr00] Jia Li; R. M. Gray; Context-based multiscale classification of document images using wavelet coefficient distributions, *IEEE Transactions on Image Processing*, Band 9 Ausgabe 9, Seiten 1604-1616, September 2000

[Uns95] M. Unser; Texture Classification and Segmentation Using Wavelet Frames, *IEEE Transactions on Image Processing*, Band 4 Ausgabe 11, Seiten 1549-1560, Nov. 1995

[RiVe91] O. Rioul; M. Vetterli; Wavelets and Signal Processing, *IEEE Signal Processing Magazine*, Band 8 Ausgabe 4, Seiten 14-38, Oktober 1991

[SmRo96] S. R. Smoot; L. A. Rowe; Laplacian Model for AC DCT Terms in Image and Video Coding, *Proceedings of the Ninth IEEE Image and Multidimension Digital Signal Processing Workshop*, Belize City, Belize, März 1996

[Robi] The engineer's ultimate guide to wavelet analysis, Robi Polikar, http://engineering.rowan.edu/˜polikar/WAVELETS/WTtutorial.html

[TargJr] TargetJr ImageProcessing library, class FloatOperators http://www.esat.kuleuven.ac.be/˜targetjr/

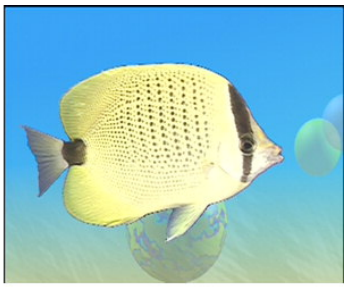
[Lena] The Lena story http://www.lenna.org

