Vergleich von Techniken zur lokalen Strukturanalyse in der Bildverarbeitung

Diplomarbeit

am Lehrstuhl für Kognitive Systeme
Institut für Informatik und Praktische Mathematik
Christian-Albrechts-Universität zu Kiel

Prof. Dr. Gerald Sommer

vorgelegt von

Stephan Mischnick

Betreuer: Michael Felsberg

11. Juli 2001

Inhaltsverzeichnis

Kapitel 1

Einleitung

Eine wesentliche Aufgabe der Bildverarbeitung ist die Bildanalyse. Diese kann auf vielfältige Art und Weise durchgeführt werden. In dieser Arbeit beschränkt sich die Analyse von Bildern auf lokale Strukturen, die in ihnen enthalten sein können. Es werden Bilder, die von Abbildungssystemen wie z.B. CCD-Kameras oder Scannern erzeugt werden, hinsichtlich lokaler Strukturen analysiert. Unter der lokalen Struktur eines Bildes bei einem Bildpunkt versteht man im Wesentlichen folgende Merkmale:

- Kanten (zusammenhängende Ränder von Bildobjekten)
- Verbindungspunkte (z.B. Y-, X- und T-Verbindungen)
- Orientierungen und Richtungen (Verallgemeinerung von Verbindungspunkten)
- Flächen und Texturen (gleichmäßige Muster).

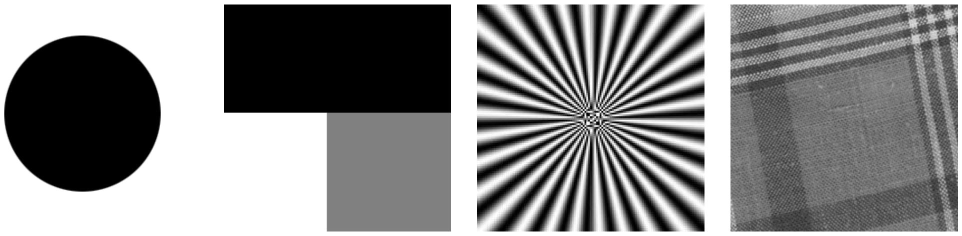


Abbildung 1.1: Beispiele für grundlegende Merkmale

Abbildung 1.1 stellt einige Merkmale exemplarisch vor (Die Bilder von links nach rechts betrachtet: Kanten am Rand des Kreises, T-Verbindungspunkt in der Mitte, Multi-Richtungen im

Zentrum des Siemenssterns und eine gleichmäßige Textur). Ziel der lokalen Strukturanalyse ist es, den Bildpunkten eines Bildes diese Merkmale sinnvoll zuzuordnen. Die dabei verwendeten Filter dienen dazu, die im Bild vorhandenen Strukturen in für die Weiterverarbeitung passendere Strukturen, umzuwandeln. Diese können möglicherweise auch symbolischer Natur sein.

Es treten grundsätzliche Probleme auf, die die Analyse negativ beeinflussen und zu falschen Zuordnungen (*false-positives*) führen können. Diese Probleme entstehen beim physikalischen Abbildungsvorgang, dort wird das eigentliche Signal mit Störungen bzw. Ungenauigkeiten behaftet. Solche können z.B. in der Form von

- Rauschen (in der Regel normalverteilt und mittelwertfrei)
- Übertragungsfehler (sog. Salz- und Pfefferrauschen)
- Verzerrungen (nichtlineare Abbildungen)
- Unschärfe (ein Punkt wird auf eine Fläche abgebildet)

charakterisiert werden.

In dieser Arbeit werden sechs verschiedene Verfahren hinsichtlich der lokalen Strukturanalyse betrachtet. Es handelt sich dabei um die Verfahren

1. Polarpyramiden (Yu, Daniilidis und Sommer [26])
2. Steuerbare Filter (Freeman und Adelson [10])
3. Wedgefilter (Simoncelli und Farid [21])
4. Strukturtensor (Jähne [14])
5. Phasenkongruenz (Kovesi [15])
6. Monogenes Signal (Felsberg und Sommer [8])

Das erste Ziel dieser Arbeit ist es, die sechs Verfahren hinsichtlich ihrer Verwendung und der Abbildungsprobleme zu untersuchen (Kapitel 4). Entscheidend ist dabei, dass von einer Bildermenge ausgegangen wird, die nicht künstlich erzeugt wurde. Man hat also nicht die Möglichkeit auf *a priori* Wissen zurückzugreifen. Ein weiteres Ziel ist es, die Verfahren, hinsichtlich ihrer Tauglichkeit zur Detektion der oben genannten Merkmale, einzuordnen (Kapitel 5).

Diese Verfahren haben unterschiedliche Ansätze für die Analyse, die zum Teil mathematisch und zum Teil intuitiv begründet werden. In Kapitel 3 werden die Ansätze dargelegt.

In Kapitel 2 werden die nötigen mathematischen Grundlagen, d.h. die für das Verständis wichtigen Definitionen, Sätze und Theoreme formuliert. Sie ermöglichen es, den mathematischen Hintergrund der einzelnen Ansätze besser zu verstehen.

Kapitel 2

Mathematische Grundlagen

Um 2D-Signale bzw. Bilder mathematisch zu erfassen, werden sie als Funktionen geschrieben. Die in der Arbeit betrachteten Grauwertbilder lassen sich für ein zweidimensionales kartesisches Koordinatensystem wie folgt definieren:

Definition 2.1 (Bild) *Eine Abbildung* $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$*,*

$$f(x,y) = f_{xy} \qquad \textit{mit} \quad x, y, f_{xy} \in \mathbb{R}. \tag{2.1}$$

heißt Bild bzw. Bildfunktion. Diese sind spezielle Ausprägungen von 2D-Signalen. Filter bzw. Filterfunktionen werden ebenfalls in dieser Form angegeben. •

Für ein Grauwertbild, welches z.B. durch eine Kamera aufgenommen wurde, gelten üblicherweise Einschränkungen des Definitions- und Wertebereiches. Die Bildfunktion f ist dann in der Regel eine Abbildung $f : \{0..g_x - 1, 0..g_y - 1\} \longrightarrow \{0..255\}$, wobei g_x und g_y jeweils die Größe der Dimension in entsprechender Richtung angeben.

Mitunter werden Bildfunktionen im Polarkoordinatensystem betrachtet. Die Bildfunktion (2.1) hat dann die Form

$$f(r,\phi) = f_{r\phi} \qquad \text{mit} \quad f_{r\phi} \in \mathbb{R}, \phi \in [0, 2\pi), r \in \mathbb{R}^+ \tag{2.2}$$

wobei die folgende Koordinatentransformation benutzt wird:

$$\mathbb{R}^2 \longrightarrow (\mathbb{R}^+, [0, 2\pi)) \quad \text{mit} \quad (x, y) \longmapsto (\sqrt{x^2 + y^2}, \arg(x, y)). \tag{2.3}$$

Diese Transformation wird als Polarkoordinatentransformation bezeichnet.

2.1 Grundbegriffe der Steuerbarkeitstheorie

In der Steuerbarkeitstheorie werden Filterfunktionen betrachtet, die sich als Linearkombination einer festen Menge von Basisfunktionen schreiben lassen. Mit solchen Linearkombinationen lassen sich Filterantworten oft günstiger berechnen.

2.1.1 Arten der Steuerbarkeit

Transformationen können mehrere Freiheitsgrade aufweisen. Beispielsweise kann sich eine Transformation aus einer Drehung und einer Verschiebung zusammensetzen. Wir beschränken uns hier auf die Drehung mit positivem Rotationssinn um den Ursprung, die nur einen Freiheitsgrad hat. Eine Drehung einer Bildfunktion f um den Winkel $\theta \in [0, 2\pi)$ wird kurz geschrieben als f^{θ}.

Definition 2.2 (speziell steuerbar) *Eine Bildfunktion f heißt speziell steuerbar bzgl. eines Winkels $\theta \in [0, 2\pi)$, wenn es eine Menge $B \overset{\text{def}}{=} \{f^{\theta_l} | l \in \{1, \ldots, M\}, M \in \mathbb{N}\}$ von Bildfunktionen gibt, derart dass*

$$f^{\theta} = \sum_{l=1}^{M} k_l(\theta) f^{\theta_l} \tag{2.4}$$

für geeignete Funktionen $k_l : [0, 2\pi) \longrightarrow \mathbb{R}$ gilt. Die Menge B wird als Basismenge bezeichnet, die k_l heißen Interpolationsfunktionen. •

In dieser Definition der Steuerbarkeit einer Bildfunktion, die z.B. auch von Freeman in [10] benutzt wird, werden spezielle Basiselemente gefordert. Bei den Basiselementen handelt es sich um ursprungsrotierte Versionen der Ausgangsbildfunktion. Ohne diese Einschränkung erhält man eine verallgemeinerte Form der Steuerbarkeit.

Definition 2.3 (steuerbar) *Eine Bildfunktion f heißt steuerbar bzgl. eines Winkels $\theta \in [0, 2\pi)$, wenn es eine Menge $B \overset{\text{def}}{=} \{g_l | l \in \{1, \ldots, M\}, M \in \mathbb{N}\}$ von Bildfunktionen gibt, derart dass*

$$f^{\theta} = \sum_{l=1}^{M} k_l(\theta) g_l \tag{2.5}$$

für geeignete Interpolationsfunktionen $k_l : [0, 2\pi) \longrightarrow \mathbb{R}$ gilt. •

Diese Definition wird z.B. in [17, 23] im Rahmen der Lie-Theorie eingesetzt. Sie eignet sich auch beweistechnisch zur Verifizierung hilfreicher Theoreme, wie z.B. in [10], die ursprünglich auf der speziellen Steuerbarkeit aufbauen.

Zwischen der speziellen Steuerbarkeit und der Steuerbarkeit besteht der folgende einfache Zusammenhang:

Lemma 1 *Sei f eine Bildfunktion, dann gilt: Ist f speziell steuerbar bzgl. eines Winkels θ, so ist f auch steuerbar bzgl. des Winkels θ.* •

Beweis Sei f eine Bildfunktion, die speziell steuerbar bzgl. des Winkels θ ist. Dann gilt:

$$f^{\theta} = \sum_{l=1}^{M} k_l(\theta) f^{\theta_l}, M \in \mathbb{N}. \tag{2.6}$$

Setze für alle $l \in \{1, \dots, M\}$

$$g_l \overset{\text{def}}{=} f^{\theta_l}, \tag{2.7}$$

so erhält man direkt die Definition der Steuerbarkeit. •

2.1.2 Anwendung der Steuerbarkeit

Ist f eine Filterfunktion, die durch die Menge $\{g_l | l \in \{1, \dots, M\}, M \in \mathbb{N}\}$ für geeignete Interpolationsfunktionen $k_l(\theta)$ gesteuert wird, so gilt aufgrund der Linearität der Faltungsoperation für eine Bildfunktion p:

Theorem 1 *Sei f steuerbar (in der Form gemäß (2.5)) bzgl.eines Winkel $\theta \in [0, 2\pi)$, dann gilt für eine Bildfunktion p:*

$$p * f^{\theta} = \sum_{l=1}^{M} k_l(\theta)(p * g_l) \tag{2.8}$$

Der Vorteil der Steuerbarkeit besteht also darin, dass die Bildfunktion lediglich mit allen Basisfunktionen gefaltet werden muss, um dann mit Hilfe der Interpolationsfunktionen $k_l(\theta)$ jede mögliche Filterantwort zu berechnen. Im Gegensatz dazu müsste man anderenfalls zu jedem möglichen f^{θ} eine Faltung mit der Bildfunktion durchführen, was natürlich ineffizent ist.

2.2 Fouriertransformation

2.2.1 Die Fouriertransformation

Eine der wichtigsten Transformationen in der Bildverarbeitung ist die Fouriertransformation, die für ein eindimensionales Signal f die Form

$$F(k) = \frac{1}{2\pi} \int_{-\infty}^{\infty} f(x) e^{-ikx} dx$$

hat. Diese Transformation zählt zu den unitären Transformationen, d.h. der Wertebereich besteht aus den komplexen Zahlen $\mathbb{C}$ und es existiert ein Skalarprodukt. Weiterhin ist die Transformation vollständig. Es existiert also eine inverse Transformation

$$f(x) = \int_{-\infty}^{\infty} F(k) e^{ikx} dx$$

mit der Eigenschaft

$$f = \mathcal{F}^{-1}(\mathcal{F}(f)),$$

wobei $\mathcal{F}$ die Fouriertransformation und $\mathcal{F}^{-1}$ ihre Inverse darstellt.

Die Fouriertransformation ist separabel, d.h. für z.B. zwei Dimensionen existiert ebenfalls eine Fouriertransformation, die sich als Hintereinanderausführung zweier eindimensionaler Fouriertransformationen entlang der verschiedenen Koordinatenachsen schreiben lässt.

Für die Bildverarbeitung ist die Fouriertransformation von zweierlei Interesse. Zum einen bietet die Fouriertransformation eine hilfreiche Interpretationsmöglichkeit von Signalen und Bildfunktionen in Abhängigkeit von Frequenzen und Winkeln, zum anderen liefert sie die Möglichkeit über die schnelle Fouriertransformation (FFT), die für lineare Filteroperationen unerlässliche Faltung effizienter für größere Bildfunktionen zu berechnen. Dabei wird der Faltungssatz für zwei Bildfunktionen f und g ausgenutzt.

$$f * g = \mathcal{F}^{-1}(\mathcal{F}(f) \cdot \mathcal{F}(g)) \tag{2.9}$$

Weitere wichtige Eigenschaften der Fouriertransformation sind z.B. in [14] angegeben.

2.2.2 Steuerbarkeit durch Ausnutzung von Fourierreihen

In diesem Abschnitt wird die Steuerbarkeit im Rahmen der Fouriertheorie behandelt. Es wird angenommen, dass eine Bildfunktion f im Polarkoordinatensystem durch eine Fourierreihe

$$f(r, \phi) = \sum_{n=-N}^{N} a_n(r) e^{in\phi} \tag{2.10}$$

dargestellt werden kann. Ein für das nachfolgende Theorem nützliches Lemma lautet:

Lemma 2 *Sei f eine Bildfunktion mit der Fourierreihendarstellung*

$$f(r, \phi) = \sum_{n=-N}^{N} a_n(r) e^{in\phi}, \tag{2.11}$$

dann ist jede um θ rotierte Version von f ebenfalls eine Fourierreihe mit der Darstellung

$$f^{\theta}(r, \phi) = \sum_{n=-N}^{N} a_n(r) e^{in\phi} e^{in\theta}. \tag{2.12}$$

•

Beweis Sei f eine Bildfunktion mit der Fourierreihendarstellung (2.11) und $\theta \in [0, 2\pi)$ ein Winkel, dann gilt wegen

$$f^\theta(r,\phi) = f(r, \phi + \theta), \tag{2.13}$$

dass sich f^θ als Fourierreihe der Form

$$\begin{aligned} f^\theta(r,\phi) &= \sum_{n=-N}^{N} a_n(r) e^{in(\phi+\theta)} && (2.14)\\ &= \sum_{n=-N}^{N} a_n(r) e^{in\phi} e^{in\theta} && (2.15)\end{aligned}$$

darstellen lässt. •

Es hat sich in [21] herausgestellt, dass f in der Form (2.10) bzgl. eines Winkels θ speziell gesteuert werden kann, wenn folgende Äquivalenz gilt:

Theorem 2 *Sei f eine Bildfunktion in Polarkoordinaten, die sich als Fourierreihe schreiben lässt (2.10). Dann gilt: f ist speziell steuerbar bzgl. eines Winkels θ genau dann, wenn das Gleichungssystem für $N, M \in \mathbb{N}$*

$$\begin{pmatrix} 1 & \cdots & 1 \\ e^{i\theta_1} & \cdots & e^{i\theta_M} \\ \vdots & \ddots & \vdots \\ e^{iN\theta_1} & \cdots & e^{iN\theta_M} \end{pmatrix} \begin{pmatrix} k_1(\theta) \\ \vdots \\ k_M(\theta) \end{pmatrix} = \begin{pmatrix} 1 \\ e^{i\theta} \\ \vdots \\ e^{iN\theta} \end{pmatrix} \tag{2.16}$$

in Bezug auf die Interpolationsfunktionen k_n, $n \in \{1, \ldots, M\}$ lösbar ist.

Beweis Sei f eine Bildfunktion die sich als Fourierreihe darstellen lässt (2.10). Die Äquivalenz wird in einem Schritt gezeigt. Sei f speziell steuerbar bzgl. eines Winkels θ. Dann gilt mit Lemma 2 für die f^{θ_i}:

$$\begin{aligned} f^\theta(r,\phi) &= \sum_{l=1}^{M} k_l(\theta) f^{\theta_l} && (2.17)\\ &= \sum_{l=1}^{M} k_l(\theta) \sum_{n=-N}^{N} a_n(r) e^{in\phi} e^{in\theta_l}. && (2.18)\end{aligned}$$

Setzt man für f^θ ebenfalls die entsprechende Fourierreihe ein, so gilt:

$$\begin{aligned} f(r,\phi) &= \sum_{n=-N}^{N} a_n(r) e^{in\phi} e^{in\theta} && (2.19)\\ &= \sum_{l=1}^{M} k_l(\theta) \sum_{n=-N}^{N} a_n(r) e^{in\phi} e^{in\theta_l}, && (2.20)\end{aligned}$$

mit der Abkürzung $f_s \overset{\text{def}}{=} a_n(r)e^{in\phi}$ vereinfacht sich die Schreibweise zu

$$\sum_{n=-N}^{N} f_s e^{in\theta} = \sum_{l=1}^{M} \sum_{n=-N}^{N} k_l(\theta) e^{in\theta_l} f_s. \tag{2.21}$$

Ein Koeffizientenvergleich ergibt M Gleichungen

$$e^{in\theta} = \sum_{l=1}^{M} k_l(\theta) e^{in\theta_l}, \tag{2.22}$$

welche sich im Gleichungssystem (2.16) widerspiegeln. Die umgekehrte Implikation ergibt sich direkt, da die einzelnen Umformungen und Argumente in anderer Richtung analog sind. •

Das Theorem 2 liefert ein Verfahren, mit dem die Menge der Interpolationsfunktionen $k_l(\theta)$ gefunden werden kann, um die Bildfunktion f bzgl. des Winkels θ speziell zu steuern.

Um von einer Basismenge reden zu können, muss gezeigt werden, dass die Menge minimal in Bezug auf die Steuerbarkeit ist. Es ergibt sich ein weiteres Theorem:

Theorem 3 *Sei f eine Bildfunktion in Polarkoordinaten, die sich als Fourierreihe darstellen lässt. Sei T die Anzahl der von Null verschiedenen Koeffizienten der Fourierreihe, dann ist f steuerbar mit der minimalen Anzahl T von Basisfunktionen.*

Beweis Sei f eine Bildfunktion in Polarkoordinaten, die sich als Fourierreihe darstellen lässt. Sei T die Anzahl der von Null verschiedenen Fourierkoeffizienten. Mit dem Lemma 1 und dem Theorem 2 findet man Basisfunktionen g_l ($l \in \{1, \ldots, M\}, M \in \mathbb{N}$) derart, dass f steuerbar bzgl. eines Winkels θ ist. Es gilt also:

$$\begin{aligned} f^{\theta}(r,\phi) &= \sum_{l=1}^{M} k_l(\theta) g_l(r,\phi) \\ &= \sum_{n=-N}^{N} a_n(r) e^{in(\theta+\phi)} \end{aligned}$$

Projeziert man diese Gleichung auf $e^{im\phi}$ und teilt durch $a_m(r) \neq 0$, so erhält man Bedingungsgleichungen:

$$e^{im\theta} = \sum_{l=1}^{M} k_l(\theta) c_{lm}(r), 0 \leq m \leq N, \tag{2.23}$$

wobei

$$c_{lm} \overset{\text{def}}{=} \frac{g_l(r,\phi)}{a_m(r)e^{im\phi}}. \tag{2.24}$$

Orthonormalisiert man die Gleichungen (2.23), so lässt sich die Matrixgleichung

$$I = CKC^t \tag{2.25}$$

aufstellen. Dabei ist $I_{(T\times T)}$ die Identitätsmatrix, $C_{(T\times M)}$ eine Matrix mit Elementen aus (2.24) und $K_{(M\times M)}$ eine Matrix, deren Elemente gegeben sind durch:

$$K_{pl} = \frac{1}{2\pi}\int_{-\pi}^{\pi} k_p(\theta)k_l(\theta)d\theta. \tag{2.26}$$

Da I den Rang T hat, muss K mindestens den Rang T haben. Damit f steuerbar bzgl. θ ist, muss also $M \geq T$ gelten. •

Theorem 4 *Sie $f(x,y) = w(r)P_N(x,y)$, wobei $w(r)$ eine rotationssymmetrische Fensterfunktion und P_N ein Polynom n-ten Grades ist, dessen Koeffizienten von r abhängig sind. Dann gilt: $2N+1$ Basisfunktionen werden benötigt, um f zu steuern. Besitzt f nur gerade (ungerade) Summanden, so werden nur $N+1$ Basisfunktionen benötigt.*

Beweis Sei f mit den Eigenschaften in der Voraussetzung gegeben. Betrachtet man einen Summanden des Polynoms P_N in Polarkoordinaten, so gilt für k,n mit $0 \leq k \leq n$:

$$x^k y^{n-k} = r^n \cos^k(\phi)\sin^{n-k}(\phi), \tag{2.27}$$

wobei $x = r\cos(\phi)$ und $y = r\sin(\phi)$ sind. Entwickelt man das Produkt für Summanden geraden Grades in einer Fourierreihe, so stellt man fest, dass nur die Frequenzen $n\phi, (n-2)\phi, \ldots, 0, \ldots, -(n-2)\phi, -n\phi$ auftreten. Man erhält also nur gerade Frequenzen. Nach Theorem 1 werden dann $N+1$ Basisfunktionen zum Steuern benötigt. Gleiches gilt für Summanden ungeraden Grades von P_N.

Für ein allgemeines Polynom N-ten Grades benötigt man sowohl alle geraden als auch ungeraden Frequenzanteile, d.h. es werden insgesamt $2N+1$ Frequenzen benutzt. Folglich gibt es $2N+1$ Basisfunktionen, die f steuern. •

2.3 Hilberttransformation

Die in dieser Arbeit vorkommenden Filter werden so entwickelt, dass nicht einzelne Pixel von Bildfunktionen, sondern die lokale Energie einer Umgebung eines Punktes betrachtet wird. Dabei wird hier der Begriff der Energie als ein Maß für den Integrationswert über die Umgebung des Pixel verstanden.

Um Filter für die Energieerfassung zu konstruieren, wird davon ausgegangen, dass das Signal bzw. die Bildfunktion sich fouriertransformieren lässt. Ziel ist es, einen Filter zu finden, der phasenunabhängig wirkt und die Amplitude unverändert lässt. In [14] wird dazu auf das trigonometrische Additionstheorem

$$\cos^2\theta + \sin^2\theta = 1 \tag{2.28}$$

zurückgegriffen (siehe Abbildung 2.1 und Abbildung 2.2). Dieses Theorem liefert einen Hinweis, wie ein Filterpaar zu konstruieren ist, welches die Amplitude unverändert lässt und eine Phasenverschiebung von $+\frac{\pi}{2}$ bewirkt. Die Transformation, die zu einer gegebenen Bildfunktion f eine entsprechende Transformierte f_H, die die erwähnten Eigenschaften hat, erzeugt, ist die Hilberttransformation:

$$f_H(x) = \frac{1}{\pi}\int_{-\infty}^{\infty} \frac{f(x')}{x'-x}dx'. \tag{2.29}$$

In der Praxis wird die Hilberttransformation für eine Bildfunktion f mit Hilfe der Fouriertransformation berechnet. Es wird folgender Algorithmus durchgeführt:

1. f wird zu F fouriertransformiert
2. $F_H = F \cdot H(k)$; Multiplikation mit dem Hilbertfilter $H(k)$

$$H(k) = \begin{cases} i & k > 0 \\ 0 & k = 0 \\ -i & k < 0 \end{cases}$$

3. F_H wird mit der inversen Fouriertransformation zu f_H transformiert.

Die so entstehende Bildfunktion f_H wird als Hilberttransformation von f bezeichnet. Ist f ein Filter, so wird das Paar (f, f_H) als Quadraturfilterpaar bezeichnet. Wichtige Quadraturfilterpaare sind z.B. die Gaborfilter (siehe z.B. Abbildung 2.3). Gaborfilter wählen einen Frequenzbereich mittels einer Gaußschen Glockenkurve aus (Bandpassfilter). Im Zweidimensionalen haben Gaborfilter zusätzlich die Möglichkeit der richtungsabhängigen Frequenzbereichsauswahl.

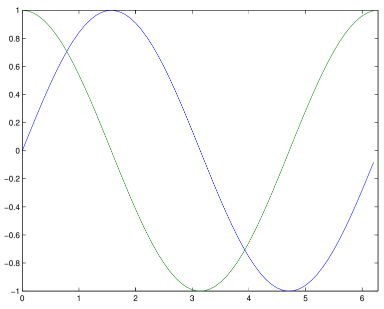


Abbildung 2.1: cos und sin als Ausgangsfunktionen.

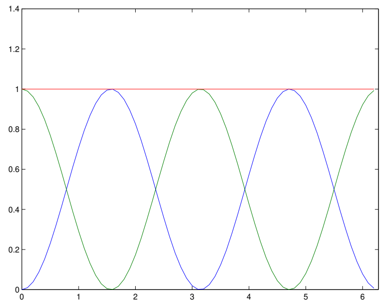


Abbildung 2.2: Quadrate der Amplituden und Summe der Quadrate.

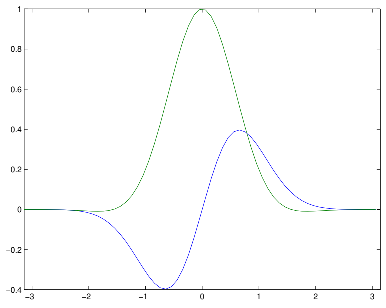


Abbildung 2.3: 1D-Gaborfilterpaar im Ortsraum.

Kapitel 3

Verfahren zur Strukturauswertung

3.1 Visualisierung von Strukturinformationen

3.1.1 Visualisierung von Richtungen

Ein Problem bei der Analyse von lokalen Richtungen ist deren Visualisierung. Für eine Beispielumgebung eines Aufpunktes P_A (siehe Abbildung 3.1), sind die folgenden Darstellungen (wie z.B. in [10]) in der Literatur üblich.

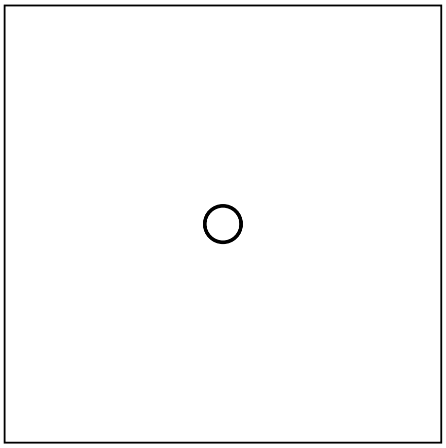


Abbildung 3.1: Beispielumgebung eines Aufpunktes P_A.

In der ersten Darstellungsform (Abbildung 3.2) wird die Energie – häufig wird die Summe der Gradienten benutzt – entlang einer Ursprungsgeraden mit einem Richtungswinkel θ dargestellt. Es wird dazu ein kartesisches Koordinatensystem verwendet, welches sich besonders

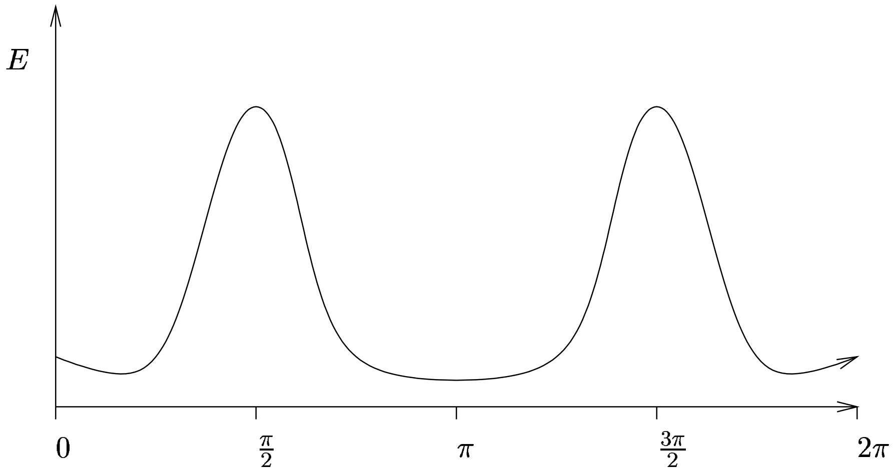


Abbildung 3.2: Energie-Darstellungsform 1.

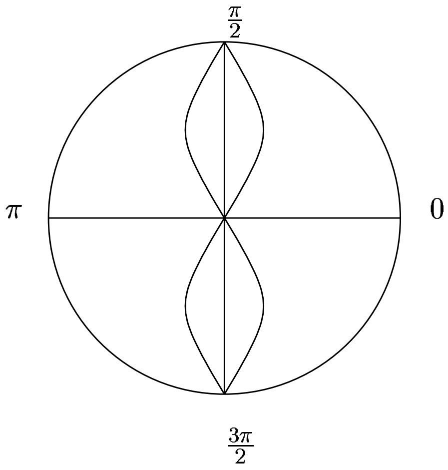


Abbildung 3.3: Energie-Darstellungsform 2.

dazu eignet, Phasenverschiebungen aufzuzeigen. Diese werden in Form einer Verschiebung in Winkelrichtung deutlich sichtbar. Die zweite Darstellungsform (Abbildung 3.3) eignet sich zur qualitativen Überprüfung. Überlagert man diese Form mit dem Ausgangsbild (Abbildung 3.4), so kann man die Qualität des Ergebnisses sofort (subjektiv) abschätzen.