Anhang A

Testdatensatz

akiyo

boat



bream

carphone

football

foreman

kerstin

lena_c

miss

news

salesman

sean

stefan

weather

Autorennen1

Autorennen2

Autorennen3

Flughafen

Flughafen2

Flughafen3

Flughafen4

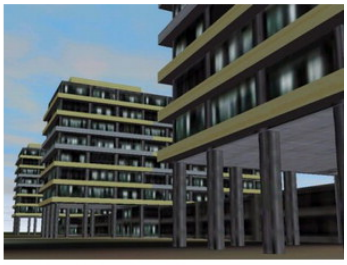
Florian

Fussball

Football_VR

Hockey

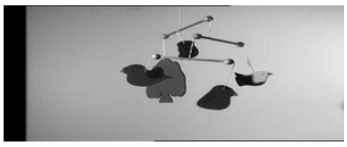
Luftbild1



Uni-Bochum

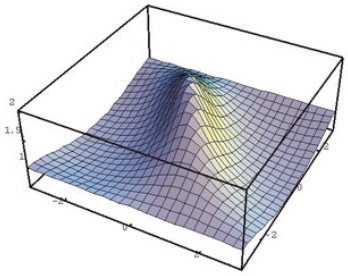
blume

kewgarden

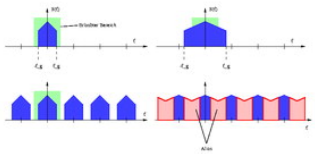


mobile

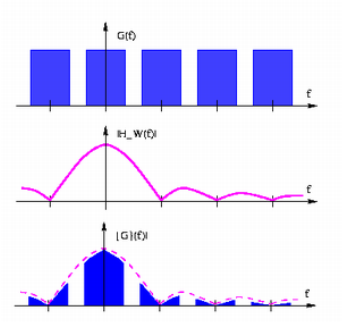
teen



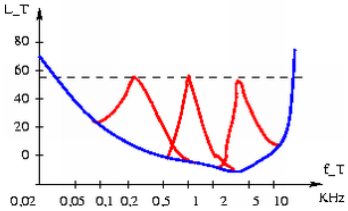
3d_plot



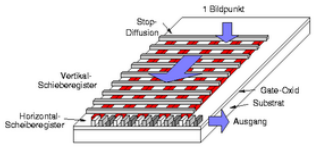
abtastung_mit_und_ohne_alias



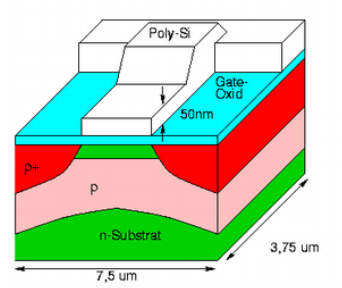
abtastung_mit_wiedergabeapertur



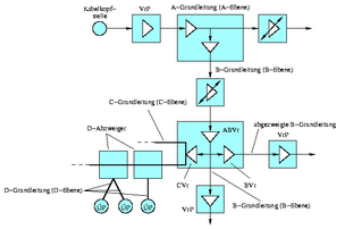
bild_reimerR3



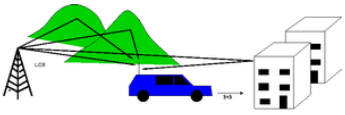
ccd-ganz



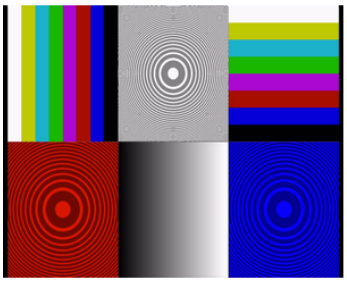
ccd-pixel_c



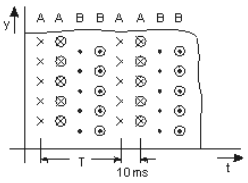
kabelnetz



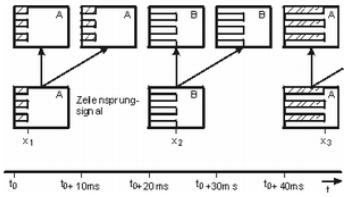
mobilfunk_kanal



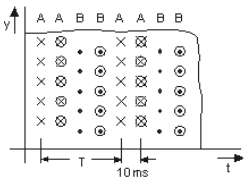
mulpic



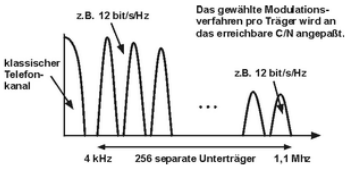
100Hz



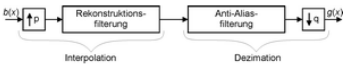
100Hz2



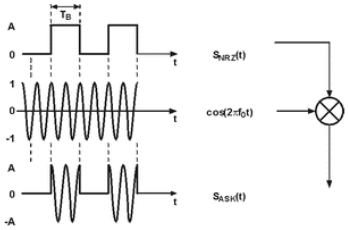
100Hzplus



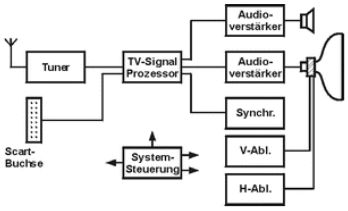
1D_Spektren



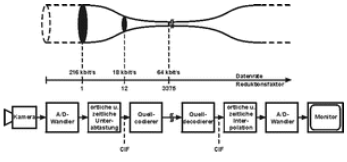
Allgemeine_Skalierung



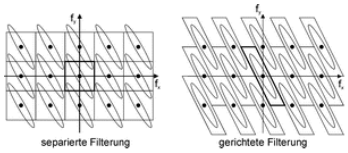
ASK_Modulation



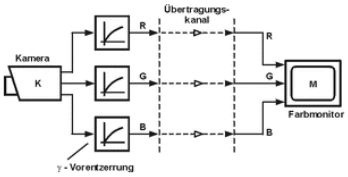
Blockschaltb_Standard_TV_Ger



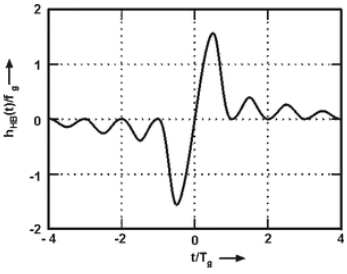
Flaschenhals



gerichtete_Filterung

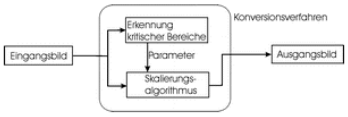


Grundprinzip_Farbdarstellung

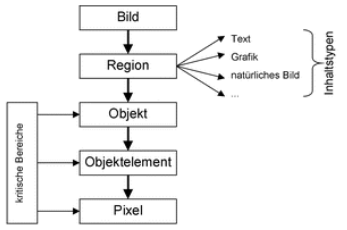


Impulsantwort

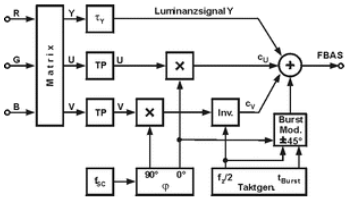
Kombinierte_Skalierung



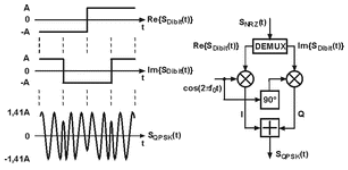
Konversionsverfahren



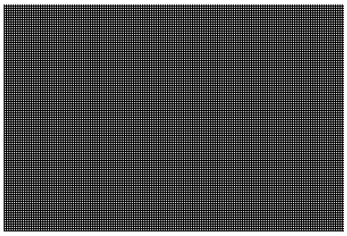
Nomenklatur



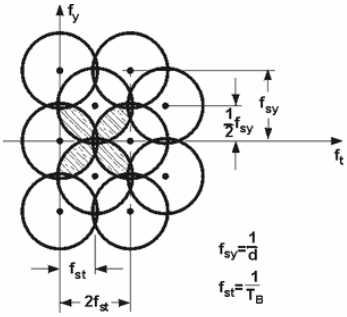
PAL-Farbencoder



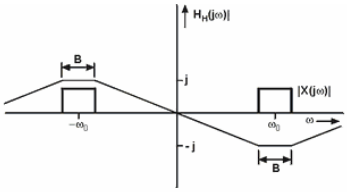
QPSK_Modulation



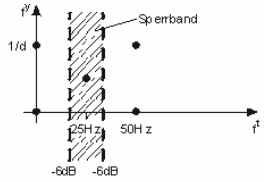
Raster



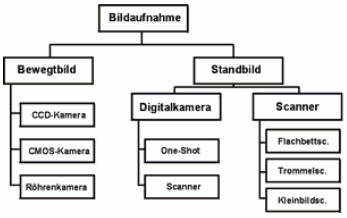
Spektren_bei_Zeilensprungabt



Spektrum_Hilberttrafo

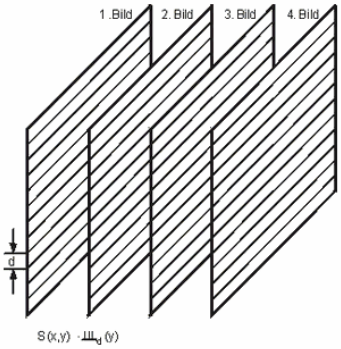


Sperrband



Struktugramm

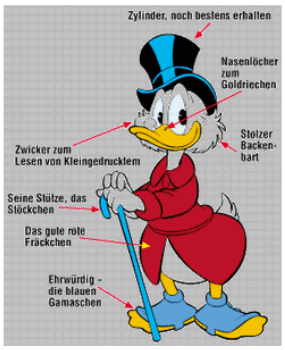
Unterabtastung



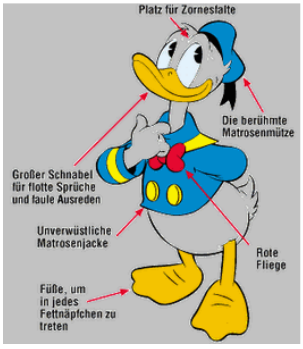
vert_zeitl_abtas

alien

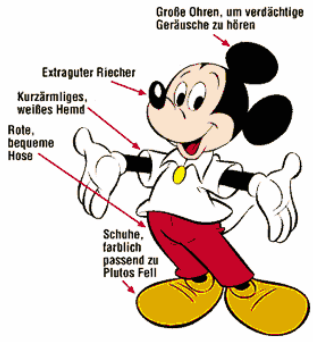
candle



dagobert



donald



micky

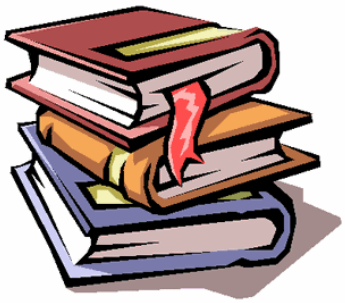
hase2

hase

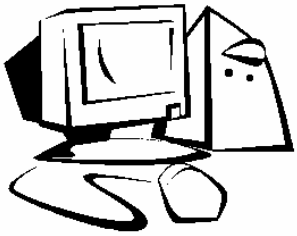
schiff

wizard

zigarett



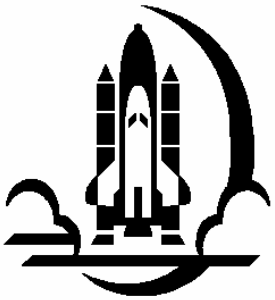
books



computer

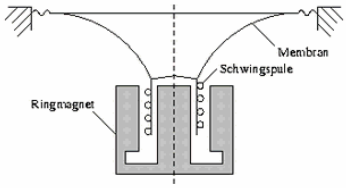
handy

man

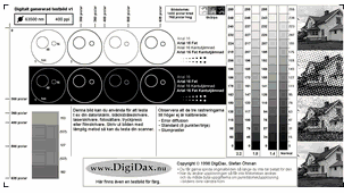


shuttle

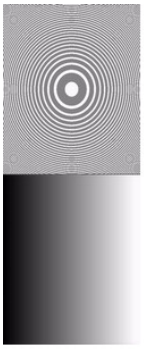
skyline



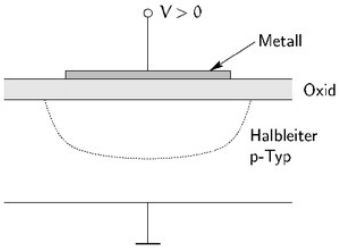
audio_bild_sw



grauskala



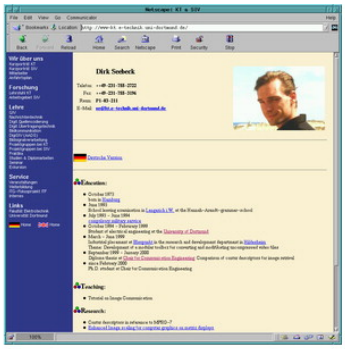
grey_tint



MOS_Kondensator

Fachschaft2

google



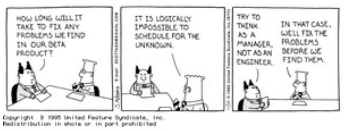
se

Banking

Bengen1

Bengen2

Bengen3



dilbert4

dilbert7

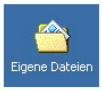
Fernsehn

Monitor

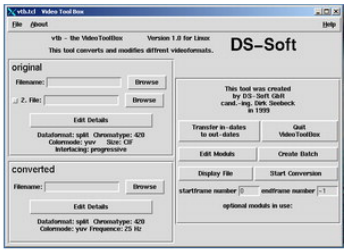
freeamp

Icons

matrox-Dvd

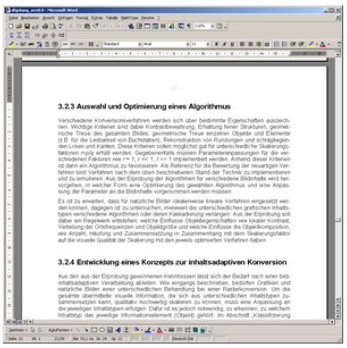


ordner



vtb

Word-Arbeitsleiste



Word-komplett

Word-Leiste-unten

zip

$$\beta^{0}(x)=\begin{cases}1 & \text{für } -0.5<\dfrac{x}{x_0}<0.5\\ 0.5 & \left|\dfrac{x}{x_0}\right|=0.5\\ 0 & \text{sonst}\end{cases}$$

Formel1

$$h_{ideal}(x,y)=\frac{\sin\left(\pi\,x/x_1\right)}{\pi\,x/x_1}\cdot\frac{\sin\left(\pi\,y/y_1\right)}{\pi\,y/y_1}$$

5

$$H_{ideal}\left(f_x,f_y\right)=f_{x_1}f_{y_1}\mathrm{R}_{f_{x_1}}\left(f_x\right)\mathrm{R}_{f_{y_1}}\left(f_x\right)$$

Formel2

$$H_{AR}(\Omega)=\frac{1}{1-\sum_{n=1}^{N}b_i\cdot e^{-jn\Omega}}$$

Formel3

$$g(x,y)=\left(\sum_{n_x,n_y=-\infty}^{\infty}b\left(n_x,n_y\right)\delta\left(x-n_x\cdot x_0,y-n_y\cdot y_0\right)\right)**h(x,y)$$

Formel4

Bildkommunikation I, Wintersemester 2000/2001

1. Einleitung, Grundlagen der Bildkommunikation
Einordnung der Vorlesung, Schwerpunkte
Blockschaltbild eines Übertragungssystems
Wahrnehmung von Licht und Farbe

2. Prinzipien von Bildkommunikationssystemen
Randbedingungen der Bewegtbildübertragung
Mehrdimensionale Abtastung
Bildformate
Bandbreiten und Datenraten

3. Bildaufnahme
CCD-Sensoren
Bewegtbildkameras
Standbildkameras
Scanner

Inhalt

Standart		B/G	D/K	I	L	M
		CCIR	ORT	Gr_Brit	Frankreich	FCC
		Mitteleuropa	Osteuropa			USA
Frequenzbereiche		VHF/UHF	VHF/UHF	VHF/UHF	UHF	VHF/UHF
Zeilen je Vollbild		625	625	625	625	525
Halbbildfrequenz	Hz	50	50	50	50	60 (59,98)
Zeilenfrequenz	Hz	15625	15625	15625	15625	15750 (15734,264)
Zeilen-Synchronisationsdauer	µs	4,7	4,7	4,7	4,7	5 (4,6)
Zeilen-Austastimpulsdauer	µs	12	12	12	12	10,8 (11)
Vordere Schwarzschulter	µs	1,5	1,5	1,5	1,5	1,9 (1,75)
Bild-Austastdauer	Zeilen	25	25	25	25	19 ... 21
Videobandbreite	MHz	5	6	5,5	6	4,2
HF-Kanalbandbreite	MHz	7(B) / 8(G)	8	8	8	6

Tabelle_mit_Graulinien

		x	y	z
EBU	R	0.64	0.33	0.03
	G	0.29	0.60	0.11
	B	0.15	0.06	0.79
	D65	0.313	0.329	0.358
FCC	R	0.67	0.33	0.0
	G	0.21	0.71	0.08
	B	0.14	0.08	0.78
	C	0.310	0.316	0.374

Tabelle_mit_Linien

	Europa	USA
Zeilenzahl:	**625**	**525**
Zeilendauer:	**64 µs**	**64 µs**
Zeilenfrequenz:	**15625 Hz**	**15750 Hz**
Hor. Austastlücke:	**12 µs**	**10,8 µs**
Bildfrequenz Vollbild:	**25 Hz**	**30 Hz**
Zeilensprung:	**2:1**	**2:1**
Halbbildfrequenz:	**50 Hz**	**60 Hz**
Halbbilddauer:	**20 ms**	**16,6 ms**
Vertikale Austastlücke:	**25 Zeilen**	**20 Zeilen**
Bildformat:	**4:3**	**4:3**
Grenzfrequenz Bildsignal:	**5 MHz**	**4,2 MHz**

Tabelle_ohne_Linien

Enhanced Image Scaling for Computer Grafics
Enhanced Image Scaling for Computer Grafics
Enhanced Image Scaling for Computer Grafics
Enhanced Image Scaling for Computer Grafics

Enhanced Image Scaling for Computer Grafics
Enhanced Image Scaling for Computer Grafics
Enhanced Image Scaling for Computer Grafics
Enhanced Image Scaling for Computer Grafics

Text

Verschiedene Konversionsverfahren werden sich über bestimmte Eigenschaften auszeichnen. Wichtige Kriterien sind dabei Kontrastbewahrung, Erhaltung feiner Strukturen, geometrische Treue des gesamten Bildes, geometrische Treue einzelner Objekte und Elemente (z.B. für die Lesbarkeit von Buchstaben), Rekonstruktion von Rundungen und schräglegenden Linien und Kanten. Diese Kriterien sollen möglichst gut für unterschiedliche Skalierungsfaktoren r=p/q erfüllt werden. Gegebenenfalls müssen Parameteranpassungen für die verschiedenen Faktoren wie r = 1, r << 1, r >> 1 implementiert werden. Anhand dieser Kriterien ist dann ein Algorithmus zu favorisieren. Als Referenz für die Bewertung der neuartigen Verfahren sind Verfahren nach dem oben beschriebenen Stand der Technik zu implementieren und zu simulieren. Aus der Erprobung der Algorithmen für verschiedene Bildinhalte wird hervorgehen, in welcher Form eine Optimierung des gewählten Algorithmus und eine Anpassung der Parameter an die Bildinhalte vorgenommen werden müssen.