In Beispiel 3.4 wird eine deutliche Übereinstimmung entlang der Richtungen sichtbar. In diesem Fall spricht man auch von der (Haupt-) Orientierung, da das Vorzeichen der Richtung keine Rolle spielt.

In dieser Arbeit werden kontextbezogen beide Darstellungsformen benutzt. Für Visualisie-

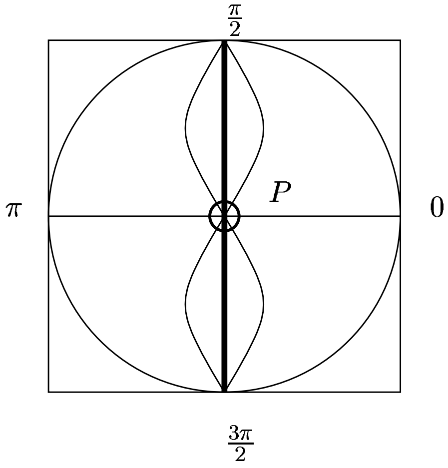


Abbildung 3.4: Beispiel einer Überlagerung.

rung, z.B. der Wirkung von Quadraturfiltern, eignet sich besonders die erste Darstellungsform. Mitunter wird auch die Ableitung der Energie in Bezug auf den Winkel genutzt, dies hilft Veränderungen in der Energie festzustellen. Es ermöglicht einem auch leichter Kanten zu erkennen.

3.1.2 Visualisierung von Filterantworten

Bei einigen in dieser Arbeit vorkommenden Verfahren werden für jeden Bildpunkt eines Bildes Merkmale berechnet. Diese Merkmale können geeignet kodiert werden (z.B. in Grauwerten), um dann wieder in einem Bild dargestellt zu werden. Diese Art der Darstellung ermöglicht es einem, schnell eine Übersicht zu gewinnen. Jähne benutzt in [14], beispielsweise bei der lokalen Orientierungsanalyse mit Hilfe des Strukturtensors, ein Kohärenzmaß für die lokale Orientierung. Dieses ermöglicht, Merkmale in Nachbarschaften zu betrachten und zu bewerten. In Abbildung 3.5 ist das von Jähne vorgestellte Kohärenzmaß für das Bild *blox* (siehe Abschnitt 4.1) dargestellt. Man erkennt deutlich die Kanten der Objekte (weiß dargestellt).

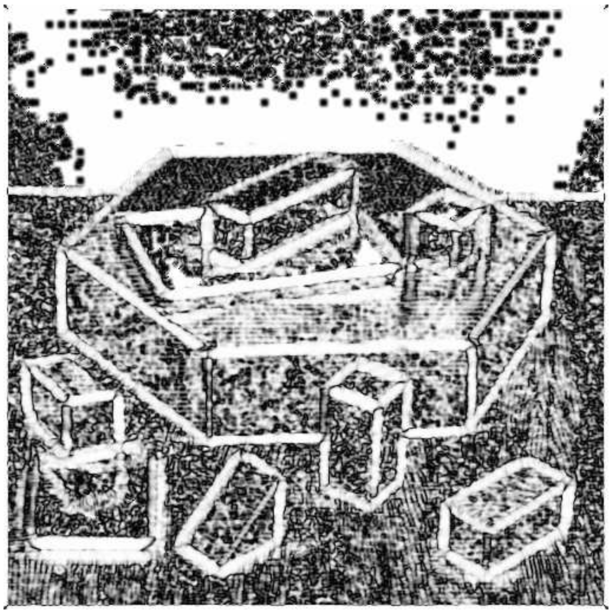


Abbildung 3.5: Darstellung eines Kohärenzmaßes für die lokale Orientierung gemäß Jähne [14].

3.2 Polarpyramiden

Das Verfahren der Polarpyramiden wurde von Yu in [26, 27, 25] vorgestellt. Es wird dazu benutzt, Verbindungspunkte (junctions) auf Kanten und Ecken zu klassifizieren. Man geht davon aus, dass bereits ein Verbindungspunkt durch einen anderen Algorithmus bereitgestellt wurde. Unter der Klassifikation bei diesem Verfahren wird die Zuordnung verstanden, die dem Verbindungspunkt eine Menge von Winkeln zuordnet. Die Elemente dieser Menge geben wieder, in welche Richtungen Geraden vom Verbindungspunkt aus wegführen.

3.2.1 Polartransformation einer Punktumgebung

In einem ersten Schritt wird eine Umgebung des Verbindungspunktes, die eine Kreisform mit dem Radius r besitzt, durch eine Polartransformation in eine rechteckige Form transformiert. Die Abzissenachsen in dieser Form entspricht dem Winkel θ und die Ordinatenachse der Distanz d vom Verbindungspunkt (siehe Abbildung 3.6). Dieses Koordinatensystem vereinfacht die weitere Verarbeitung durch lineare Filter.

Die verwendete Abbildungsvorschrift für die Polartransformation hat die Form:

$$g(\theta, d) = f_P(x + d\cos\theta, y + d\sin\theta) \tag{3.1}$$

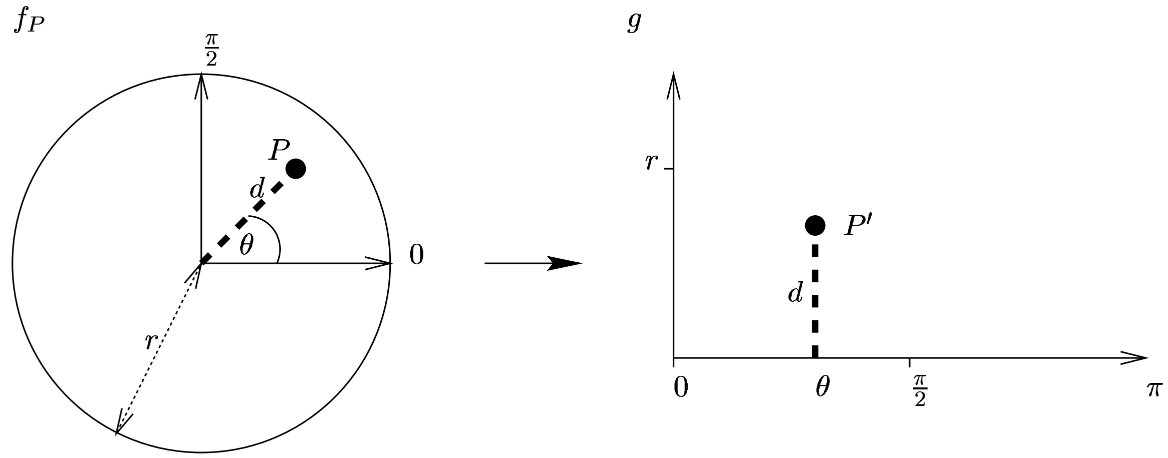


Abbildung 3.6: Polartransformation eines Punktes P

wobei $P = (x, y)$ die Koordinaten des Verbindungspunktes im Bild f_P sind, θ Werte aus dem Intervall $[0, 2\pi)$ und d Werte aus dem Intervall $[0, r]$ sind.

Um aus der polartransformierten Darstellung der Umgebung eines Verbindungspunktes P ein Maß für die *Wahrscheinlichkeit* einer Ursprungsgeraden mit dem Winkel θ zu bekommen, werden folgende Dinge benötigt:

- ein erweiterter Rechteckfilter mit der Filtermaske

$$B \overset{\text{def}}{=} \frac{1}{w(r - r_{min})} \cdot \begin{pmatrix} 1 & \cdots & 1 & \cdots & 1_r \\ \vdots & & & & \vdots \\ 1 & & \cdots & & 1_{r_{min}} \\ \vdots & & & & \vdots \\ 0 & \cdots & \boxed{0_{[\frac{w}{2}]}} & \cdots & 0 \end{pmatrix} \tag{3.2}$$

  wobei r der Umgebungsradius, r_{min} ein Radius mit der Eigenschaft $0 \leq r_{min} < r$ und w die Breite des Filters ist. Das Reziproke des Vorfaktors entspricht der Anzahl der Einsen, die in der Maske auftreten, und dient als Normierungsfaktor. Der Aufpunkt der Filtermaske liegt horizontal zentriert in der letzten Zeile der Filtermaske (in Gleichung (3.2) durch einen Kasten gekennzeichnet). Diese Filter ähneln den Wedgefiltern in [21] bei Benutzung der inversen Polartransformation.

- einer zusammengesetzten Funktion $h : [0, 2\pi) \mapsto \mathbb{R}^+$

$$h(\theta) \overset{\text{def}}{=} |g'(\theta) * G_1| \qquad \text{mit} \quad g'(\theta) \overset{\text{def}}{=} ||g(\theta, 0) * B||, \quad \theta \in [0, 2\pi) \tag{3.3}$$

wobei $||.||$ als Energiefunktion zu sehen ist. Der Filter G_1 ist eindimensional und beinhaltet als Filtermaske eine abgeleitete Gaußfunktion. Er ist ein Ableitungsoperator, der dazu dient, die wesentlichen Informationen, d.h. die Ecken und Kanten, herauszustellen.

Bei der Funktion $h(\theta)$ handelt es sich um ein mögliches Maß für die gesuchten Winkel. Berechnet man $h(\theta)$ für den gesamten Definitionsbereich $[0, 2\pi)$ der polartransformierten Umgebung des Verbindungspunktes und bestimmt die lokalen Maxima, so spiegeln die gefundenen Maxima die gesuchten Winkel für die Klassifikation wider.

3.2.2 Das Pyramidenkonzept

Das bisher beschriebene Verfahren wird von Yu in [26] um das Konzept der Pyramiden, welches von Burt und Adelson in [4] für 2D-Bilder entwickelt wurde, für 1D-Signale angepaßt. Dieses Konzept beinhaltet eine effizientere Berechnung von Faltungen mit größeren Glättungsmasken, welches in Hinsicht auf Bandbreiten größerer Strukturen nützlich ist. Die Grundlegende Idee dabei ist, das Abtasttheorem konsequent zu nutzen. Liegen alle im Glättungsfilter vorkommenden Frequenzen unterhalb der halben Grenzfrequenz (größte mögliche Frequenz), so kann eine Glättung mit einem doppelt so großen Glättungsfilter dadurch erreicht werden, dass man nur jeden zweiten Punkt einer Zeile und nur jede zweite Zeile aus dem zu filternden Bild betrachtet. Das Abtasttheorem stellt dabei die korrekte Rekonstruktion der Filterantwort durch Interpolation sicher. Wie in [1] beschrieben ist, eignen sich besonders Gaußfilter als Glättungsfilter, die der eben beschriebenen Eigenschaft genügen. Um die Anwengung des Pyramidenkonzeptes zu ermöglichen, wird nun der Rechteckfilter (3.2) in folgender Weise modifiziert:

$$B_G \overset{\text{def}}{=} \frac{B^P}{||B^P||} \quad ,$$

$$\text{wobei} \quad B^P \overset{\text{def}}{=} \begin{pmatrix} g^P(\theta)_r \\ \vdots \\ g^P(\theta)_{r_{min}} \\ 0 \end{pmatrix} \tag{3.4}$$

$$\text{und} \qquad g^P(\theta) \overset{\text{def}}{=} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{\theta^2}{2\sigma^2}}, \quad \theta \in [-\tfrac{w}{2}, \tfrac{w}{2}]$$

Die Varianz σ der Gaußfunktion in (3.4) wird mit

$$\sigma = \frac{1}{5}\sqrt{\frac{w}{2}} \tag{3.5}$$

angegeben, damit der wesentliche Bereich des Gaußfilters mit einem Energieanteil größer als 99% in der Filtermaske berücksichtigt ist. Wendet man diesen Filter, wie er in Gleichung (3.3) angegeben ist, an, so erhält man ein eindimensionales Signal bzw. die unterste Stufe der Pyramide. Nun lässt sich das Pyramidenkonzept entwickeln. Die zweite Stufe der Pyramide ergibt sich aus der ersten, indem jeweils fünf benachbarte Werte (die Anzahl fünf hält den Berechnungsaufwand niedrig und es ist trotzdem eine adequate Glättung möglich) mit einer geeigneten Gewichtung zu einem neuen Wert der zweiten Stufe verrechnet werden. Die Berechnung entspricht einer Glättung mit einer Unterabtastung. Wie der Abbildung 3.7 entnommen werden kann, halbiert sich die Anzahl der Werte in der nächsten Stufe. Führt man das Verfahren sukzessiv fort, so erhält man ein Gebilde, welches einer Pyramide ähnelt.

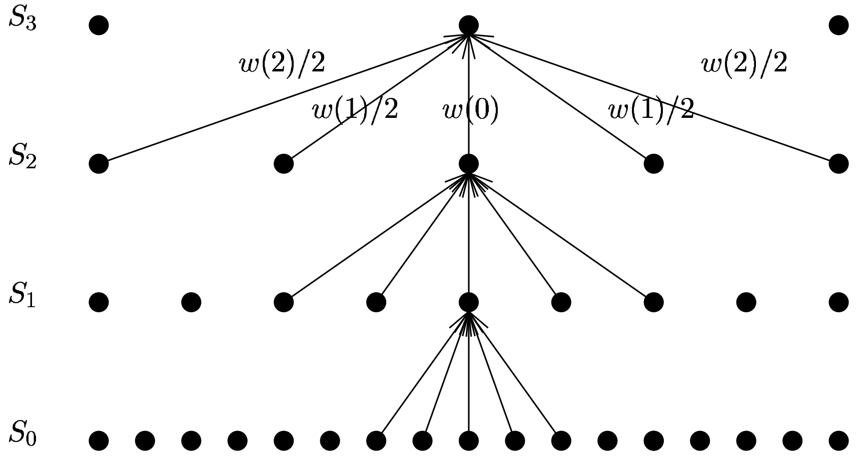


Abbildung 3.7: Veranschaulichung einer Gaußpyramide aus [4] ohne Berücksichtigung der Periodizität der betrachteten Winkel.

Formal lässt sich die Pyramide durch iterative Anwendung der Gleichung

$$h_l(i) = \sum_{n=-2}^{2} w(n) h_{l-1}(2i+n) \tag{3.6}$$

beschreiben. Bei h_l handelt es sich um die Wertefunktion h in der l-ten Schicht der Pyramide, wobei l kleiner der Anzahl der Schichten N ist. Der Werteindex i ist beschränkt durch die Anzahl der Werte in der entsprechenden Schicht h_l. Die Gewichtsfunktion w soll die drei Eigenschaften

1. $\sum_{n=-2}^{2} w(n) = 1$, d.h. w ist normalisiert

2. $w(n) = w(|n|)$, d.h. w ist symmetrisch

3. $w(0) + 2w(1) = 2w(2)$, d.h. w ist gleichverteilt

besitzen. Diese Eigenschaften sind erfüllt, falls die Gewichtsfunktion w wie folgt angegeben werden kann

$$w(0) = a, \quad w(1) = w(-1) = \frac{1}{4}, \quad w(2) = \frac{1}{4} + \frac{a}{2}. \tag{3.7}$$

Geht man davon aus, dass es ein $M \in \mathbb{N}$ gibt, sodass für die Anzahl der Werte c von $h = h_0$ gilt $c = M2^N$ (Aufgrund der Periodizität sind betrachte man den rechten Rand nicht), so lässt sich die Pyramidenstruktur gemäß Gleichung (3.6) für N Schichten erstellen. Um die Wirkung eines Glättungsfilters auf einer Schicht der Wertepyramide korrekt darzustellen, muss jede Schicht mit Hilfe einer Gaußinterpolation wieder auf die Größe der ursprünglichen Werteanzahl gebracht werden. Man beginnt mit einer Schicht l, die durch die Gleichung (vgl. Gleichung (3.6))

$$h_{l-1}(i) = \sum_{n=-2}^{2} w(n) h_l(\frac{i-n}{2}) \tag{3.8}$$

zu größeren Schichten interpoliert wird. So erhält man schließlich eine interpolierte Schicht, deren Größe der Ausgangsschicht entspricht. Eine Erweiterung der Schicht l zu dieser Größe wird in der Regel nur für Visualisierungszwecke gebraucht. Die Pyramide wird als Gaußpyramide bezeichnet, da die Gewichtsfunktion w gaußähnlich ist. Jede Schicht der Gaußpyramide entspricht der Filterantwort des Bildes mit einem, der Pyramidenstufe entsprechenden, Gaußfilter.