Textblock

Verschiedene Konversionsverfahren werden sich über bestimmte Eigenschaften auszeichnen. Wichtige Kriterien sind dabei Kontrastbewahrung, Erhaltung feiner Strukturen, geometrische Treue des gesamten Bildes, geometrische Treue einzelner Objekte und Elemente (z.B. für die Lesbarkeit von Buchstaben), Rekonstruktion von Rundungen und schräglegenden Linien und Kanten. Diese Kriterien sollen möglichst gut für unterschiedliche Skalierungsfaktoren r=p/q erfüllt werden. Gegebenenfalls müssen Parameteranpassungen für die verschiedenen Faktoren wie r = 1, r << 1, r >> 1 implementiert werden. Anhand dieser Kriterien ist dann ein Algorithmus zu favorisieren. Als Referenz für die Bewertung der neuartigen Verfahren sind Verfahren nach dem oben beschriebenen Stand der Technik zu implementieren und zu simulieren. Aus der Erprobung der Algorithmen für verschiedene Bildinhalte wird hervorgehen, in welcher Form eine Optimierung des gewählten Algorithmus und eine Anpassung der Parameter an die Bildinhalte vorgenommen werden müssen.

Textblock_kursiv

3.2.3 Auswahl und Optimierung eines Algorithmus

Ueberschrift

Anhang B

Statistische Bildeigenschaften

Bild	Durchschnittliche Sättigung	Anteil schwarzer Pixel	Anteil weißer Pixel	Anteil grauer Pixel	Anteil voll-gesättigter Pixel	Anteil halb-gesättigter Pixel	Verwendete Helligkeitsstufen	Verwendete Sättigungsstufen	Verwendete Farbtonstufen	Verwendete HSV-166-Werte	Varianz der Intensität	Entropie der Intensität	Wechselhäufigkeit der Intensität	Entropie im HSV-166-Raum	Wechselhäufigkeit im HSV-166-Raum
akiyo	42.67	0.40	2.06	29.24	0.08	15.93	246	240	101	51	3344.43	7.37	78.57	3.01	8.86
boat	18.99	3.15	0.00	36.16	0.01	2.91	207	246	165	55	2047.59	6.98	89.40	2.17	15.98
bream	102.50	0.37	0.30	1.11	0.21	38.97	225	249	149	64	1862.75	6.51	58.76	3.39	10.94
carphone	19.13	0.79	7.05	20.08	0.00	7.36	244	194	125	44	3329.41	7.29	85.30	2.98	13.81
football	35.98	6.58	0.88	6.17	0.19	28.53	249	254	251	111	2356.06	7.07	89.45	3.43	27.20
foreman	26.22	0.78	6.46	24.16	0.02	2.69	237	236	105	54	2911.16	7.30	86.50	2.78	13.56
kerstin	26.48	0.00	4.11	19.47	0.00	0.68	183	158	157	40	3853.47	6.84	88.23	2.94	22.76
lena_c	88.77	0.00	0.01	0.07	0.26	58.97	197	242	114	55	2319.60	7.45	88.91	3.76	31.46
miss	33.01	0.35	0.00	2.41	0.00	18.01	217	221	84	41	863.79	5.68	79.37	2.49	12.73
news	27.22	0.26	1.97	24.91	0.28	7.24	246	256	116	68	4332.34	6.69	80.62	2.47	11.33
salesman	23.07	0.35	0.00	8.90	0.31	14.90	192	253	159	45	1204.29	6.84	89.54	2.74	19.17
sean	41.60	0.04	2.12	0.88	0.15	4.80	244	254	111	58	5307.31	6.81	68.07	3.37	18.27
stefan	37.46	0.11	5.40	8.42	0.00	1.53	236	209	157	79	2469.97	7.25	84.87	3.21	21.91
weather	137.20	0.24	9.85	1.51	31.36	25.97	255	256	233	134	5920.92	7.48	63.23	3.69	18.96
Autorennen1	36.20	7.92	0.26	4.46	2.04	28.18	256	255	184	98	1194.68	6.85	78.55	3.03	15.15
Autorennen2	80.43	0.96	0.00	0.52	0.30	63.76	256	252	152	56	1312.67	6.82	74.85	3.28	14.94
Autorennen3	64.52	0.24	0.17	0.04	3.71	11.95	252	256	105	46	2606.53	7.06	70.52	3.14	12.49
Flughafen	29.95	1.46	6.02	34.75	0.29	40.06	255	253	179	79	2408.06	6.43	49.08	2.69	7.64
Flughafen2	15.46	0.03	36.54	38.87	0.04	1.12	252	249	131	58	2780.71	6.55	33.93	1.80	4.17
Flughafen3	23.20	1.86	8.28	42.43	0.15	1.81	256	256	148	67	3772.75	6.18	23.37	1.68	3.24
Flughafen4	29.12	3.52	3.32	28.59	0.55	17.97	256	255	195	83	2646.47	7.48	76.26	3.50	15.32
Florian	14.45	0.56	9.45	41.73	0.02	0.27	256	240	127	54	3026.80	7.50	73.25	2.46	9.60
Fussball	47.85	0.98	3.52	15.52	0.45	5.31	239	251	133	64	3740.05	7.26	81.68	3.92	25.27
Football_VR	40.71	4.42	0.74	14.12	3.77	22.65	256	256	164	90	2534.50	7.47	84.43	4.11	25.83
Hockey	14.28	7.01	60.53	4.70	2.62	7.27	256	256	152	80	9222.08	6.06	73.47	2.11	13.30
Luftbild1	20.05	0.00	10.12	34.35	0.00	0.00	206	98	43	16	1631.02	6.84	67.70	1.38	6.18
Uni-Bochum	20.88	12.18	0.20	26.09	0.02	0.63	249	237	108	28	3134.51	7.24	70.67	2.51	9.75
blume	0.00	23.31	0.13	76.56	0.00	0.00	244	1	0	4	4044.78	7.49	81.84	1.82	7.91
kewgarden	0.00	8.10	0.00	91.90	0.00	0.00	214	1	0	4	2195.27	7.29	84.51	1.59	9.15
lena	0.00	0.00	0.02	99.98	0.00	0.00	216	1	0	4	2300.37	7.45	89.12	1.80	8.18
mobile	0.00	7.19	0.25	92.57	0.00	0.00	248	1	0	4	3389.17	6.95	79.05	1.57	4.24
teen	0.00	5.65	1.90	92.44	0.00	0.00	253	1	0	4	4140.52	7.76	81.31	1.97	6.84
3d_plot	11.13	0.00	62.55	4.25	0.00	0.15	119	133	45	19	5012.26	3.18	8.14	1.42	8.14
abtast_alias	19.11	0.90	86.68	0.00	1.01	6.90	2	5	3	6	2788.69	0.83	1.58	0.81	1.44
abtast_apertur	28.46	1.34	85.19	0.00	4.12	9.35	2	3	2	4	4756.92	0.82	2.47	0.72	2.03
bild_reimerR3	9.96	2.36	93.74	0.00	3.90	0.00	2	2	2	4	2942.41	0.44	4.22	0.44	4.22
ccd-ganz	7.74	6.84	81.10	8.17	2.17	1.71	6	3	2	6	4877.31	1.12	4.70	0.72	4.70
ccd-pixel_c	76.39	4.00	54.40	0.00	28.76	0.00	3	3	3	6	5202.37	2.01	3.79	2.01	3.79
kabelnetz	12.08	3.92	82.51	0.00	0.00	0.00	2	2	1	3	2459.40	0.80	3.50	0.80	3.50
mobilfunk_kan	33.45	7.81	76.81	0.00	15.37	0.00	3	2	2	4	6881.02	1.10	2.36	1.10	2.36
mulpic	100.21	7.74	9.50	25.15	51.85	4.93	256	239	81	64	5880.87	6.57	39.71	3.79	17.73
100Hz	0.00	0.00	93.44	6.56	0.00	0.00	3	1	0	2	3167.57	0.36	7.50	0.35	7.25
100Hz2	0.00	0.00	86.30	13.70	0.00	0.00	2	1	0	2	6145.24	0.58	5.20	0.58	5.20
100Hzplus	0.00	0.28	93.25	6.47	0.00	0.00	4	1	0	2	3288.83	0.39	7.83	0.36	7.45
1D_Spektren	0.00	0.00	89.67	10.33	0.00	0.00	2	1	0	2	4815.34	0.48	3.19	0.48	3.19
Allg_Skal	0.00	8.38	91.62	0.00	0.00	0.00	2	1	0	2	4993.58	0.42	1.83	0.42	1.83
ASK_Modula	0.00	0.00	93.29	6.71	0.00	0.00	2	1	0	2	3252.30	0.35	1.41	0.35	1.41
Blockschaltb	0.00	0.00	89.35	10.65	0.00	0.00	2	1	0	2	4945.08	0.49	3.08	0.49	3.08
Flaschenhals	0.00	0.00	90.38	9.62	0.00	0.00	2	1	0	2	4520.82	0.46	3.09	0.46	3.09
gerich_Filt	0.00	9.04	90.96	0.00	0.00	0.00	2	1	0	2	5346.45	0.44	2.32	0.44	2.32
Grundp_Farb	0.00	0.00	91.66	8.34	0.00	0.00	2	1	0	2	3975.19	0.41	2.42	0.41	2.42
Impulsantw	0.00	0.00	93.76	6.24	0.00	0.00	2	1	0	2	3043.44	0.34	2.25	0.34	2.25
Komb_Skal	0.00	5.87	94.13	0.00	0.00	0.00	2	1	0	2	3593.48	0.32	1.23	0.32	1.23
Konverfahren	0.00	5.52	94.48	0.00	0.00	0.00	2	1	0	2	3391.99	0.31	1.75	0.31	1.75
Nomenklatur	0.00	4.44	95.56	0.00	0.00	0.00	2	1	0	2	2756.26	0.26	0.86	0.26	0.86
PAL-Farbenc	0.00	0.00	89.54	10.46	0.00	0.00	2	1	0	2	4869.10	0.48	2.73	0.48	2.73
QPSK_Modula	0.00	0.00	91.66	8.34	0.00	0.00	2	1	0	2	3975.23	0.41	2.42	0.41	2.42
Raster	0.00	75.00	25.00	0.00	0.00	0.00	2	1	0	2	12192.19	0.81	49.75	0.81	49.75
Spek_Zeilen	0.00	0.00	90.46	9.54	0.00	0.00	2	1	0	2	4484.61	0.45	3.63	0.45	3.63
Spek_Hilbert	0.00	0.00	95.79	4.21	0.00	0.00	2	1	0	2	2097.21	0.25	1.41	0.25	1.41

Bild	Durchschnittliche Sättigung	Anteil schwarzer Pixel	Anteil weißer Pixel	Anteil grauer Pixel	Anteil voll-gesättigter Pixel	Anteil halb-gesättigter Pixel	Verwendete Helligkeitsstufen	Verwendete Sättigungsstufen	Verwendete Farbtonstufen	Verwendete HSV-166-Werte	Varianz der Intensität	Entropie der Intensität	Wechselhäufigkeit der Intensität	Entropie im HSV-166-Raum	Wechselhäufigkeit im HSV-166-Raum
Sperrband	0.00	0.00	94.33	5.67	0.00	0.00	2	1	0	2	2782.16	0.31	5.64	0.31	5.64
Struktugramm	0.00	9.44	86.29	4.27	0.00	0.00	159	1	0	4	6129.39	1.08	5.44	0.66	3.97
Unterabtastung	0.00	8.20	91.80	0.00	0.00	0.00	2	1	0	2	4894.02	0.41	1.72	0.41	1.72
vert_abtast	0.00	0.00	83.61	16.39	0.00	0.00	2	1	0	2	7122.72	0.64	9.05	0.64	9.05
alien	55.70	12.55	52.87	0.00	34.58	0.00	16	4	3	8	9743.86	1.94	21.55	1.71	8.47
candle	21.73	8.26	75.70	1.47	10.75	3.11	148	133	37	23	7589.49	2.09	16.81	1.37	10.20
dagobert	28.64	5.23	0.60	76.74	13.27	3.25	58	37	16	23	3294.93	2.09	14.29	1.68	10.62
donald	33.02	3.01	0.52	79.86	15.20	0.39	57	36	16	25	2130.52	1.84	11.92	1.45	9.25
micky	24.80	8.58	68.50	6.73	9.98	0.66	16	14	5	9	7122.18	1.98	15.24	1.42	10.45
hase2	8.96	8.63	56.43	22.83	0.04	2.45	157	79	7	26	7037.46	2.46	3.20	1.85	1.96
hase	14.85	6.87	53.47	0.49	0.00	0.00	153	84	15	13	5302.32	2.71	7.41	1.64	2.53
schiff	19.73	13.70	64.40	0.01	0.00	1.23	3	4	2	5	7537.00	1.55	4.71	1.55	4.71
wizard	23.25	7.46	70.20	1.80	3.33	7.74	29	18	4	13	5885.10	1.89	12.49	1.60	11.24
zigarett	0.55	11.49	88.33	0.00	0.12	0.05	8	6	2	5	6624.76	0.73	10.46	0.54	9.59
books	31.21	21.79	32.59	0.00	0.00	17.10	11	14	5	12	9074.05	2.96	4.13	2.48	4.09
computer	0.00	20.14	79.86	0.00	0.00	0.00	2	1	0	2	10459.82	0.72	4.05	0.72	4.05
handy	0.00	30.39	69.61	0.00	0.00	0.00	2	1	0	2	13755.13	0.89	2.76	0.89	2.76
man	90.87	21.16	19.70	0.55	25.48	6.73	256	202	9	21	7768.30	3.22	10.51	2.64	6.74
shuttle	0.00	26.96	73.04	0.00	0.00	0.00	12	1	0	2	12805.59	0.85	2.85	0.84	2.79
skyline	23.43	38.69	14.76	0.00	0.00	0.00	4	4	0	2	12553.68	2.11	6.36	0.96	4.43
abtast_alias	0.00	0.90	86.68	12.42	0.00	0.00	7	1	0	3	2741.82	0.83	1.58	0.48	1.51
abtast_apertur	0.00	1.34	85.19	13.47	0.00	0.00	5	1	0	3	4645.68	0.82	2.47	0.74	2.37
audio_bild	0.00	4.20	85.19	10.61	0.00	0.00	3	1	0	3	2862.06	0.73	4.54	0.73	4.54
bild_reimerR3	0.00	2.36	93.74	3.90	0.00	0.00	4	1	0	3	2913.73	0.44	4.22	0.40	4.21
ccd-pixel	0.00	4.00	54.40	41.59	0.00	0.00	6	1	0	4	5255.46	2.01	3.79	1.35	3.79
ccd_ganz	0.00	6.84	81.10	12.06	0.00	0.00	8	1	0	4	4871.35	1.12	4.70	0.68	4.70
grauskala	0.00	23.46	64.20	12.34	0.00	0.00	256	1	0	4	11724.58	2.29	12.39	1.24	9.24
grey_tint	3.00	4.29	17.06	77.36	0.00	0.00	250	50	0	4	4468.66	7.47	45.68	1.90	26.87
kabelnetz	0.00	3.92	82.51	13.57	0.00	0.00	3	1	0	2	2459.40	0.80	3.50	0.24	3.50
mobilfunk_kan	0.00	7.81	76.81	15.37	0.00	0.00	4	1	0	3	6905.05	1.10	2.36	1.01	2.23
MOS_Kondens	0.00	1.94	89.34	8.72	0.00	0.00	143	1	0	4	1371.76	0.79	2.95	0.28	1.44
mulpic	0.00	20.12	8.86	71.03	0.00	0.00	63	1	0	4	5796.13	5.34	50.30	1.83	12.46
Fachschaft2	53.51	68.52	4.92	0.18	25.66	0.55	33	73	26	21	3095.67	1.46	12.49	1.40	12.08
google	6.68	0.73	92.11	3.01	1.78	2.03	82	128	32	47	1903.73	1.03	5.36	0.52	3.34
se	43.52	1.85	5.84	12.69	0.87	16.43	54	185	65	54	5428.48	2.49	8.14	1.50	6.26
Banking	0.00	2.59	84.66	12.74	0.00	0.00	8	1	0	4	3350.40	1.16	15.64	0.77	11.25
Bengen1	0.73	0.00	86.82	9.72	0.00	0.00	16	9	0	4	2936.81	1.15	14.91	0.67	9.91
Bengen2	0.82	5.92	80.87	13.21	0.00	0.00	8	8	0	4	4501.13	1.21	14.06	0.82	9.99
Bengen3	2.05	22.00	63.34	14.66	0.00	0.00	5	5	0	4	10105.00	1.54	15.34	1.27	12.08
dilbert4	0.00	4.49	84.34	11.18	0.00	0.00	8	1	0	4	4436.58	1.05	15.78	0.78	12.95
dilbert7	0.00	5.37	80.70	13.92	0.00	0.00	8	1	0	4	5136.89	1.23	19.89	0.90	16.15
Fernsehn	0.00	7.18	77.76	15.05	0.00	0.00	8	1	0	4	6039.17	1.47	17.60	1.02	12.43
Monitor	0.00	4.59	80.03	15.37	0.00	0.00	8	1	0	4	4525.33	1.41	18.87	0.94	13.74
freeamp	8.34	1.86	0.00	69.74	0.00	0.00	173	27	0	4	2338.22	5.16	24.60	1.51	8.20
Icons	105.48	0.41	2.29	0.56	1.59	94.10	34	38	40	54	747.54	0.71	5.33	0.61	4.76
matrox-Dvd	3.16	20.98	13.62	63.53	1.53	0.18	255	65	12	14	6998.16	6.31	51.83	2.11	12.66
ordner	101.76	0.10	4.17	0.00	2.53	85.83	8	11	10	18	2090.55	1.07	11.11	0.99	10.21
vtb	3.98	4.72	4.58	86.48	0.11	2.24	185	190	34	21	2945.95	1.51	7.68	0.92	5.77
Word-Leiste	13.47	2.04	33.06	54.21	0.48	4.89	23	62	24	25	3102.00	2.41	11.14	1.17	6.15
Word-komplett	3.85	6.42	67.32	24.41	0.20	0.74	138	140	17	20	4873.10	1.62	11.02	0.90	7.99
Word-unten	10.62	2.39	9.70	87.09	0.82	0.00	6	3	6	13	2406.23	1.27	6.57	0.82	5.21
zip	102.32	2.08	3.98	1.00	10.74	81.18	14	11	10	16	2080.56	1.46	12.99	1.27	12.55
Formel1	0.00	2.22	97.78	0.00	0.00	0.00	2	1	0	2	1409.47	0.15	0.78	0.15	0.78
Formel2	0.00	5.16	94.84	0.00	0.00	0.00	2	1	0	2	3184.75	0.29	1.80	0.29	1.80
Formel3	0.00	5.07	94.93	0.00	0.00	0.00	2	1	0	2	3127.69	0.29	1.76	0.29	1.76
Formel4	0.00	3.81	96.19	0.00	0.00	0.00	2	1	0	2	2382.77	0.23	1.53	0.23	1.53
Inhalt	0.00	10.16	89.84	0.00	0.00	0.00	2	1	0	2	5934.87	0.47	2.80	0.47	2.80
Tab_Graulinien	0.00	7.88	87.03	5.09	0.00	0.00	3	1	0	2	4780.66	0.68	5.97	0.40	4.09
Tab_Linien	0.00	6.98	93.02	0.00	0.00	0.00	2	1	0	2	4219.90	0.37	2.52	0.37	2.52
Tab_ohne_Lin	0.00	7.63	92.37	0.00	0.00	0.00	2	1	0	2	4582.38	0.39	2.29	0.39	2.29
Text	0.00	7.17	92.83	0.00	0.00	0.00	2	1	0	2	4326.00	0.37	6.16	0.37	6.16
Textblock	0.00	12.34	87.66	0.00	0.00	0.00	2	1	0	2	7035.32	0.54	12.35	0.54	12.35
Textb_kursiv	0.00	12.35	87.65	0.00	0.00	0.00	2	1	0	2	7040.41	0.54	12.80	0.54	12.80
Ueberschrift	0.00	23.51	76.49	0.00	0.00	0.00	2	1	0	2	11694.34	0.79	9.49	0.79	9.49