Yu benutzt in [25] Gaußpyramiden, die durch verschieden große Gewichtsfunktionen und verschiedene Unterabtastraten erzeugt werden:

Unterabtastung	*Faktor*	w_0	w_1	w_2	w_3	w_4	w_5
2	1/16	6	4	1			
3	1/264	82	66	22	3		
5	1/5120	1022	950	725	299	74	1

. (3.9)

Die veränderte Rechteck-Filtermaske für die Erzeugung der Pyramide, hat dann z.B., für eine Unterabtastung von zwei Pixel, die Form:

$$B_G \overset{\text{def}}{=} \frac{1}{16}(1 \quad 4 \quad 6 \quad 4 \quad 1)\,. \tag{3.10}$$

3.3 Steuerbare Filter

Ein weiteres Verfahren, welches ebenfalls die Klassifikation von Verbindungspunkten behandelt, wurde von Freeman und Adelson in [10] vorgeschlagen. Ausgehend von einem bereits lokalisierten Verbindungspunkt, wird dort ein Verfahren angegeben, welches die Idee verfolgt, mittels einer *kleinen* Menge von Basisfiltern (siehe Steuerbarkeit 2.5) den Verbindungspunkt zu bearbeiten. Diese Filter unterscheiden sich in ihrer Orientierung (vgl. Abschnitt 2.2.2) und berechnen die lokale Orientierungsenergie. Fehlende Orientierungen werden aus den Signalantworten der Basisfilter interpoliert (siehe Theorem 1). Als Grundlage für die Filter, wird die 4-te Gaußfunktionsableitung und die Hilberttransformierte benutzt (siehe Abbildung 3.8). Die Funktionsgleichungen für den zweidimensionalen Fall lauten

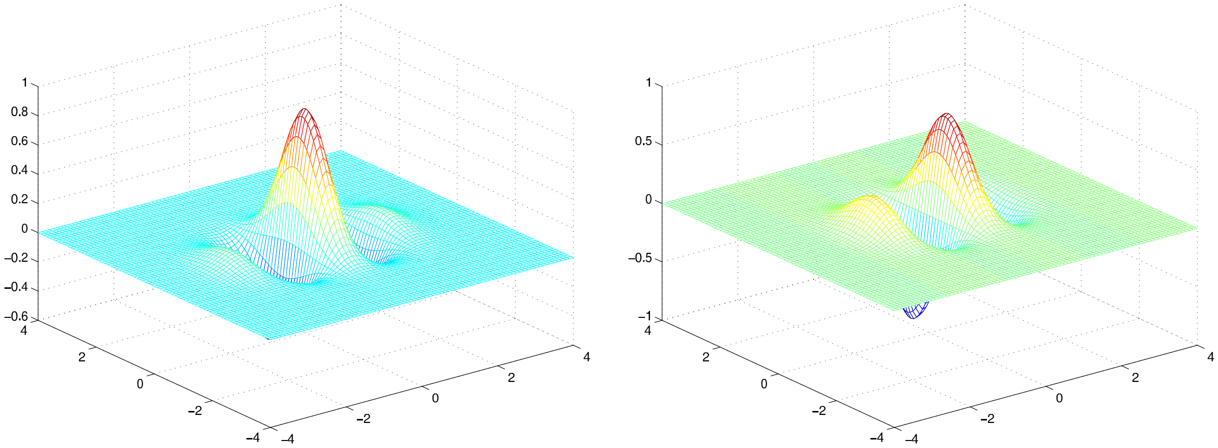


Abbildung 3.8: Eine vierte Gaußrichtungsableitung und ihre Hilberttransformierte

$$G_4(x,y) = (4 - 32x^2 + 16x^4)e^{-(x^2+y^2)} \tag{3.11}$$

bzw. in der energienormierten Form

$$G_4^N(x,y) = (0.9344 - 3.738x^2 + 1.246x^4)e^{-(x^2+y^2)} \tag{3.12}$$

Die Hilberttransformierte hat dann die Darstellung

$$H_4^N(x,y) = (2.858x - 2.982x^3 + 0.3975x^5)e^{-(x^2+y^2)}. \tag{3.13}$$

Unter Ausnutzung der Theoreme 2 und 4 kann man die folgenden Steuerungsgleichungen für G_4^θ und H_4^θ herleiten. Für G_4^θ gilt:

$$G_4^\theta = k_a(\theta)G_{4a} + k_b(\theta)G_{4b} + k_c(\theta)G_{4c} + k_d(\theta)G_{4d} + k_e(\theta)G_{4e} \tag{3.14}$$

wobei die einzelnen Basisfunktionen $G_{4x}, x \in \{a,b,c,d,e\}$ sich folgendermaßen zusammensetzen,

$$\begin{aligned}
G_{4a} &= 1.246(0.75 - 3x^2 + x^4)e^{-(x^2+y^2)} \\
G_{4b} &= 1.246(-1.5x + x^3)ye^{-(x^2+y^2)} \\
G_{4c} &= 1.246(x^2 - 0.5)(y^2 - 0.5)e^{-(x^2+y^2)} \\
G_{4d} &= 1.246(-1.5y + y^3)xe^{-(x^2+y^2)} \\
G_{4e} &= 1.246(0.75 - 3y^2 + y^4)e^{-(x^2+y^2)}
\end{aligned} \tag{3.15}$$

und die zugehörigen Interpolationsfunktionen $k_x, x \in \{a,b,c,d,e\}$ lauten

$$\begin{aligned}
k_a(\theta) &= cos^4(\theta) \\
k_b(\theta) &= -4cos^3(\theta)sin(\theta) \\
k_c(\theta) &= 6cos^2(\theta)sin^2(\theta) \quad . \\
k_d(\theta) &= -4cos(\theta)sin^3(\theta) \\
k_e(\theta) &= sin^4(\theta)
\end{aligned} \tag{3.16}$$

Die Steuerungsgleichung für die Hilberttransformierte H_4^θ lautet:

$$H_4^\theta = k_a(\theta)H_{4a} + k_b(\theta)H_{4b} + k_c(\theta)H_{4c} + k_d(\theta)H_{4d} + k_e(\theta)H_{4e} + k_f(\theta)H_{4f} \tag{3.17}$$

wobei die einzelnen Basisfunktionen $G_x, x \in \{a,b,c,d,e,f\}$ sich folgendermaßen zusammensetzen,

$$\begin{aligned}
H_{4a} &= 0.3975(7.189x - 7.501x^3 + x^5)e^{-(x^2+y^2)} \\
H_{4b} &= 0.3975(1.438 - 4.501x^2 + x^4)ye^{-(x^2+y^2)} \\
H_{4c} &= 0.3975(x^3 - 2.225x)(y^2 - 0.6638)e^{-(x^2+y^2)} \\
H_{4d} &= 0.3975(y^3 - 2.225y)(x^2 - 0.6638)e^{-(x^2+y^2)} \\
H_{4e} &= 0.3975(1.438 - 4.501y^2 + y^4)xe^{-(x^2+y^2)} \\
H_{4f} &= 0.3975(7.189y - 7.501y^3 + y^5)e^{-(x^2+y^2)}
\end{aligned} \tag{3.18}$$

und die zugehörigen Interpolationsfunktionen $k_x, x \in \{a,b,c,d,e,f\}$ lauten

$$\begin{aligned}
k_a(\theta) &= cos^5(\theta) \\
k_b(\theta) &= -5cos^4(\theta)sin(\theta) \\
k_c(\theta) &= 10cos^3(\theta)sin^2(\theta) \\
k_d(\theta) &= -10cos^2(\theta)sin^3(\theta) \quad . \\
k_e(\theta) &= 5cos(\theta)sin^4(\theta) \\
k_f(\theta) &= -sin^5(\theta)
\end{aligned} \tag{3.19}$$

Die Anwendung dieser Filter (siehe Abbildung 3.9) erfolgt mit der Hilfe von Theorem 1. Die orientierte Energiefunktion kann dann für den Winkel $\theta \in [0, 2\pi)$ kann durch die Gleichung

$$E_4(\theta) = (G_4(\theta))^2 + (H_4(\theta))^2 \tag{3.20}$$

berechnet werden.

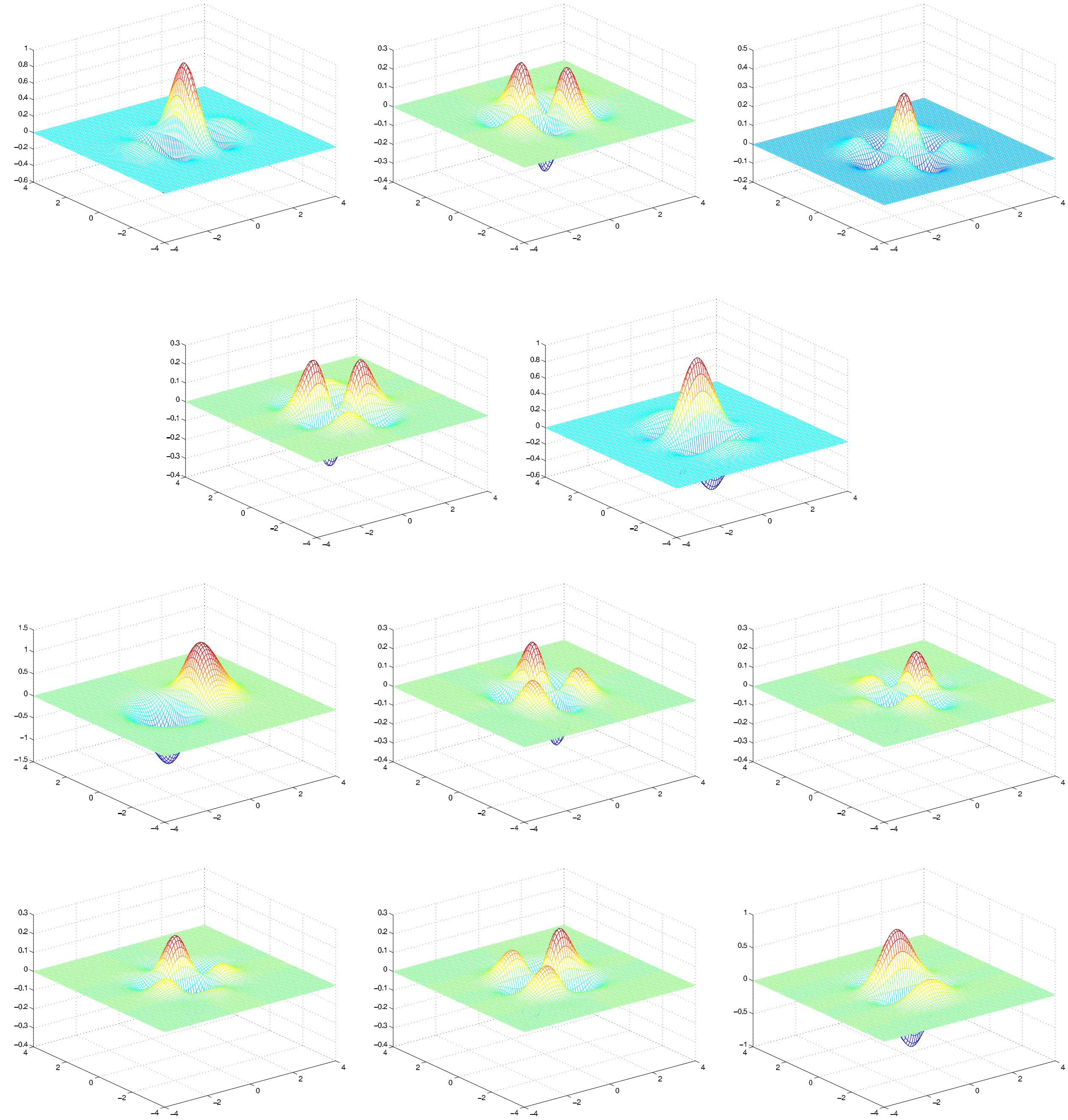


Abbildung 3.9: Basisfunktionen für die vierte Gaußableitung (die beiden oberen Reihen) und Basisfunktionen für die Hilberttransformierte (die beiden unteren Reihen).

3.4 Wedgefilter

Dieser Ansatz beschäftigt sich mit der lokalen Analyse von Richtungen und Verbindungspunkten. Die zur Analyse benutzten Filter sollen dabei asymmetrisch und steuerbar bzgl. eines Winkels θ sein (vgl. 3.3). Ziel bei der Anwendung der Filter ist es, eine Karte zu erstellen, die die Energie (vgl. 2. Darstellungsform) entsprechend der Winkelorientierungen darstellt (siehe Simoncelli [21]).

Bei der Konstruktion von Wedgefiltern wird davon ausgegangen, dass die zu generierenden Filter als Fourierreihe darstellbar sind. Dabei wird unterschieden, ob die Fourierreihen gerade bzw. ungerade sind, sich also darstellen lassen als

$$h_e(\phi) = \sum_{n=1}^{N} w_n \cos(n\phi), \qquad h_o(\phi) = \sum_{n=1}^{N} w_n \sin(n\phi). \tag{3.21}$$

Für die resultierende Energiedarstellung ist wichtig, dass $h_e(\phi)$ die hilberttransformierte Version von $h_o(\phi)$ ist und umgekehrt. Dies wird in der Energiefunktion

$$E(\phi) = h_e^2(\phi) + h_o^2(\phi) \tag{3.22}$$

ausgenutzt.

Diese Energiefunktion wird bzgl. der Gewichte w maximiert. Dazu wird das Prinzip der kleinsten Quadrate benutzt. Es wird eine Funktion P in Abhängigkeit von den zu bestimmenden Gewichten angegeben,

$$P(w_1, \ldots, w_n) = \int \Lambda^2(\phi) E(\phi) d\phi \tag{3.23}$$

wobei $\Lambda(\phi) = \phi$ eine monoton wachsende Gewichtsfuntion ist. In der Praxis wird dieses Problem mit der Hilfe der Matrixalgebra berechnet.