Anhang C

Analyse der Wavelet-Koeffizienten

Bild	Laplace-Match $\overline{\chi}^2$	Konzentrationsmaß L
BunteBilder/akiyo	1.255819	0.489968
BunteBilder/boat	0.221143	0.167274
BunteBilder/bream	4.845707	0.569981
BunteBilder/carphone	1.016150	0.303099
BunteBilder/football	0.419375	0.111956
BunteBilder/foreman	0.837625	0.247413
BunteBilder/kerstin	0.076808	0.266485
BunteBilder/lena_c	0.249148	0.170455
BunteBilder/miss	0.287218	0.430974
BunteBilder/news	1.706162	0.427741
BunteBilder/salesman	0.329541	0.142094
BunteBilder/sean	3.414900	0.375541
BunteBilder/stefan	1.257330	0.213512
BunteBilder/weather	8.168571	0.422548
VR/Autorennen1	0.876929	0.299869
VR/Autorennen2	0.666105	0.333721
VR/Autorennen3	1.628442	0.315423
VR/Flughafen	8.080120	0.487538
VR/Flughafen2	9.454334	0.693918
VR/Flughafen3	13.802838	0.729108
VR/Flughafen4	1.873216	0.231638
VR/Florian	1.806694	0.290528
VR/Fussball	0.897456	0.136947
VR/Football_VR	0.838322	0.129207
VR/Hockey	2.496464	0.297523
VR/Luftbild1	1.875474	0.328717
VR/Uni-Bochum	2.499578	0.392618
GrauBilder/blume	1.344242	0.284571
GrauBilder/kewgarden	0.618264	0.232195
GrauBilder/mobile	1.271167	0.362635
GrauBilder/teen	0.823226	0.321372
BunteGrafiken/3d_plot	85.412412	0.911562
BunteGrafiken/abtastung_mit_und_ohne_alias	58.373906	1.000000
BunteGrafiken/abtastung_mit_wiedergabeapertur	65.624873	1.000000

Bild	Laplace-Match $\overline{\chi}^2$	Konzentrationsmaß L
BunteGrafiken/bild_reimerR3	97.062174	1.000000
BunteGrafiken/ccd-ganz	161.280602	1.000000
BunteGrafiken/ccd-pixel_c	91.550740	1.000000
BunteGrafiken/kabelnetz	107.968408	1.000000
BunteGrafiken/mobilfunk_kanal	81.226598	1.000000
BunteGrafiken/mulpic	14.106450	0.518493
SWGrafik/100Hz	187.179869	1.000000
SWGrafik/100Hz2	196.155213	1.000000
SWGrafik/100Hzplus	179.010544	1.000000
SWGrafik/1D_Spektren	238.580879	1.000000
SWGrafik/Allgemeine_Skalierung	77.969184	1.000000
SWGrafik/ASK_Modulation	75.249142	1.000000
SWGrafik/Blockschaltb_Standard_TV_Ger	85.307132	1.000000
SWGrafik/Flaschenhals	299.048543	1.000000
SWGrafik/gerichtete_Filterung	344.555779	1.000000
SWGrafik/Grundprinzip_Farbdarstellung	80.766494	1.000000
SWGrafik/Impulsantwort	73.843305	1.000000
SWGrafik/Kombinierte_Skalierung	69.168557	1.000000
SWGrafik/Konversionsverfahren	79.390178	1.000000
SWGrafik/Nomenklatur	64.710999	1.000000
SWGrafik/PAL-Farbencoder	82.697017	1.000000
SWGrafik/QPSK_Modulation	83.012391	1.000000
SWGrafik/Raster	825.303082	1.000000
SWGrafik/Spektren_bei_Zeilensprungabtastung	92.225775	1.000000
SWGrafik/Spektrum_Hilberttrafo	64.242868	1.000000
SWGrafik/Sperrband	107.213106	1.000000
SWGrafik/Struktugramm	84.964998	0.996996
SWGrafik/Unterabtastung	76.577350	1.000000
SWGrafik/vert_zeitl_abtas	150.836806	1.000000
Cliparts/alien	54.689549	1.000000
Cliparts/candle	59.642579	0.697510
Scans/dagobert	54.824494	0.833915
Cliparts/donald	54.321395	0.886628
Cliparts/micky	63.688233	0.935232
Cliparts/hase2	36.330331	0.955335
Cliparts/hase	31.390996	0.855592
Cliparts/schiff	84.731420	1.000000
Cliparts/wizard	81.626769	0.999925
Cliparts/zigarett	178.427916	1.000000
Cliparts_fertig/books	78.490403	1.000000
Cliparts_fertig/computer	113.393492	1.000000
Cliparts_fertig/handy	99.031660	1.000000
Cliparts_fertig/man	40.242208	0.810220
Cliparts_fertig/shuttle	94.229167	1.000000
Cliparts_fertig/skyline	143.675155	1.000000
GrauGrafik/audio_bild_sw	107.582144	1.000000
GrauGrafik/grauskala	160.238399	0.912854
GrauGrafik/grey_tint	11.651182	0.299807
GrauGrafik/MOS_Kondensator	67.934192	0.990487

Bild	Laplace-Match $\overline{\chi}^2$	Konzentrationsmaß L
Html/Fachschaft2	182.465054	0.998356
Html/google	71.336076	0.942011
Html/se	114.507693	0.964029
Scans/Banking	64.498342	0.976564
Scans/Bengen1	58.726236	0.869392
Scans/Bengen2	65.789423	0.993923
Scans/Bengen3	71.604124	0.987973
Scans/dilbert4	99.088657	1.000000
Scans/dilbert7	91.558536	1.000000
Scans/Fernsehn	63.607667	0.968561
Scans/Monitor	66.795423	0.976097
Screenshots/freeamp	31.406263	0.654457
Screenshots/Icons	51.416229	0.960478
Screenshots/matrox-Dvd	13.879575	0.422812
Screenshots/ordner	60.611700	1.000000
Screenshots/vtb	155.921887	0.999677
Screenshots/Word-Arbeitsleiste	78.880237	0.959923
Screenshots/Word-komplett	176.687124	0.998382
Screenshots/Word-Leiste-unten	98.141534	1.000000
Screenshots/zip	79.427895	1.000000
Text/Formel1	86.577834	1.000000
Text/Formel2	84.597771	1.000000
Text/Formel3	81.758589	1.000000
Text/Formel4	82.235479	1.000000
Text/Inhalt	319.072875	1.000000
Text/Tabelle_mit_Graulinien	199.276625	1.000000
Text/Tabelle_mit_Linien	340.194626	1.000000
Text/Tabelle_ohne_Linien	88.265902	1.000000
Text/Text	189.145333	1.000000
Text/Textblock	212.030139	1.000000
Text/Textblock_kursiv	210.331756	1.000000
Text/Ueberschrift	193.599806	1.000000

Anhang D

Source Code

D.1 BmpFunc.cpp

```
// Hilfsfunktionen zum Lesen und Schreiben von BMP-Dateien

#include "bmpfunc.h"

bool open_bmp(char *filename, int &xsize, int &ysize, int &picsize,
              int &padding, unsigned short *header, rgb_pixel **data)
{
    FILE *bmp = fopen(filename,"rb");
    if (!bmp)
    {
        printf("Konnte Eingabe-Datei nicht oeffnen.");
        return false;
    }

    fread(header,54,1,bmp);

    if (header[0]!='MB' || header[14]!=24 || header[15]!=0)
    {
        printf("Falsches Dateiformat.");
        return false;
    }

    xsize = header[9];
    ysize = header[11];
    picsize = xsize * ysize;

    if (xsize<20 || ysize<20)
    {
        printf("Bisschen klein, hmmm?");
        return false;
    }

    *data = new rgb_pixel[picsize];
```

```
    fseek(bmp, *(unsigned int *)(header+5), SEEK_SET);

    padding = xsize & 3;    //stimmt nur bei 24 bpp!!
    for (int i=0; i<ysize; i++)
    {
        fread(&(*data)[i*xsize],xsize,3,bmp);
        if (padding) fseek(bmp, padding, SEEK_CUR);
    }
    fclose(bmp);

    return true;
}

void write_bmp(char *filename, int xsize, int ysize, int padding,
               unsigned short *header, rgb_pixel *data)
{
    FILE *bmpout = fopen(filename,"wb");
    fwrite(header,54,1,bmpout);

    int xsize_padded = xsize*3+padding;
    for (int i=0; i<ysize; i++)
        fwrite(&data[i*xsize],xsize_padded,1,bmpout);

    fclose(bmpout);
}
```

D.2 BmpFunc.h

```
#include <stdio.h>
#define M_PI 3.14159265358979323846264338327

struct rgb_pixel {
    unsigned char b;
    unsigned char g;
    unsigned char r;
};

struct float_pixel {
    float b;
    float g;
    float r;
};

bool open_bmp(char *filename, int &xsize, int &ysize, int &picsize,
              int &padding, unsigned short *header, rgb_pixel **data);
void write_bmp(char *filename, int xsize, int ysize, int padding,
               unsigned short *header, rgb_pixel *data);
```

D.3 Scale.cpp

```
// Beim Kompilieren gepatchte Library zum korrekten Runden anstelle
// von Abschneiden bei der Fließkomma/Int - Konvertierung verwenden!!

// Skaliert ein Bild mit Hilfe verschiedener Skalieralgorithmen
// Kommandozeilenparameter:
// 1. Dateiname eines Bildes im Format 24Bit BMP
// 2. Dateiname eines Bildes für die Ausgabe (im selben Format)
// 3. Skalieralgorithmus (optional)
//    N=Nearest Neighbour, A=Averaging, B=Bilinear/Dreieck,
//    R=TP mit Rechteck-Fenster, H=TP mit Hann-Fenster,
//    L=TP mit Lanczos3-Fenster (default)
// 4. Skalierfaktor (optional, default 0.5)

#include <math.h>
#include "bmpfunc.h"

bool average(double pos, double &value)
{
    if (pos < 0.5 && -pos <= 0.5) { value=1; return false; }
    return true;
}

bool bilinear(double pos, double &value)
{
    if (pos < 0) pos = -pos;
    if (pos < 1) { value = 1 - pos; return false; }
    return true;
}

#define iwindow 100

bool ideal_tp(double pos, double &value)
{
    if (pos == 0)
    {
        value = 1;
        return false;
    }
    else if (pos <= iwindow && -pos <= iwindow)
    {
        pos*=M_PI;
        value = sin(pos)/pos;
        return false;
    }
    return true;
}
```

```
#define hwindow 3

bool hann_tp(double pos, double &value)
{
    if (pos == 0)
    {
        value = 1;
        return false;
    }
    else if (pos <= hwindow && -pos <= hwindow)
    {
        pos*=M_PI;
        double si = sin(pos)/pos;
        double ha = (1+cos(pos/hwindow))/2;
        value = si * ha;
        return false;
    }
    return true;
}

#define lwindow 3

bool lanczos(double pos, double &value)
{
    if (pos == 0)
    {
        value = 1;
        return false;
    }
    else if (pos <= lwindow && -pos <= lwindow)
    {
        pos*=M_PI;
        double si = sin(pos)/pos;
        pos/=lwindow;
        double la = sin(pos)/pos;
        value = si * la;
        return false;
    }
    return true;
}

void main(int argc,char *argv[])
{
    if (argc < 3)
    {
        printf("SCALE <BMP in> <BMP out> [N/A/B/R/H/L] [factor]");
        return;
    }

    bool (*filter)(double, double &);
    double factor = 0.5;
```

```
bool scaledown = true;

if (argc >= 4)
{
    if (argc >= 5)
        sscanf(argv[4],"%lf",&factor);

    switch (*argv[3] | 0x20)
    {
        case 'n': filter = average; scaledown=false; break;
        case 'a': filter = average; break;
        case 'b': filter = bilinear; break;
        case 'h': filter = hann_tp; break;
        case 'r': filter = ideal_tp; break;
        default : filter = lanczos;
    }
}
scaledown = scaledown && (factor < 1);
int xsize, ysize, picsize, padding;
unsigned short buf[27];
rgb_pixel *image;
float_pixel *tmp_pict;

if (!open_bmp(argv[1], xsize,ysize,picsize, padding, buf, &image))
    return;

int xsize_new = xsize*factor, ysize_new = ysize*factor;

tmp_pict = new float_pixel[xsize_new*ysize];

for (int y=0; y<ysize; y++)
    for (int x=0; x<xsize_new; x++)
    {
        double xorig=x/factor;
        int xorig_int=floor(xorig), filter_pos=0;
        double filterbase=(xorig-xorig_int);
        if (scaledown) filterbase*=factor;
        rgb_pixel *pix;

        bool filter1_ende=false, filter2_ende=false;
        double newr=0,newg=0,newb=0;

        do
        {
            double filter_value;
            if (!filter1_ende && filter_pos)
            {                          // folgende Pixel
                if ( !(filter1_ende = filter(filterbase -
                       (scaledown ? filter_pos*factor : filter_pos),
                       filter_value))
                    && filter_value != 0)
```

```
                {
                    if (scaledown) filter_value*=factor;
                    int pos = xorig_int + filter_pos;
                    pix=&image[y*xsize +
                               (pos < xsize ? pos : xsize-1)];
                    newr += pix->r * filter_value;
                    newg += pix->g * filter_value;
                    newb += pix->b * filter_value;
                }
            }
            if (!filter2_ende)
            {   // [ggf. passender und] vorhergehende Pixel
                if ( !(filter2_ende = filter(filterbase +
                      (scaledown ? filter_pos*factor : filter_pos),
                      filter_value))
                    && filter_value != 0)
                {
                    if (scaledown) filter_value*=factor;
                    int pos = xorig_int - filter_pos;
                    pix=&image[y*xsize + (pos >= 0 ? pos : 0)];
                    newr += pix->r * filter_value;
                    newg += pix->g * filter_value;
                    newb += pix->b * filter_value;
                }
            }
            filter_pos++;
        } while (!filter1_ende || !filter2_ende);

        float_pixel *fpix = &tmp_pict[y*xsize_new + x];

        fpix->r = newr;
        fpix->g = newg;
        fpix->b = newb;
    }
delete[] image;
image = new rgb_pixel[xsize_new*ysize_new];

for (int x=0; x<xsize_new; x++)
    for (int y=0; y<ysize_new; y++)
    {
        double yorig=y/factor;
        int yorig_int=floor(yorig), filter_pos=0;
        double filterbase=(yorig-yorig_int);
        if (scaledown) filterbase*=factor;
        float_pixel *pix;
        bool filter1_ende=false, filter2_ende=false;
        double newr=0,newg=0,newb=0;
        do
        {
            double filter_value;
            if (!filter1_ende && filter_pos)
```

```
            {                          // folgende Pixel
                if ( !(filter1_ende = filter(filterbase -
                      (scaledown ? filter_pos*factor : filter_pos),
                      filter_value))
                    && filter_value != 0)
                {
                    if (scaledown) filter_value*=factor;
                    int pos = yorig_int + filter_pos;
                    pix=&tmp_pict[(pos < ysize ? pos : ysize-1)
                                  *xsize_new + x];
                    newr += pix->r * filter_value;
                    newg += pix->g * filter_value;
                    newb += pix->b * filter_value;
                }
            }
            if (!filter2_ende)
            {   // [ggf. passender und] vorhergehende Pixel
                if ( !(filter2_ende = filter(filterbase +
                      (scaledown ? filter_pos*factor : filter_pos),
                      filter_value))
                    && filter_value != 0)
                {
                    if (scaledown) filter_value*=factor;
                    int pos = yorig_int - filter_pos;
                    pix=&tmp_pict[(pos >= 0 ? pos : 0)
                                  *xsize_new + x];
                    newr += pix->r * filter_value;
                    newg += pix->g * filter_value;
                    newb += pix->b * filter_value;
                }
            }
            filter_pos++;
        } while (!filter1_ende || !filter2_ende);

        rgb_pixel *rpix = &image[y*xsize_new + x];

        int tmp_int, clip=255;
        tmp_int=newr;
        if (tmp_int<0) tmp_int=-tmp_int;
        if (tmp_int>clip) tmp_int=clip;
        rpix->r = tmp_int;
        tmp_int=newg;
        if (tmp_int<0) tmp_int=-tmp_int;
        if (tmp_int>clip) tmp_int=clip;
        rpix->g = tmp_int;
        tmp_int=newb;
        if (tmp_int<0) tmp_int=-tmp_int;
        if (tmp_int>clip) tmp_int=clip;
        rpix->b = tmp_int;
    }
```

```
    delete[] tmp_pict;

    buf[9]=xsize_new;
    buf[11]=ysize_new;

    write_bmp(argv[2], xsize_new, ysize_new, xsize_new&3, buf, image);

    delete[] image;
}
```

D.4 ProjProf.cpp

```
// Zeichnet projection profiles und X-Y-cuts in ein Bild ein
// Dimensionen und Position der projection profiles muessen fuer das
// jeweilige Bild angepasst werden!
// Kommandozeilenparameter:
// Dateiname eines Bildes im Format 24Bit BMP, schwarz-weiss

#include "bmpfunc.h"

void main(int argc,char *argv[])
{
    if (argc != 3)
    {
        printf("Bitte Dateinamen für Ein- und Ausgabe angeben.");
        return;
    }

    int xsize, ysize, picsize, padding;
    unsigned short buf[27];
    rgb_pixel *data;

    if (!open_bmp(argv[1], xsize, ysize, picsize, padding, buf, &data))
        return;

    int anzahl[882];
    for (int i=0; i<882; i++) anzahl[i]=0;

    for (int y=895-14; y>=895-755; y--) {
        anzahl[y] = 0;
        for (int x=0; x<=620; x++)
            if (data[y*xsize+x].r==0) anzahl[y]++;
        anzahl[y]/=5;
        if (anzahl[y]>86) anzahl[y]=86;
        for (x=0; x<=anzahl[y]; x++)
        {
            int pos = y*xsize+635+x;
            data[pos].r=data[pos].g=data[pos].b=0;
        }
```

```
    }

    for (y=895-14; y>=895-755; y--) {
        if (anzahl[y-3] >= anzahl[y]*2 || anzahl[y+3] >= anzahl[y]*2)
        {
            for (int x=0; x<=720; x+=3)
            {
                int pos = y*xsize+x;
                data[pos].r=data[pos].g=data[pos].b=0;
            }
            y-=15;
        }
    }

    for (i=0; i<620; i++) anzahl[i]=0;

    for (int x=12; x<=615; x++) {
        anzahl[x] = 0;
        for (int y=895-14; y>=895-755; y--)
            if (data[y*xsize+x].r==0) anzahl[x]++;
        anzahl[x]/=5;
        if (anzahl[x]>86) anzahl[x]=86;
        for (y=0; y<=anzahl[x]; y++)
        {
            int pos = (895-861+y)*xsize+x;
            data[pos].r=data[pos].g=data[pos].b=0;
        }
    }

    for (x=12; x<=615; x++) {
        if (anzahl[x-3]>=anzahl[x]*1.4 || anzahl[x+3]>=anzahl[x]*1.4)
        {
            for (y=895-14; y>=895-861; y-=3)
            {
                int pos = y*xsize+x;
                data[pos].r=data[pos].g=data[pos].b=0;
            }
            x+=8;
        }
    }

    write_bmp(argv[2], xsize, ysize, padding, buf, data);

    delete[] data;
}
```