Zunächst werden einige Lemmata benötigt, die das Verständnis des nachfolgenden Satzes vereinfachen. Sie beinhalten Umformungsschritte, die innerhalb der Matrixalgebra ausgeführt werden. Die Beweise der Lemmata sind im Anhang 6 nachzulesen. Dazu gilt im Folgenden:

- $w = (w_n)$ sind die Gewichte
- $\Lambda = diag(\lambda(\phi_1), \ldots, \lambda(\phi_m))$, mit $\phi_m = 2\pi m/M$
- $m \times n -$ Matrix $\quad S = (S_{mn}) = (\sin(2\pi nm/M))$
- $m \times n -$ Matrix $\quad C = (C_{mn}) = (\cos(2\pi nm/M))$.

Das folgende kleine Lemma beschreibt, wie sich eine mit M Werten abgetastete Kosinusfunktion im Interval $[0, 2\pi)$ mit den gesuchten Gewichten verknüpfen lässt.

Lemma 3

$$\Lambda C w = \begin{pmatrix} \lambda(\phi_1) \sum_{n=1}^{N} w_n \cos(2\pi n \cdot 1/M) \\ \vdots \\ \lambda(\phi_M) \sum_{n=1}^{N} w_n \cos(2\pi n \cdot M/M) \end{pmatrix} \tag{3.24}$$

Vergleichbares gilt für die abgetastete Sinusfunktion. Das folgende Lemma beschreibt den Anteil der geraden Filter an der gesamten Energiefunktion. Für die ungeraden Filter stellt man Analoges fest.

Lemma 4

$$\sum_{m=1}^{M} \lambda^2(\phi_m) h_e^2(\phi_m) = |\Lambda C w|^2 \tag{3.25}$$

Jetzt ist es möglich die Energiefunktion (siehe Gleichnung 3.22) entsprechend zu formulieren.

Lemma 5

$$\sum_{m=1}^{M} \lambda^2(\phi_m) E(\phi_m) = |\Lambda C w|^2 + |\Lambda S w|^2 \tag{3.26}$$

Mit dem nachfolgenden Satz können die Gewichte berechnet werden, die maximal bzgl. der Energiefunktion sind.

Satz 1

$$\sum_{m=1}^{M} \lambda^2(\phi_m) E(\phi_m) = w^t [C^t \Lambda^t \Lambda C + S^t \Lambda^t \Lambda S] w \tag{3.27}$$

Beweis

$$\begin{aligned} |\Lambda C w|^2 + |\Lambda S w|^2 &= <\Lambda C w, \Lambda C w> + <\Lambda S w, \Lambda S w> \\ &= (\Lambda C w)^t \Lambda C w + (\Lambda S w)^t \Lambda S w \\ &= (C w)^t \Lambda^t \Lambda C w + (S w)^t \Lambda^t \Lambda S w \\ &= w^t C^t \Lambda^t \Lambda C w + w^t S^t \Lambda^t \Lambda S w \\ &= w^t [C^t \Lambda^t \Lambda C + S^t \Lambda^t \Lambda S] w \end{aligned}$$

•

Der Gewichtsvektor w ist dann der minimale Eigenvektor der Matrix

$$[C^t \Lambda^t \Lambda C + S^t \Lambda^t \Lambda S]. \tag{3.28}$$

Erhöht man die Anzahl der Summanden in den Fourierreihen h_e und h_o, so erhält man eine bessere Unterscheidung der gerichteten Energien. Allerdings erhöht sich auch die Anzahl der Basisfilter.

Bisher wurden die Wedgefilter lediglich für einen Winkel betrachtet. Um diesen Filter zum Steuern benutzen zu können, wird wieder eine Menge von Basisfiltern erzeugt. Es handelt sich hier um die spezielle Steuerbarkeit.

Die um α gedrehten geraden bzw. ungeraden Filter haben die Form

$$h_e(\theta - \alpha) = \sum_{n=1}^{N} w_n \cos(\theta - \alpha) \tag{3.29}$$

und

$$h_o(\theta - \alpha) = \sum_{n=1}^{N} w_n \sin(\theta - \alpha). \tag{3.30}$$

Gleichungen (3.29) und (3.30) können zu einem Skalarprodukt umgeformt werden

$$h_e(\theta - \alpha) = K_e(\alpha) \cdot f(\theta) \tag{3.31}$$

und

$$h_o(\theta - \alpha) = K_o(\alpha) \cdot f(\theta). \tag{3.32}$$

Dabei sind die Vektoren $f(\theta), K_e(\alpha)$ und $K_o(\alpha)$ wie folgt aufgebaut:

$$f(\theta) = \begin{pmatrix} w_1 \cos(\theta) \\ w_1 \sin(\theta) \\ \vdots \\ w_n \cos(\theta) \\ w_n \sin(\theta) \end{pmatrix}, K_e = \begin{pmatrix} \cos(\alpha) \\ \sin(\alpha) \\ \vdots \\ \cos(N\alpha) \\ \sin(N\alpha) \end{pmatrix}, K_o = \begin{pmatrix} -\sin(\alpha) \\ \cos(\alpha) \\ \vdots \\ -\sin(N\alpha) \\ \cos(N\alpha) \end{pmatrix}. \tag{3.33}$$

Um eine Basismenge zu erhalten, werden rotierte Wedgefilter für die Winkel

$$\{\alpha_j = \frac{\pi j}{n} | 1 \leq j \leq 2n\} \tag{3.34}$$

betrachtet.

Man kann jetzt für alle α_j die Gleichungen (3.29) und (3.32) in Gleichungssysteme zusammenfassen:

$$h_e(\theta - \alpha) = k_e(\alpha) K h'(\theta) \qquad h_o((\theta - \alpha) = k_o(\alpha) K h'(\theta) \tag{3.35}$$

wobei

$$K = \begin{pmatrix} \cos(\alpha_1) & sin(\alpha_1) & \cdots & \cos(N\alpha_1) & \sin(N\alpha_1) \\ -\sin(\alpha_1) & \cos(\alpha_1) & \cdots & -\sin(N\alpha_1) & \cos(N\alpha_1) \\ \vdots & \vdots & \ddots & \vdots & \\ \cos(\alpha_N) & sin(\alpha_N) & \cdots & \cos(N\alpha_N) & \sin(N\alpha_N) \\ -\sin(\alpha_N) & \cos(\alpha_N) & \cdots & -\sin(N\alpha_N) & \cos(N\alpha_N) \end{pmatrix}, \tag{3.36}$$

$$h'(\phi) = \begin{pmatrix} h_e(\phi - \alpha_1) \\ h_o(\phi - \alpha_1) \\ \vdots \\ h_e(\phi - \alpha_N) \\ h_o(\phi - \alpha_N) \end{pmatrix}, \text{mit} \qquad \alpha_n = 2\pi n/N, 1 \leq n \leq N \tag{3.37}$$

sind.

Die Basismenge ist implizit im Vektor $h'(\theta)$ enthalten. Er beinhaltet alle rotierten Versionen des nichtgedrehten Wedgefilters.

Bisher wurde der Wedgefilter nur in angulärer Richtung betrachtet, d.h. der Radiusanteil in der Polardarstellung war konstant. Um die typische Keilform des Wedgefilters zu erzeugen wird in [21] die radial-Funktion $g : \mathbb{R}^+ \mapsto \mathbb{R}^+$

$$g(r) = \begin{cases} 0 & r < R_1 \\ \frac{1}{2}(1 - \cos(\pi(r - R_1)/\delta_1)) & R_1 \leq r < R_1 + \delta_1 \\ \frac{1}{2}(1 + \cos(\pi(r + \delta_2 - R_2)/\delta_2)) & R_2 - \delta_2 \leq r < R_2 \\ 0 & R_2 \leq r \\ 1 & R_1 + \delta_1 \leq r < R_2 - \delta_2 \end{cases} \tag{3.38}$$

vorgeschlagen. Die Werte für R_1 und R_2 geben die Radiusbeschränkungen vor, die Winkel δ_1 und δ_2 dienen zur Konstruktion weicher Übergänge.

Simoncelli hat in [21] für die radial-Funktion die Werte $R_1 = 2, \delta_1 = 2$ und $R_2 = 30, \delta_2 = 3$ benutzt. Er berechnet Wedgefilter der Größenordnung 63^2 Pixel. Diese reduziert er iterativ durch Glättung und Unterabtastung jeweils um die Hälfte, bis diese eine Größenordnung von 7^2 Pixel haben. Eine Filterbasis für fünf Richtungen ist in Abbildung 3.10, jedes Basiselement besteht aus einem Filterpaar $(h_e(\theta), h_o(\theta))$. Der Anschaulichkeit halber sind hier die Filter in ihrer ursprünglichen Größenordnung, d.h. 63^2 Pixel, dargestellt.

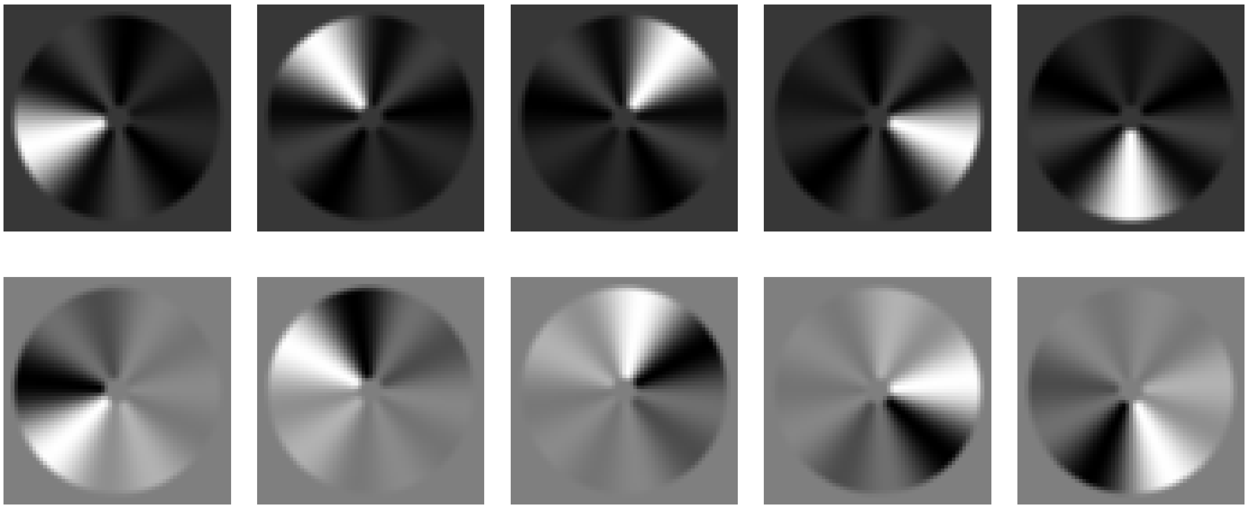


Abbildung 3.10: Wedgebasisfilter für fünf Richtungen (obere Reihe gerade Filteranteile, untere Reihe ungerade Filteranteile).

3.5 Phasenkongruenz

Die Bedeutung der Phase für die Strukturen in Bildern wird z.B. sehr anschaulich von Jähne in [14] dargestellt. Es wird festgestellt, dass die für die Struktur wichtige Information in der Phase enthalten ist. Die Amplitude gibt lediglich das Vorhandensein einer Struktur wieder, aber nicht deren Ort. Strukturen zeichnen sich durch feste Phasenbeziehungen aus. Das von Kovesi in [15] vorgestellte Verfahren nutzt diese Eigenschaft der Strukturen aus, um Strukturen wie Ecken und Kanten in Bildern zu lokalisieren.

3.5.1 Der Begriff der Phasenkongruenz

Geht man davon aus, dass man ein eindimensionales Signal in Form einer Fourierreihe vorliegen hat, so versteht man unter der Phasenkongruenz (phase congruency) für ein Strukturmerkmal an der Stelle x, dass die für die Struktur an der Stelle x entscheidenden Anteile der Fourierreihe in Phase liegen. Klassische Beispiele für Phasenkongruenz sind die Fourierreihen:

$$\begin{aligned} square(x) &= \frac{4}{\pi}\sum_{n=0}^{N}\frac{\sin((2n+1)x)}{2n+1} \\ triangle(x) &= \frac{4}{\pi}\sum_{n=0}^{N}\frac{\sin((2n+1)x)}{(2n+1)^2\cdot(-1)^n} \end{aligned}$$

deren Graphen im Intervall $[-\pi, \pi]$ in Abbildung 3.11 dargestellt sind. Wie man in der Abbildung

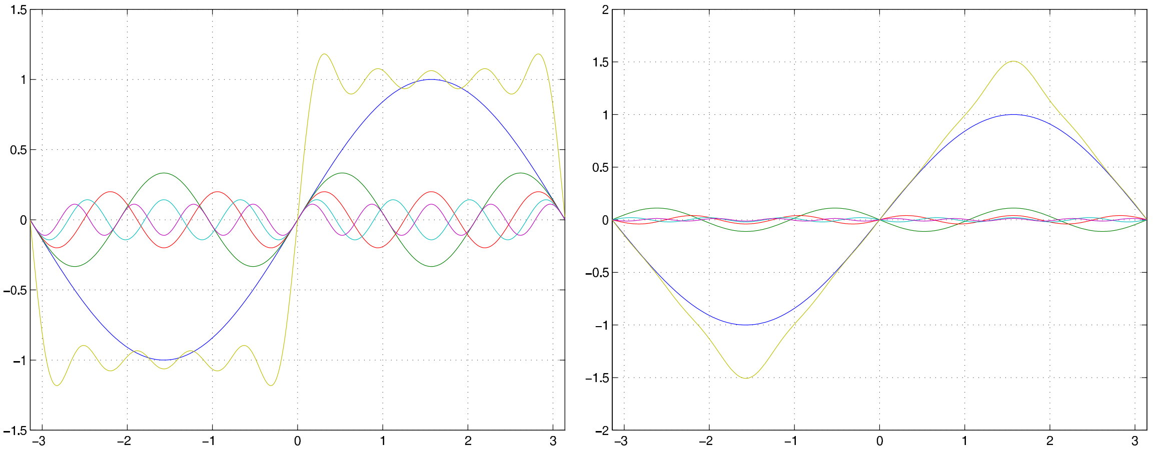


Abbildung 3.11: Graphen von $square(x)$ und $triangle(x)$

erkennt, liegen für $square(x)$ an der Stelle $x_0 = 0$, die ersten fünf Reihenglieder in Phase, d.h. ihre Phasen sind kongruent. Für $triangle(x)$ gibt es drei solche Stellen: $x_0 = \frac{-\pi}{2}; x_1 = 0; x_2 = \frac{\pi}{2}$. An diesen Stellen spricht man von Phasenkongruenz.

Formal wird ein Maß für Phasenkongruenz gemäß der Definition von Morrone und Owens [18] benutzt. Ausgehend von der Fourierreihe eines Signals

$$s(x) = \sum_{j=-N}^{N} a_j e^{ijx} \tag{3.39}$$

wird folgende Funktion definiert:

$$PC(x) = \underset{\Phi(x) \in [0, 2\pi]}{max} \frac{\sum_{j=-N}^{N} a_j \cos(\Phi_j(x) - \Phi(x))}{\sum_{j=-N}^{N} a_j} \tag{3.40}$$

wobei a_j der j-te Fourierkoeffizient ist. Die lokale Phase der j-ten Fourierkomponente für die Position x wird durch $\Phi_j(x)$ gegeben. Bei $\Phi(x)$ handelt es sich um das amplitudengewichtete Mittel der Phasen an der Stelle x.