D.5 Bildmess.cpp

```
// Ermittelt verschiedene statistische Bildeigenschaften wie Anteile
// bestimmter Farben und Sättigungen oder Entropie und Wechselmass
// und hängt die Ausgabe der Ergebnisse an die Datei "OUT.TXT" an
// Kommandozeilenparameter:
// Dateiname eines Bildes im Format 24Bit BMP

#include <math.h>
#include "bmpfunc.h"

#define LOG2 0.3010299956639811952137

void main(int argc,char *argv[])
{
    if (argc != 2)
    {
        printf("Bitte Dateinamen des Bildes angeben.");
        return;
    }

    int xsize, ysize, picsize, padding;
    unsigned short buf[27];
    rgb_pixel *data;

    if (!open_bmp(argv[1], xsize, ysize, picsize, padding, buf, &data))
        return;

    unsigned char *data_hsv = new unsigned char[picsize];
    unsigned char *data_intens = new unsigned char[picsize];

    double fake_sat_avg=0, intens_avg=0;
    int anteil_schwarz=0,anteil_weiss=0,anteil_grau=0,
        anteil_vollsat=0,anteil_halbsat=0;
    bool is_hell[256], is_sat[256], is_hue[256];
    int intens_anz[256], hsv_anz[166];

    for (i=0; i<256; i++) is_hell[i]=is_sat[i]=is_hue[i]=
                          intens_anz[i]=0;
    for (i=0; i<166; i++) hsv_anz[i]=0;

    for (int pos=0; pos<picsize; pos++)
    {
        int b,g,r,min,max,dif,intens;
        int br_quant, sat_quant, hue_quant, hsv;
        double sat,hue,b1,g1,r1;

        r = data[pos].r;
        g = data[pos].g;
        b = data[pos].b;
```

```
intens = (r*3 + g*6 + b)/10;
data_intens[pos] = intens;
intens_anz[intens]++;
intens_avg += intens;

max = b>g ? b:g; if (r>max) max=r;
min = b<g ? b:g; if (r<min) min=r;

dif = max-min;
fake_sat_avg+=dif;

if (max!=0) sat = double(dif)/max; else sat = 0;

is_hell[max]=true;
sat_quant = sat*255.99999;
is_sat[sat_quant]=true;

if (max<25)
    anteil_schwarz++;
else
if (sat_quant<25)
{
    if (max>230) anteil_weiss++;
    else anteil_grau++;
}
else
if (sat_quant>230)
    anteil_vollsat++;
else
if (sat_quant>127)
    anteil_halbsat++;

sat_quant = sat*3.9999999;        // quantisieren auf 4 Stufen
br_quant = max/255.0*3.9999999;   // quantisieren auf 4 Stufen

if ( sat_quant==0 || br_quant==0)
    hsv = br_quant + 162;         // 4 Graustufen
else
{
    b1 = double(max-b)/dif;
    g1 = double(max-g)/dif;
    r1 = double(max-r)/dif;

    if (max==r)
    {
        if (min==g) hue=5+b1; else hue=1-g1;
    }
    else if (max==g)
    {
        if (min==b) hue=1+r1; else hue=3-b1;
```

```
            }
            else
            {
                if (min==r) hue=3+g1; else hue=5-r1;
            }

            is_hue[int(hue/6.0*255.99999)]=true;
            hue_quant = hue/6.0*17.999999; //quantisieren auf 18 Stufen
            hsv = ( (br_quant-1)*3 + sat_quant-1 )*18 + hue_quant;
        }

        hsv_anz[hsv]++;
        data_hsv[pos] = hsv;
    }

    fake_sat_avg/=picsize;
    intens_avg/=picsize;
    double ant_schw = (double)anteil_schwarz/picsize*100;
    double ant_weis = (double)anteil_weiss/picsize*100;
    double ant_grau = (double)anteil_grau/picsize*100;
    double ant_voll = (double)anteil_vollsat/picsize*100;
    double ant_halb = (double)anteil_halbsat/picsize*100;

    double intens_var=0, intens_entropy=0, intens_wechsel,
           hsv_entropy=0, hsv_wechsel;

    for (pos=0; pos<picsize; pos++)
    {
        double tmp = data_intens[pos] - intens_avg;
        intens_var += tmp * tmp;
    }
    intens_var/=picsize;

    int intens_x=0, intens_y=0, hsv_x=0, hsv_y=0;

    for (int y=0; y<ysize; y++)
        for (int x=0; x<xsize; x++)
        {
            unsigned char *posi, *posh;

            posi = &data_intens[y*xsize+x];
            posh = &data_hsv[y*xsize+x];

            if (x)
            {
                if (*posi != *(posi-1)) intens_x++;
                if (*posh != *(posh-1)) hsv_x++;
            }
            if (y)
            {
                if (*posi != *(posi-xsize)) intens_y++;
```

```
                if (*posh != *(posh-xsize)) hsv_y++;
            }
        }

    intens_x = intens_x < intens_y ? intens_x : intens_y;
    intens_wechsel = (double)intens_x/picsize*100;

    hsv_x = hsv_x < hsv_y ? hsv_x : hsv_y;
    hsv_wechsel = (double)hsv_x/picsize*100;

    unsigned int anzahl_hell=0, anzahl_sat=0,anzahl_hue=0,anzahl_hsv=0;

    for (i=0; i<256; i++) {
        if (is_hell[i]) anzahl_hell++;
        if (is_sat[i]) anzahl_sat++;
        if (is_hue[i]) anzahl_hue++;

        if (intens_anz[i]) {
            double pk=(double)intens_anz[i]/picsize;
            intens_entropy -= pk * log10(pk)/LOG2;
        }
    }
    for (i=0; i<166; i++)
        if (hsv_anz[i])
        {
            anzahl_hsv++;
            double pk=(double)hsv_anz[i]/picsize;
            hsv_entropy -= pk * log10(pk)/LOG2;
        }

    FILE *out = fopen("OUT.TXT","ab");
    fprintf(out,"Bild: %s\n\n\
Durchschnittliche \"Saettigung\": %6.2f\n\
Anteil Schwarz:               : %6.2f%%\n\
Anteil Weiss:                 : %6.2f%%\n\
Anteil Grau:                  : %6.2f%%\n\
Anteil volle Saettigung       : %6.2f%%\n\
Anteil halbe Saettigung       : %6.2f%%\n\
Anzahl Helligkeiten           : %3d\n\
Anzahl Saettigungen           : %3d\n\
Anzahl Farbtoene              : %3d\n\
Anzahl 166-HSV-Werte          : %3d\n\
Intensitaets-Varianz          : %6.2f\n\
Intensitaets-Entropie         : %6.2f\n\
Intensitaets-Wechselmass      : %6.2f\n\
166-HSV-Entropie              : %6.2f\n\
166-HSV-Wechselmass           : %6.2f\n\n",
    argv[1],fake_sat_avg,ant_schw,ant_weis,ant_grau,ant_voll,ant_halb,
    anzahl_hell,anzahl_sat,anzahl_hue,anzahl_hsv,
    intens_var,intens_entropy,intens_wechsel,hsv_entropy,hsv_wechsel);
    fclose(out);
```

```
    delete[] data;
    delete[] data_intens;
    delete[] data_hsv;
}
```