Die Funktion PC ist ein Maß für die Phasenkongruenz, deren Maxima Strukturmerkmale im Signal beschreiben. Da die Gleichung (3.40) in der Praxis bei der Berechnung Probleme bereitet (Effizienz- und Genauigkeitsprobleme), wird als Alternative von Venkatesh und Owens in [24] die Berechnung der Phasenkongruenz über die lokale Energie des Signals vorgestellt. Die lokale Energie wird mittels der Hilberttransformation des Signals bestimmt:

$$E(x) = \sqrt{F^2(x) + H^2(x)}. \tag{3.41}$$

$F(x)$ entspricht dem Signal abzüglich des Mittelwertes, $H(x)$ ist die Hilberttransformierte von $F(x)$. Folgender Satz von Venkatesh und Owens rechtfertigt diese Art der Berechnung.

Satz 2 *Sei $x \in \mathbb{R}$. Dann gilt für ein Signal in Fourierreihendarstellung (3.39) an der Stelle x:*

$$\begin{aligned} E(x) &= PC(x) \cdot \sum_{j=-n}^{N} a_j \\ &= \underset{\Phi(x) \in [0, 2\pi]}{max} \sum_{j=-N}^{N} a_j \cos(\Phi_j(x) - \Phi(x)) \end{aligned} \tag{3.42}$$

Die lokale Energie ist direkt-proportional zur Phasenkongruenz-Funktion PC.

3.5.2 Berechnung der Phasenkongruenz mittels Wavelets

Um die Phase Kongruenz zu berechnen, wählt Koresi in [15] den Weg der Wavelettransformation. Es werden nichtorthogonale Wavelets benutzt, die in Form symmetrischer und nichtsymmetrischer Quadraturfilterpaare angewendet werden (vgl. Hilberttransformation zur Berechnung der lokalen Energie). Kovesi wählt in Anlehnung an Morlet [13] eine logarithmische Garborfunktion

$$G(\omega) = e^{\frac{-\log^2 \frac{\omega}{\omega_0}}{2 \log^2 \frac{k}{\omega_0}}}, \tag{3.43}$$

wobei ω_0 die Mittelwertfrequenz ist. Das Verhältnis $\frac{k}{\omega_0}$ gibt die zu betrachtende Bandbreite an. Die benutzten gerade–symmetrischen und ungerade–symmetrischen Wavelets in der Skala n sind gegeben durch:

$$\begin{aligned} M_n^e(x) &= \cos(x) \cdot \mathcal{F}^{-1}\{G\}(x) \\ M_n^o(x) &= \sin(x) \cdot \mathcal{F}^{-1}\{G\}(x) \end{aligned}$$

Die Antworten dieser Filter in Bezug auf die Bildfunktion an der Stelle x sind durch die Faltungen

$$\begin{aligned} e_n(x) &= f(x) * M_n^e(x) \\ e_o(x) &= f(x) * M_n^o(x) \end{aligned}$$

gegeben. Für die Berechnung der Phase Kongruenz wird nun der Satz 2 benutzt. Bei der Berechnung werden die Schätzungen

$$F(x) \simeq \sum_n e_n(x)$$

$$\begin{aligned} H(x) &\simeq \sum_n o_n(x) \\ \sum_n a_n(x) &\simeq \sum_n \sqrt{e_n(x)^2 + o_n(x)^2} \end{aligned}$$

verwendet. Mit Satz 2 ergibt sich für die Phasenkongruenz:

$$PC(x) = \frac{E(x)}{\sum_n a_n(x)}, \tag{3.44}$$

wobei $E(x) = \sqrt{F(x)^2 + H(x)^2}$ ist.

Bisher wurden lediglich Signale in einer Dimension betrachtet. Um die Phasenkongruenz berechnen zu können, benötigt man eine Phasenverschiebung um π im Signal. Dieses wurde im Eindimensionalen durch ungerade- bzw. gerade–symmetrische Filter erreicht. Im zweidimensionalen Fall kann man jedoch keine rotierten ungerade- bzw. gerade-symmetrischen Filter herstellen. Man ist gezwungen, die eindimensionale Phasenkongruenz zu berechnen und diese dann in geeigneter Form zu einer zweidimensionalen Repräsentation zu vereinigen.

Die bisher vorgestellten eindimensionalen Filter können auf einfache Weise zu zweidimensionalen erweitert werden. Dazu werden eindimensionale Gaußfilter verwendet, die senkrecht mit den bisherigen Filtern gefaltet werden. Die Gaußfilter werden als *spreading functions* benutzt. Die entstehenden Filter sind separabel. Durch die Wahl von Gaußfiltern werden die Phasen des Signals nicht beeinflußt. Diese Filter werden um einen Ursprung in gleichmäßiger Weise angeordnet, um entsprechende Orientierungen zu detektieren. Kovesi hat in [15] gezeigt, dass eine gleichmäßige Verteilung in 30^o–Schritten ein geeigneter Kompromiß zwischen Anzahl der Orientierungen und Genauigkeit der Detektion ist. Die Phasenkongruenz für den zweidimensionalen Fall wird durch Summation erreicht:

$$PC_2(x) = \frac{\sum_\phi \sum_n E(x)}{\sum_\phi \sum_n A_{n\phi}(x)} \tag{3.45}$$

wobei ϕ der Winkel der Orientierung ist.

3.6 Strukturtensor

In diesem Abschnitt geht es um die Detektion einer Hauptorientierung, d.h. ausgehend von einem Punkt x in einer Umgebung u soll der Winkel θ gefunden werden, der eine Linie vom Punkt x aus beschreibt, auf der die Summe der Gradienten maximal wird.

Dieses Verfahren kann deshalb nur für einfache Nachbarschaften benutzt werden. Komplexe Nachbarschaften, wie z.B. zwei Linien von einem Punkt x aus, können nicht erfasst werden. Es werden Tensoren erster Ordnung benutzt. Der Zusatz *erster Ordnung* besagt, dass in den Tensoren nur Ableitungen erster Ordnung vorkommen.

Ein Tensor zweiter Stufe ist eine Bilinearform $f : \mathbb{R} \times \mathbb{R} \longrightarrow \mathbb{R}$. Eine solche Form wird hier verwendet, um eine lokale Hauptorientierung für einen Punkt x in einer Umgebung u zu finden. Man möchte also die Umgebung eines Punktes auf einen Orientierungswinkel abbilden.

Für eine einfache Nachbarschaft in einer Bildfunktion f gilt die Bedingung:

$$f(x) = f(< x, n >), \tag{3.46}$$

wobei n ein Einheitsvektor ist, der auf einer Linie mit entsprechender Orientierung und konstanten Grauwerten steht.

Die Projektion des Vektors x auf den Einheitsvektor n liefert eine Abhängigkeit, die besagt, dass die Grauwerte entlang der Richtung von n sich durch eine skalare Größe, der Koordinate, unterscheiden. Es gilt:

$$\nabla f(< x, n >) = \begin{pmatrix} \frac{\partial f(<x,n>)}{\partial x_1} \\ \frac{\partial f(<x,n>)}{\partial x_2} \end{pmatrix} = \begin{pmatrix} k_1 f'(< x, n >) \\ k_2 f'(< x, n >) \end{pmatrix} = n f'(< x, n >) \tag{3.47}$$

Mit f' wird die Ableitung von f nach der skalaren Größe $< x, n >$ bezeichnet. Gleichung (3.47) zeigt, dass der Gradient in Richtung von n liegt.

In der Regel liegt jedoch nicht eine einfache Nachbarschaft vor, da z.B. durch Rauschen die Bedingung (3.46) nicht ausreichend ist. Es empfiehlt sich eine Optimierungsstrategie zu benutzen, die mittels eines Maßes die optimale Orientierung findet. Dabei ist die optimale Orientierung diejenige, die die geringste Abweichung von der Gradientenrichtung hat. Die optimale Orientierung spiegelt sich in der Gleichung

$$< \nabla f, n >^2 = |\nabla f|^2 \cos^2(\arg(\nabla f, n)) \tag{3.48}$$

wider. Der Ausdruck $< \nabla f, n >^2$ ist proportional zum Kosinusquadrat des Winkels zwischen ∇f und n. Weiterhin ist er maximal, wenn ∇f parallel zu n ist. In einer Umgebung u eines Punktes x, die durch die Fensterfunktion w gegeben ist, muss für eine Bildfunktion f der folgende Ausdruck maximiert werden:

$$\int w(x-x')(<\nabla f(x'), n>)^2 d^2x'. \tag{3.49}$$

Der zu maximierende Ausdruck (3.49) kann mit Hilfe des Tensorproduktes in eine Matrixform gebracht werden. Damit ist der Ausdruck

$$n^T I n \tag{3.50}$$

mit

$$I = \int w(x-x')(\nabla f(x') \otimes \nabla f(x'))d^2x' \tag{3.51}$$

für den Richtungsvektor n zu maximieren. Für die einzelnen Komponenten von I gilt dann:

$$I_{pq}(x) = \int_{-\infty}^{\infty} w(x-x') \left(\frac{\partial f(x')}{\partial x'_p} \frac{\partial f(x')}{\partial x'_q} \right) d^2x'. \tag{3.52}$$

Setzt sich der Einheitsvektor n aus dem Tupel $(\cos\theta, \sin\theta)$ für einen Orientierungswinkel θ zusammen, so ergibt sich

$$I = (n'_x, n'_y) \begin{pmatrix} I_{x'x'} & 0 \\ 0 & I_{y'y'} \end{pmatrix} \begin{pmatrix} n'_x \\ n'_y \end{pmatrix} = I_{x'x'} \cos^2\theta + I_{y'y'} \sin^2\theta. \tag{3.53}$$

Gilt o.B.d.A. $I_{x'x'} > I_{y'y'}$ so maximiert $n' = (1,0)$ den Ausdruck (3.53). Möchte man die Orientierung bestimmen, so ist das Eigenwertproblem

$$Ik_w = \lambda_w k_w \tag{3.54}$$

zu lösen. Diese Gleichung besagt, dass ein Eigenvektor k_w ohne Drehung proportional zum Produkt Ik_w ist. Der Orientierungswinkel wird dann durch

$$\theta = \arg(k_w) \tag{3.55}$$

gegeben.

In der Praxis kann der Strukturtensor durch eine Kombination von linearer Faltung und nichtlinearen Punktoperationen berechnet werden. Zunächst werden die Richtungsableitungen entlang der Koordinatenachsen berechnet. Die Ableitungen werden dabei mit dem optimierten Sobeloperator erzeugt. Für die Komponente I_{pq} des Strukturtensors berechnet man:

$$I_{pq} = B(S_p \cdot S_q), \tag{3.56}$$

wobei B ein zweidimensionaler Binomialfilter, $S_{p,q}$ die Filterantworten der Sobeloperatoren in Bezug auf das Bild sind. Die nichtlineare Punktoperation ist die punktweise Multiplikation zwischen diesen beiden Filterantworten. Aus dem so entstandenen Strukturtensor erhält man den Orientierungsvektor durch die Gleichung

$$o = \begin{pmatrix} I_{xx} - I_{yy} \\ 2I_{xy} \end{pmatrix} \tag{3.57}$$

Der Orientierungswinkel ergibt sich aus dem Argument dieses Vektors. Der Betrag liefert ein Bestimmtheitsmaß für diesen Orientierungswinkel.

3.7 Monogenes Signal

Im Folgenden beschreiben die Vektoren u und x Quaternionen. Betrachtet man ein eindimensionales Signal f und dessen Hilberttransformierte f_H, so wird das analytische Signal definiert.

Definition 3.1 (analytisches Signal) *Sei f ein Signal und $f_H \overset{\text{def}}{=} \mathcal{F}^{-1}\{isign(u)\}$ die Hilberttransformierte von f, so heißt*

$$f_A = f - f_H i$$

das analytische Signal f.

Die wesentliche Eigenschaft des analytischen Signals ist die Aufteilung des Signals f in die entscheidenden Bestandteile – Phase und Amplitude (*split of identity*). Wie bereits in [14] festgestellt wurde, repräsentiert die Phase ein qualitatives Maß für Beschaffenheit von lokaler Struktur und die Amplitude ein qualitatives Maß für das Vorhandensein von Struktur. Um zweidimensionale Signale in dieser Art aufzuteilen, wird ein entsprechendes Signal benötigt. Wie man in Definition (3.1) sieht, wird zur Generierung des analytischen Signals die Hilberttransformierte des Signals herangezogen.

Felsberg stellt in [8] die folgende Transformation vor.

Definition 3.2 (sphärische Hilberttransformation) *Sei $F(u)$ die Fouriertransformierte eines zweidimensionalen Signals f, so heißt*

$$F_H(u) = \tilde{H}(u)F(u) = \frac{u}{|u|}F(u)$$

die sphärische Hilberttransformation des Signals f.

Mit dieser Definition ist es möglich ein analytisches Signal für den zweidimensionalen Fall zu definieren.

Definition 3.3 (sphärisches analytisches Signal) *Sei f ein zweidimensionales Signal und f_H die sphärische Hilberttransformierte von f, dann heißt*

$$f_A = f - f_H i$$

das sphärische analytische Signal von f.

Um die Energie eines zweidimensionalen Signals berechnen zu können, wird eine Energiefunktion benötigt. Sie lautet (hergeleitet aus $F_A \overset{\text{def}}{=} \mathcal{F}\{f_A\}$ in [8]):

$$E(u) = 2\int_{\mathbb{R}^2} |F(u)|^2 du \tag{3.58}$$

Möchte man beliebige Dimensionen betrachten, so ist eine multidimensionale Generalisierung des analytischen Signals nötig. Wie bereits im zweidimensionalen Fall wird die Transformation geeignet verändert. Man definiert die Riesztransformation.

Definition 3.4 (Riesztransformation) *Sei f ein Signal, so heißt*

$$f_H(x) = \int \frac{t}{2\pi|t|^3} i f(x-t)dt \tag{3.59}$$

die Riesztransformation von f.