D.6 Blocks.cpp

```
// Führt eine Wavelet-Transformation eines Bildes aus, segmentiert
// es in Blöcke und markiert diese farbig. Rot bedeutet, daß für den
// Block eine Verarbeitung für natürliche Bildinhalte vorgenommen
// werden sollte, blau für grafische Bildinhalte, sowie grün für Text.
// Kommandozeilenparameter:
// 1. Dateiname eines Bildes im Format 24Bit BMP
// 2. Dateiname eines Bildes für die Ausgabe (im selben Format)

#include <math.h>
#include "bmpfunc.h"

// ********************************************************************

void analyse_block(float *blockdata, int blxsize, int blysize,
                   int xsize, int ymid,
                   bool &is_photo, bool &is_graph, bool &is_text)
{
    int coeff_anz[1024], blksize = blxsize*blysize, xmid = xsize/2;
    double var=0;
    for (int i=0; i<1024; i++) coeff_anz[i]=0;

    for (int y=0; y<blysize; y++)       // Ermittlung von Varianz und
        for (int x=0; x<blxsize; x++)   // Verteilung (Mittelwert 0)
        {
            double coeff;
            coeff = blockdata[y*xsize + xmid+x];
            var += coeff*coeff;
            coeff_anz[int(coeff*4+512)]++;    // HL-Band

            coeff = blockdata[ (y+ymid)*xsize + xmid+x];
            var += coeff*coeff;
            coeff_anz[int(coeff*4+512)]++;    // HH-Band

            coeff = blockdata[ (y+ymid)*xsize + x];
            var += coeff*coeff;
            coeff_anz[int(coeff*4+512)]++;    // LH-Band
        }

    int ccount = blksize*3;
    var /= ccount;
```

```
double lamb = -sqrt(2/var)/4;
double lastexp=1, nextexp, tmp;
double match = 0;

for (i=0; i<512; i++) {
    nextexp = exp(lamb*(i+1));
    double lapl = 0.5 * (lastexp - nextexp) * ccount;
    if (lapl<0.1) break;         // Funktion ist zu klein geworden
    lastexp = nextexp;

    tmp = coeff_anz[512+i] - lapl;
    match+=tmp*tmp/lapl;
    if (i)
    {
        tmp = coeff_anz[512-i] - lapl;
        match+=tmp*tmp/lapl;
    }
}
match /= ccount;                 // Laplace-Match berechnet
is_photo = match < 35;

int zones[512], max[512], j=0, lastmax=0, lastborder=0, coeff, t=1;

zones[0] = 0;
#define delta 20

while (t<1023)
{
    do {
        t++;                          // lokales Maximum finden
        if (coeff_anz[t-1] < (coeff=coeff_anz[t]))
        {
            while ( coeff == coeff_anz[t+1] && t<1023)
            {
                t++;
                coeff=coeff_anz[t];
            }
            if (coeff_anz[t+1] < coeff) break;
        }
    } while (t<1023);

    if (t == 1023) break;

    if ( (lastmax && coeff > lastmax) ||
         (!lastmax && lastborder <= coeff / delta) )
    {
        max[j] = t;
        lastmax = coeff;
    }

    do {
```

```
            t++;                        // lokales Minimum finden
            if (coeff_anz[t-1] > (coeff=coeff_anz[t]))
            {
                while ( coeff == coeff_anz[t+1] && t<1023)
                {
                    t++;
                    coeff=coeff_anz[t];
                }
                if (coeff_anz[t+1] > coeff) break;
            }
        } while (t<1023);

        if (t == 1023) break;

        if (lastmax)
        {
            if (coeff <= lastmax / delta)
            {
                zones[++j] = t;
                max[j]=0;
                lastmax=0;
                lastborder = coeff;
            }
        }
        else
        {
            if (coeff < lastborder)
            {
                zones[j] = t;
                lastborder = coeff;
            }
        }

    }
    if (!max[j]) max[j]=t-1;
    zones[++j] = t;

    double l_crit=0;
    int coeff_restl=0, coeff_maxl=0, maxlpos, coeff_max2l=0, ldist,
        coeff_restr=0, coeff_maxr=0, maxrpos, coeff_max2r=0, rdist;

    for (t=0; t<j; t++)
    {
        int left, right, coeff_near, coeff_total;
        left = max[t]-1;
        right = max[t]+1;

        coeff_near = coeff_total = 0;
        for (i=zones[t]; i<=zones[t+1]; i++)
        {
            coeff_total += coeff_anz[i];
```

```
            if (i>=left && i<=right) coeff_near += coeff_anz[i];
        }

        zones[t]=coeff_total;

        if (max[t]<412)
        {
            if (coeff_total > coeff_maxl)
            {
                coeff_maxl = coeff_total;
                maxlpos = max[t];
            }
            coeff_restl += coeff_total;
        }
        else if (max[t]>612)
        {
            if (coeff_total > coeff_maxr)
            {
                coeff_maxr = coeff_total;
                maxrpos = max[t];
            }
            coeff_restr += coeff_total;
        }

        l_crit += (double)coeff_near * coeff_near / coeff_total;
    }

    l_crit /= ccount;              // Konzentrations-Mass berechnet
    is_graph = l_crit >
               ( (blksize>=100) ? 0.94 : (blksize>=8*8? 0.97 : 0.99) );

    for (t=0; t<j; t++)
    {
        int coeff_total = zones[t];
        if (max[t]<412)
        {
            int dist = maxlpos - max[t];
            if (coeff_total>coeff_max2l && (dist>100 || dist<-100) )
            {
                coeff_max2l = coeff_total;
                ldist = dist;
            }
        }
        else if (max[t]>612)
        {
            int dist = max[t] - maxrpos;
            if (coeff_total>coeff_max2r && (dist>100 || dist<-100) )
            {
                coeff_max2r = coeff_total;
                rdist = dist;
            }
```

```
        }
    }

    is_text=false;
    int maxlsum = coeff_maxl + coeff_max2l,
        maxrsum = coeff_maxr + coeff_max2r;

    coeff_restl -= maxlsum;
    coeff_restr -= maxrsum;

    int needed = (blksize >= 100 ? 44 : 18);

    if ( (coeff_restl*15 < maxlsum) && (coeff_restr*15 < maxrsum) &&
        (maxlsum>needed || maxrsum>needed))
    {
        if ( (coeff_max2l &&
              (ldist > 0 || coeff_maxl*6 < coeff_max2l*7)) ||
             (coeff_max2r &&
              (rdist > 0 || coeff_maxr*6 < coeff_max2r*7)) )
            is_text=true;
    }

}

// ***************************************************************************

void mark_blocks_recursive(char *data, float *data_intens,
                           int xsize, int ymid, int xstart, int ystart,
                           int xend, int yend, int blocksize)
{
    for (int y=ystart; y<yend; y+=blocksize)
        for (int x=xstart; x<xend; x+=blocksize)
        {
            int bly = yend-y; if (bly>blocksize) bly=blocksize;
            int blx = xend-x; if (blx>blocksize) blx=blocksize;

            bool is_photo, is_graph, is_text;

            analyse_block(&data_intens[y*xsize + x], blx, bly,
                          xsize, ymid, is_photo, is_graph, is_text);

            char blocktype = 0;
            if (is_graph)
            {
                if (blocksize >= 8) {
                    if (is_text) blocktype = 1;
                } else blocktype = 2;
            }
            else
            {
                if (is_photo)
```

```
                    blocktype = 3;
            }
            if (!blocktype)
                mark_blocks_recursive(data, data_intens, xsize, ymid,
                                      x, y, x+blx, y+bly, blocksize/2);
            else
            {
                char *temp=&data[y*xsize*2 + x*2];
                bly*=2; blx*=2;

                for (int recty=0; recty<bly; recty++)
                    for (int rectx=0; rectx<blx; rectx++)
                        temp[recty*xsize+rectx]=blocktype;
            }
        }
}

// ********************************************************************

void main(int argc,char *argv[])
{
    if (argc != 3)
    {
        printf("Bitte Dateinamen für Ein- und Ausgabe angeben.");
        return;
    }

    int xsize, ysize, picsize, padding;
    unsigned short buf[27];
    rgb_pixel *data;

    if (!open_bmp(argv[1], xsize, ysize, picsize, padding, buf, &data))
        return;

    float *data_intens = new float[picsize];
    char *blocks = new char[picsize];

    // Umwandlung in Graustufen
    for (int pos=0; pos<picsize; pos++)
    {
        int b,g,r;
        r = data[pos].r;
        g = data[pos].g;
        b = data[pos].b;
        data_intens[pos] = (r*3 + g*6 + b)/10;
    }

    float *temp_buf = new float[xsize>ysize ? xsize:ysize];

    // Horizontale Transformation
    int xmid = xsize/2;
```

```
    for (int y=0; y < ysize; y++)
    {
        for (int x=0; x < xmid; x++)
        {
            float val1 = data_intens[y*xsize + x*2];
            float val2 = data_intens[y*xsize + x*2 + 1];
            temp_buf[x] = (val1+val2)*0.5; // Tiefpass für Haar-Wavelet
            temp_buf[x + xmid] = (val1-val2)*0.5;  // Hochpass
        }
        for (x=0; x < xsize; x++)
            data_intens[y*xsize + x] = temp_buf[x];
    }

    // Vertikale Transformation
    int ymid = ysize/2;
    for (int x=0; x < xsize; x++)
    {
        for (int y=0; y < ymid; y++)
        {
            float val1 = data_intens[y*2*xsize + x];
            float val2 = data_intens[(y*2 + 1)*xsize + x];
            temp_buf[y] = (val1+val2)*0.5; // Tiefpass für Haar-Wavelet
            temp_buf[y + ymid] = (val1-val2)*0.5;  // Hochpass
        }
        for (y=0; y < ysize; y++)
            data_intens[y*xsize + x] = temp_buf[y];
    }

    mark_blocks_recursive(blocks, data_intens, xsize, ymid, 0, 0,
                          xmid, ymid, 16);

    for (y=0; y < ysize; y++)
        for (int x=0; x < xsize; x++)
        {
            int pos = y*xsize+x;
            char bl=blocks[pos];
            if ( !x || x==xsize-1 || !y || y==ysize-1 ||
                 bl!=blocks[pos-xsize] || bl!=blocks[pos+xsize] ||
                 bl!=blocks[pos-xsize-1] || bl!=blocks[pos-xsize+1] ||
                 bl!=blocks[pos+xsize-1] || bl!=blocks[pos+xsize+1] ||
                 bl!=blocks[pos-1] || bl!=blocks[pos+1])
                switch (bl)
                {
                    case 1:
                        data[pos].g=200;                    // Text
                        data[pos].r=data[pos].b=0;
                        break;
/*                    case 2:
                        data[pos].b=255;                    // Grafik
                        data[pos].r=data[pos].g=0;
                        break;*/
```

```
                    case 3:
                        data[pos].r=255;                    // Bild
                        data[pos].g=data[pos].b=0;
                }
        }

    // Output der Block-Einteilung
    write_bmp(argv[2], xsize, ysize, padding, buf, data);

    delete[] temp_buf;
    delete[] blocks;
    delete[] data_intens;
    delete[] data;
}
```

Erklärung

Ich versichere, dass ich diese wissenschaftliche Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe. Die Stellen der Arbeit, die anderen Werken dem Wortlaut oder Sinn nach entnommen sind, wurden in jedem einzelnen Fall durch Angabe der Quelle als Entlehnung kenntlich gemacht. Das Gleiche gilt auch für beigegebene Skizzen und Darstellungen. Die Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.

Dortmund, den 28. Mai 2003

Einwilligung

Hiermit erkläre ich mich damit einverstanden, dass diese wissenschaftliche Arbeit nach den Bestimmungen des §6 Absatz 1 des Gesetzes über Urheberrecht vom 9.9.1965 in die Bereichsbibliothek aufgenommen und damit für Leser der Bibliothek öffentlich zugänglich gemacht wird.

Ferner bin ich damit einverstanden, dass gemäß §54 Absatz 1 Satz 1 dieses Gesetzes Leser zu persönlichen wissenschaftlichen Zwecken Kopien aus der Arbeit anfertigen dürfen.

Dortmund, den 28. Mai 2003