Wie man sieht, handelt es sich bei der Transformation um eine Faltung. Das sphärische analytische Signal erhält man ebenfalls durch eine Faltung.

$$f_a(x) = \int (\delta_0(t) + \frac{t}{2\pi|t|^3}) f(x-t)dt. \tag{3.60}$$

Felsberg führt in [7] das Monogene Signal mit Hilfe der Riesztransformation in der folgenden Form ein:

$$f_M = \begin{pmatrix} f \\ h_1 * f \\ h_2 * f \end{pmatrix} \tag{3.61}$$

wobei h_1 und h_2 Faltungskerne der Riesztransformation für den zweidimensionalen Fall im Ortsraum darstellen. Sie sind gegeben durch

$$h_1(x,y) = -\frac{x}{2\pi(x^2+y^2)^{\frac{3}{2}}} \tag{3.62}$$

und

$$h_2(x,y) = -\frac{y}{2\pi(x^2+y^2)^{\frac{3}{2}}}. \tag{3.63}$$

Die lokale Amplitude wird aus dem Monogenen Signal durch

$$A_f = \sqrt{f^2 + (h_1 * f)^2 + (h_2 * f)^2} \tag{3.64}$$

berechnet. Weiterhin kann man folgende Gleichungen in sphärischen Koordinaten aufstellen, um die zwei vorhandenen Phasenwinkel zu beschreiben.

$$\begin{aligned} f &= A_f \cos(\phi) && (3.65) \\ (h_1 * f) &= A_f \sin(\phi)\cos(\theta) && (3.66) \\ (h_2 * f) &= A_f \sin(\theta)\cos(\theta) && (3.67) \end{aligned}$$

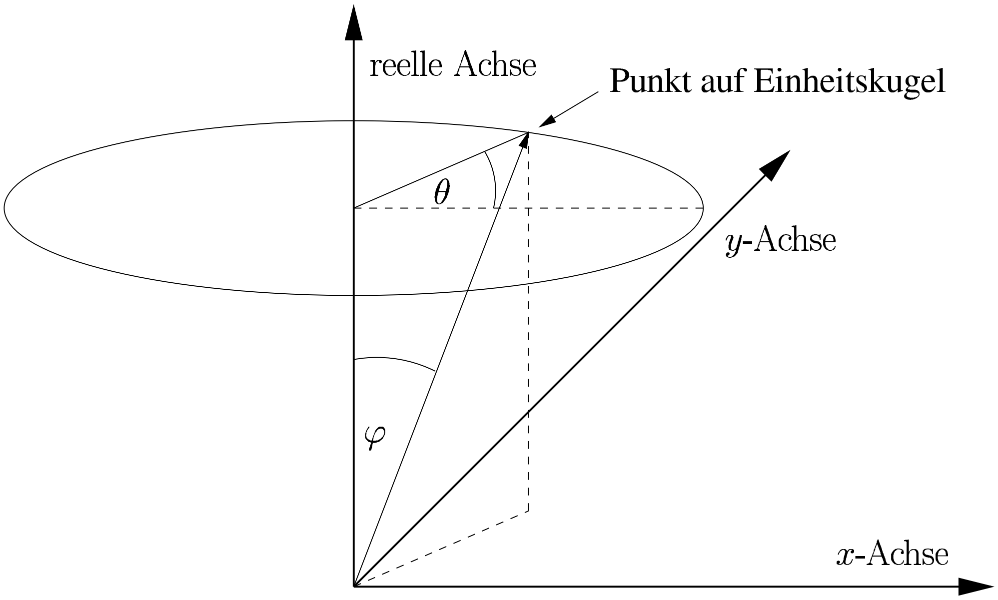


Abbildung 3.12: Anschauliche Darstellung des Monogenen Signals in sphärischen Koordinaten.

mit $\phi \in [0, 2\pi)$ und $\theta \in [0, \pi)$. Abbildung 3.12 gibt eine Vorstellung von der Einbettung des Monogenen Signals in sphärischen Koordinaten. Der Winkel ϕ wird als lokaler Phasenwinkel angesehen, θ als lokaler Orientierungswinkel.

Analog zu den Quadraturfiltern, die mit der Hilfe der Hilberttransformation berechnet werden, kann man für die Riesztransformation sogenannte sphärische Quadraturfilter angeben, die die Energie für das Monogene Signal berechnen. In [7] wird dazu ein logarithmischer normierter Bandpassfilter vorgeschlagen, der im Frequenzraum die Übertragungsfunktion

$$B(u_1, \underset{1}{u_2}) = \exp\left(-\frac{(\log^2(\sqrt{u_1^2 + u_2^2}/2^k))^2}{2\log^2(w)}\right) \tag{3.68}$$

besitzt, wobei k die mittlere Frequenz und w die Bandbreite angibt. Mit den Filtern B, H_1B und H_2B wird die Berechnung des Monogenen Signals im Frequenzraum durchgeführt (H_1 und H_2 sind die Fouriertransformierten Darstellungen von h_1 bzw. h_2). Die Faltung im Ortsraum wird durch eine Multiplikation im Frequenzraum ersetzt (Faltungstheorem).

Um lokale Orientierungen berechnen zu können, wird der Orientierungwinkel aus dem Monogenen Signal berechnet. Es ergibt sich für den Winkel

$$\theta = \mathrm{atan1}\left(\frac{h_2 * f}{h_1 * f}\right), \tag{3.69}$$

wobei dieser so verändert wird, dass er die Orientierung im Intervall $[0, \pi)$ wiedergibt.

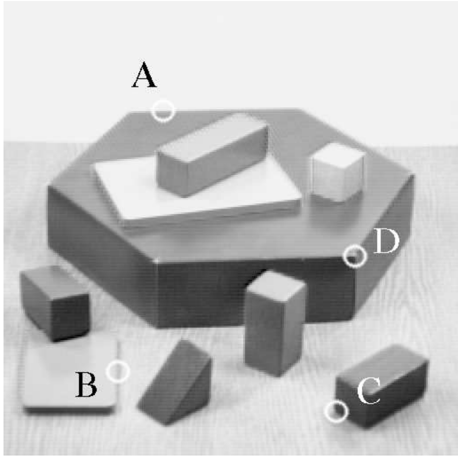
Berechnet man das Monogene Signal für verschiedene Frequenzbänder, so können die resultierenden lokalen Phasenwinkel dazu benutzt werden, das Schema der Phasenkongruenz für Kantendetektion zu nutzen (siehe Abschnitt 3.5).

Kapitel 4

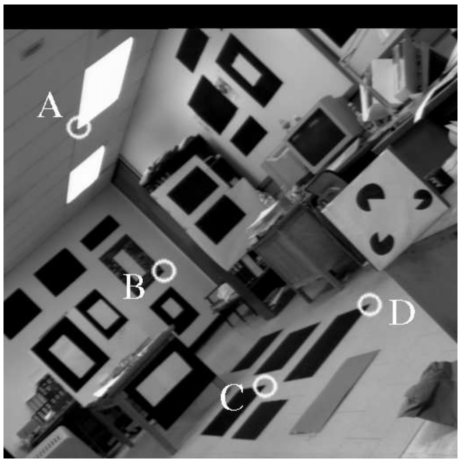
Anwendungen der Strukturauswertungsverfahren

4.1 Beschreibung der Bildmenge

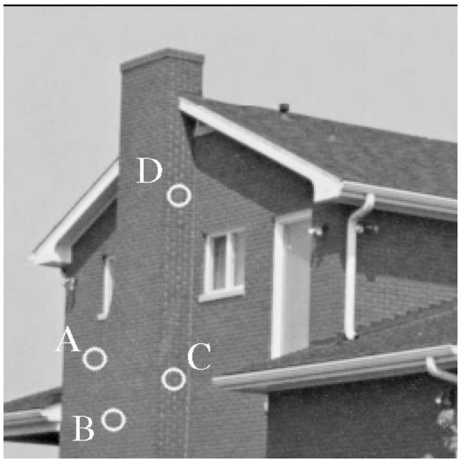
Die zur Analyse der Verfahren hinsichtlich ihrer Eigenschaften verwendeten Bilder sollten möglichst viele in der Realtität vorkommende Bedingungen wiedergeben. Um die Realität abzubilden werden physikalische Meßgeräte, etwa ein Kamera–System oder ein Bild–Scanner verwendet. Diese Geräte werden durch verschiedene Faktoren in ihrer Abbildungsfähigkeit beeinflußt. Solche Faktoren sind z.B. technisch bedingtes Rauschen des Meßgerätes oder die Beleuchtungsverhältnisse in der realen Welt. Die hier verwendete Bildermenge deckt wesentliche Faktoren ab. Sie ist sowohl durch Kamera–Systeme als auch durch Scan-Systeme entstanden. Alle Bilder haben eine Größenordnung von 512^2 Pixel. Es handelt sich um acht Bilder die kurz vorgestellt werden:



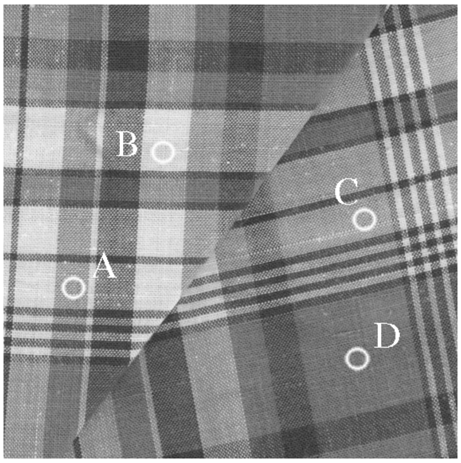
blox



lab



house



hemdgrau

Die Bilder *blox*, *lab* und *house* sind dem Internet entnommen worden (http://www.ee.surrey.ac.uk/Research/VSSP/demos/corners/originals.html).

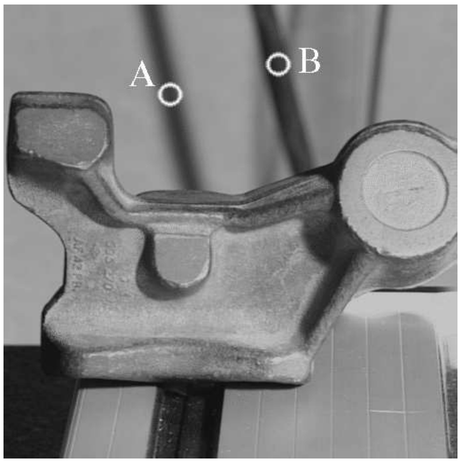
Bild *blox* enthält Bauklötze, die auf einem Tisch mit einer Holzmaserung liegen. Die Bauklötze haben homogene Oberflächen und klare Konturen. Das Bild enthält ein geringes Grundrauschen, sowie regelmäßige Zeilenstörungen. Es handelt sich um eine Kameraaufnahme. Die markierten Punkte sind Verbindungspunkte. A ist eine horizontale Kante; B ist eine vertikale Kante; C ist eine L-Verbindung und D eine Y-Verbindung.

Bild *lab* ist eine Aufnahme eines Labors. Es kommen viele Ecken und Kanten vor, die zum Teil dicht beeinanderliegen. Das Bild enthält kontrastarme Regionen und hat, das für Kameraaufnahmen typische, Grundrauschen. Die markierten Punkte sind alle L-Verbindungen.

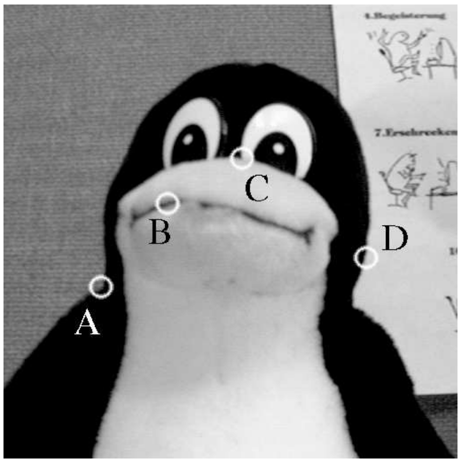
Bild *house* ist eine Kameraaufnahme einer Hausansicht. Das Haus besitzt klare Konturen sowie eine leicht texturierte Mauerfläche. Die markierten Punkte A und B liegen innerhalb einer einheitlichen texturierten Fläche. Dieses gilt auch für die Punkte C und D.

Bild *hemdgrau* enhält ein gescanntes Hemd, welches Texturen verschiedener Orientierung beinhaltet. Die vorkommenden Texturen können in zusammenhängende Flächen segmentiert werden. Alle markierten Punkte liegen in verschiedenen texturierten Flächen, die Orientierungen dieser Texturen variieren.

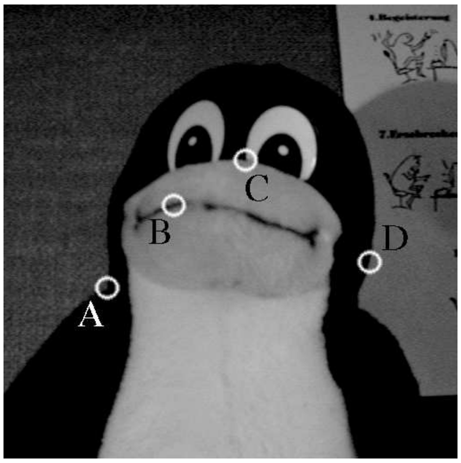
fraumitbusch



reno INRIA-Syntim©



tuxhell



tuxdunkel

Das Bild *reno* ist ebenfalls dem Internet entnommen worden (http://www-syntim.inria.fr/-syntim/analyse/images/Reno0OG0.gif).

Bild *fraumitbusch* ist ein gescanntes Bild. Die Scanvorlage ist eine gerasterte Prospektseite, was dazu führt, dass ein leichtes Moiré durch die Unterabtastung entstanden ist. Das Buschwerk enthält feine ungeordnete Strukturen.

Bild *reno* zeigt ein metallisches Objekt, welches im Vordergrund der Kameraaufnahme steht. Diese Objekt besteht aus Strukturen verschiedener Skalen. Im Hintergrund des Bildes sind unscharfe Strukturen zu erkennen. Die Punkte A und B markieren Verbindungspunkte, die eine große Strukturbreite haben. A ist dabei eine vertikale Kante und B eine X-Verbindung.

Bild *tuxhell* wurde durch eine kostengünstige, handelsübliche Web-Kamera aufgenommen. Das Objekt wurde gut ausgeleuchtet. Es ist ein großes Grundrauschen im Bild vorhanden. Die markierten Punkte stellen wieder verschiedene Verbindungpunkte dar: A ist eine L-Verbindung, B eine horizontale Kante, C eine T-Verbindung und D ist eine vertikale Kante.

Bild *tuxdunkel* ist im Gegensatz zu Bild *tuxhell* schlecht ausgeleuchtet, was bei der CCD-Kamera zu einem stärkeren Grundrauschen führt. Die markierten Punkte entsprechen denen aus dem Bild *tuxhell*.

Bei der Anwendung der Verfahren auf die in den Bildern enhaltenen lokalen Strukturen, werden im Folgenden nicht alle möglichen Kombinationen präsentiert, da dies den Umfang der Arbeit sprengen würde. Es werden für das Verhalten der Verfahren repräsentive Ergebnisse vorgestellt.

4.2 Analyse der Verfahren

In den folgenden Unterabschnitten werden die Verfahren analysiert. Als Grundlage für die Analyse werden die in Abschnitt 4.1 beschriebenen Bilder verwendet. Ein markierter Punkt eines Bildes wird im Folgenden immer in der Form: *Bild(Punkt)* geschrieben (z.B. *tuxhell*(A)). Zunächst werden alle Verfahren hinsichtlich ihrer qualitativen Aussage beurteilt. Insbesondere sollte nur das getestet werden, wozu die Verfahren auch erdacht worden sind.

4.2.1 Qualität

Unter der Qualität der Verfahren wird hier die Exaktheit bzw. Genauigkeit der Zuordnung der Merkmale zu den lokalen Strukturen verstanden. Die Merkmale die hier betrachtet werden sind

1. Kanten (zusammenhängende Ränder von Bildobjekten)
2. Verbindungspunkte (z.B. Y-, X- und T-Verbindungen)

3. Orientierungen und Richtungen
4. Flächen und Texturen (gleichmäßige Muster).

Bei Kanten kommt es darauf an, dass ihre Lage genau festgestellt wird. Mitunter kann eine verfahrensbedingte Verschiebung der Kante auftreten, welche natürlich zu einer Verfälschung des Ergebnisses führt. Bei den Verbindungspunkten sind zwei Dinge von Bedeutung. Zum einen ist es wichtig, dass der Verbindungspuntk exakt klassifiziert wird, d.h. die Zuordnung zu den Klassen Y-, X- und T-Verbindungen (dies sind noch nicht alle Klassen) sollte korrekt sein. Zum anderen ist es führ weitere Analysen wichtig, die genaue Beschaffenheit des Verbindungspunktes zu haben, d.h. die Genauigkeit der Winkel der Geraden vom Verbindungspunkt aus sind von starkem Interesse. Bei Orientierungen und Richtungen kommt es ebenfalls auf die exakten Winkel an. Liefern die Verfahren bei gleichmäßigen Flächen (auch Texturen) in lokalen Strukturen ähnliche Ergebnisse, so ist eine Segmentierung der Flächen denkbar.

In der nun folgenden Analyse werden die Verfahren hinsichtlich ihrer Verwendung beurteilt. Ein Problem, was sich dabei ergibt, ist das kein *a priori* Wissen über die Bilder existiert. Im Gegensatz zu anderen Arbeiten wird hier auf die Analyse von künstlich erzeugten Bildern verzichtet. Um etwas über die Qualität der Verfahren sagen zu können, werden Messungen, die von Hand durchgeführt worden sind, herangezogen.

Verbindungspunkte, Orientierungen und Richtungen

Zunächst wird nur das Bild $blox$ und die in ihm enthaltenen markierten Punkte $blox$(A-D) betrachtet. Diese Punkte werden in kleinere Bilder, die durch Beschneidung aus den kompletten Bildern entstehen, eingebettet. Diese kleineren Bilder beinhalten den markierten Punkt genau in der Mitte, sie haben eine Größenordnung von 31^2 Pixel. Für die Verbindungspunkte $blox$(A-D) sind folgende Richtungen gemessen worden:

Verbindungspunkt	Richtung 1	Richtung 2	Richtung 3
$blox$(A)	185°	5°	
$blox$(B)	85°	270°	
$blox$(C)	90°	335°	
$blox$(D)	65°	190°	270°

. (4.1)

Polarpyramiden Das Verfahren der Polarpyramide benötigt bei der Anwendung zusätzliche Parameter (zusätzlich zum zu untersuchenden Bildpunkt). Bei der Polartransformation wird der Radius der Umgebung benötigt und beim veränderten Rechteckfilter der kleinste Radius. Die

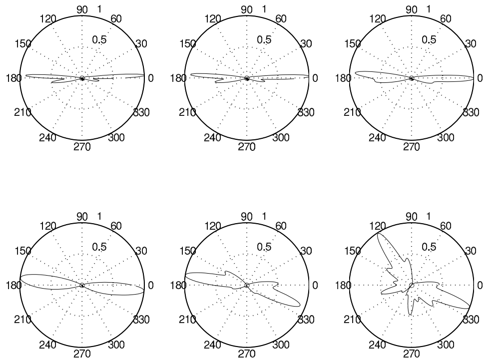


Abbildung 4.1: Horizontale Kante *blox*(A) analysiert mit der Methode der Polarpyramiden.

verwendeten Werte sind $r_{min} = 5$ und $r_{max} = 15$. Die Filterantworten sind den Abbildungen 4.1, 4.2, 4.3 und 4.4 zu entnehmen. In der ersten Reihe der Abbdildungen steht links die Filterantwort des modifizierten Rechtfilters. Die weiteren Plots entsprechen den Filterantworten der jeweiligen Pyramidenstufe. Dargestellt sind insgesamt fünf Stufen.

In Abbildung 4.1 wird eine horizontale Kante detektiert. Wie man erkennt, werden genau zwei Richtungen erkannt, die bei 0° und bei 180° liegen. Mit den Pyramidenstufen nimmt die Genaugigkeit der Detektion ab, es wird zwar immer noch eine Kante detektiert, deren Orientierung liegt aber im Vergleich zur ersten Stufe schon um 60° gedreht (von der letzten Stufe aus betrachtet). In der dritten Stufe der Polarpyramide treten nur genau zwei exakte Maxima auf. Dies ist ein Hinweis darauf, dass es sich bei *blox*(A) um eine horizontale Kante mit einer gewissen Strukturbreite handelt.

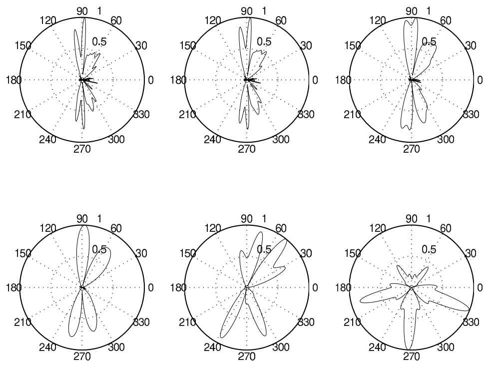


Abbildung 4.2: Vertikale Kante *blox*(B) analysiert mit der Methode der Polarpyramiden.

Eine vertikale Kante soll in Abbildung 4.2 erkannt werden. Die vertikale Kante $blox$(B) liegt in einer Region geringen Kontrastes, was zu Detektionsproblemen führt. Wie man bereits in der zweiten Stufe der Polarpyramide sieht, treten verschiedene zusätzliche Richtungen auf. Beschränkt man sich auf die zwei wesentlichen Maxima bei 90° und 270°, so findet man genau die vertikale Kante. Ab der vierten Stufe der Pyramide werden die detektierten Richtungen ungenau, in Stufe fünf der Pyramide sind sogar Maxima bei 190° und 340° vorhanden, was schon fast einer horizontalen Kante entspricht.

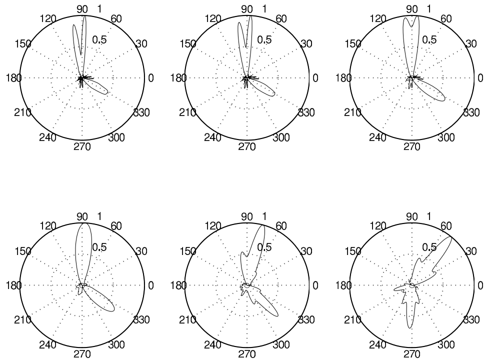


Abbildung 4.3: L-Verbindungspunkt $blox$(C) analysiert mit der Methode der Polarpyramiden.

Der L-Verbindungspunkt $blox$(C) wird in Abbildung 4.3 analysiert. Er liegt in einer Region größten Kontrastes, was eine im Verhältnis zu 4.2 bessere Detektion der Richtungen erwarten lässt. In der Tat findet man zwei wesentliche Ausprägungen in den Plots, die bei 90° und 330° liegen, die genau den Richtungen des Verbindungspunktes entsprechen. Ab der vierten Stufe nimmt die Genauigkeit der Detektion ab, was sich in der Verschiebung der Maxima widerspiegelt.

Abbildung 4.4 stellt die Ergebnisse für die Anwendung der Polarpyramide auf dem Y-Verbindungspunkt $blox$(D) dar. Hier kann man sehen, dass eine Detektion der Richtungen, bei fast optimalen Kontrastverhältnissen, zu exakten Ergebnissen führt. Die drei Orientierungen sind sehr gut erkennbar, es werden genau drei Richtungen 60°, 180° und 270° detektiert. Die Richtungen der fünften Pyramidenstufe sind nicht mehr zu gebrauchen.

Insgesamt ist das Verfahren der Polarpyramiden für die Klassifikation von Verbindungspunkten geeignet. Desweiteren kann es auch für die Richtungsanalyse eingesetzt werden.

Steuerbare Filter Bei den steuerbaren Filtern werden die Basisfilter gemäß Abschnitt 3.3 erzeugt. Die hier verwendeten Filter haben eine Größenordnung von 13^2 Pixel. Zur Anwendung werden keine weiteren Parameter bis auf den zu betrachtenden Punkt benötigt.

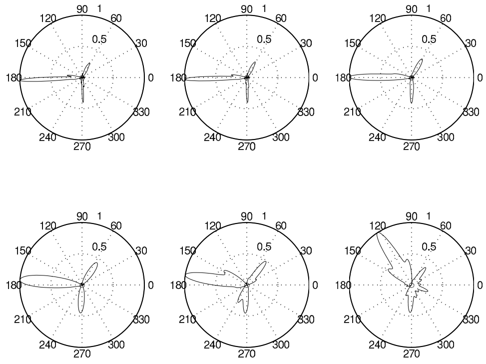


Abbildung 4.4: Y-Verbindungspunkt *blox*(D) analysiert mit der Methode der Polarpyramiden.

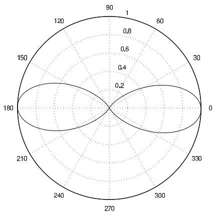


Abbildung 4.5: Horizontale Kante *blox*(A) analysiert mit der Methode der Steuerbaren Filter.

Die horizontale Kante wird durch die Steuerbaren Filter detektiert. Abbildung 4.5 zeigt deutlich die zwei ausgeprägten Maxima, die exakt um 180° auseinanderliegen. Dies gibt den ersten Hinweis darauf, dass das Verfahren nur Orientierungen detektieren kann.

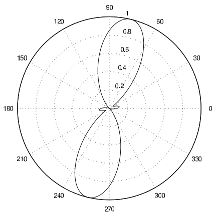


Abbildung 4.6: Vertikale Kante *blox*(B) analysiert mit der Methode der Steuerbaren Filter.

Die Methode der Steuerbaren Filter liefert für die vertikale Kante eine Detektion der Orientierung. In Abbildung 4.6 kann man eine leichte Drehung der Kante erkennen. Auch hier liegen die beiden Maxima exakt um 180° auseinander.

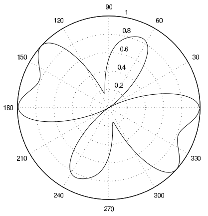


Abbildung 4.7: L-Verbindungspunkt *blox*(C) analysiert mit der Methode der Steuerbaren Filter.

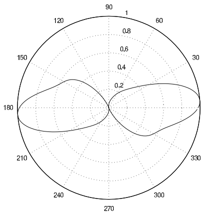


Abbildung 4.8: Y-Verbindungspunkt *blox*(D) analysiert mit der Methode der Steuerbaren Filter.

Abbildung 4.7 und 4.8 zeigen deutlich, dass die betrachteten Verbindungpunkte *blox*(C) und *blox*(D) nicht korrekt detektiert werden. Man stellt fest, dass der Energieplot der Steuerbaren Filter im Ursprung punktsymmetrisch ist. Diese Symmetrie ergibt sich daraus, dass die in den Steuerbaren Filtern verwendeten Gaussableitungen vierter Stufe lediglich eine Orientierung beschreiben können. Für die Ableitung ist es bei einer Orientierung egal, aus welcher Richtung sie bestimmt wird.

Wedgefilter Die bei der Analyse der Wedgefilter benutzte Filterbasis hat eine Größe von 20 geraden und ungeraden Wedgefiltern. Eine Wedgefilterbasis ist z.B. in Abbildung 3.10 dargestellt, die Basis für 20 Filterpaare ist hier aus Gründen der Übersicht nicht aufgeführt. Die für die Basis erzeugten Filter haben eine Größenordnung von 31^2 Pixel. Bei den Wedgefiltern müssen ebenfalls zusätzliche Parameter angegeben werden. Diese Parameter betreffen die radial-Funktion (3.38). Zum einen sind es zwei Radien (ähnlich zu denen der Polarpyramide) und zum anderen sind es noch zwei Faktoren, die die Glattheit der Übergänge in radialer Richtung bestimmen. Die hier verwendeten Parameter sind $R_1 = 5$, $R_2 = 15$, $\delta_1 = \delta_2 = 1$. Im Folgenden werden wieder die vier bekannten Punkte aus dem Bild *blox* untersucht.

Abbildung 4.9 stellt das Ergebnis der Anwendung der Filterbasis dar. Man erkennt deutlich zwei Maxima, die in Richtung 187° und 355° liegen. Der Verbindungspunkt wird also korrekt als

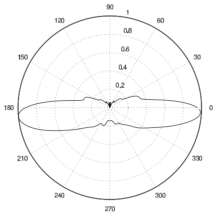


Abbildung 4.9: Horizontale Kante *blox*(A) analysiert mit der Methode der Wedgefilter.

horizontale Kante detektiert. Zu beachten ist hier, dass eine leichte Drehung der Kante gegen den Urzeigersinn detekiert wird, welche sich auch in dem Verbindungspunkt *blox*(A) wiederspiegelt.

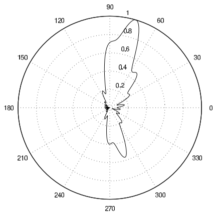


Abbildung 4.10: Vertikale Kante *blox*(B) analysiert mit der Methode der Wedgefilter.

Die vertikale Kante *blox*(B) wird von den Wedgefiltern, trotz des geringen Kontrastes, erkannt. Es gibt zwei Maxima, die genau in Richtung der Kante liegen. Das Maximum bei 285° ist ein wenig geringer ausgeprägt, was zeigt, dass die Energie in dieser Richtung verhältnismäßig gering ist. Dies gibt einen Hinweis darauf, dass der Kontrast in dieser Richtung leicht abnimmt.

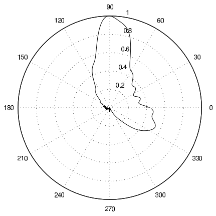


Abbildung 4.11: L-Verbindungspunkt *blox*(C) analysiert mit der Methode der Wedgefilter.

Abbildung 4.11 zeigt, dass auch der L-Verbindungspunkt *blox*(C) identifiziert wird. Es ergeben sich die Richtungen 90° und 330°. Ähnlich wie bei der vertikalen Kante zuvor, gibt es ein weniger ausgeprägtes Maximum. Betrachtet man die Umgebung um den Punkt *blox*(C), so stellt man fest, dass der zugehörige Quader einen leichten Schatten nach unten wirft. Dieser führt zu einem geringeren Kontrast im Verhältnis zu dem an der linken Seite des Quaders.

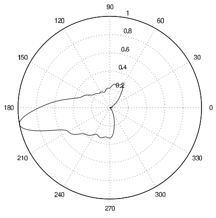


Abbildung 4.12: Y-Verbindungspunkt $blox$(D) analysiert mit der Methode der Wedgefilter.

Die Y-Verbindung $blox$(D) wird zwar durch die Wedgefilter in 4.12 wiedergegeben, jedoch sind die drei Maxima sehr verschieden. Die Richtung 192° ist besonders dominant, was wiederum auf den dort größeren Kontrast im Verhältnis zu den anderen beiden Richtungen zurückzuführen ist.

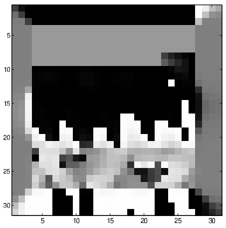
Insgesamt stellt man fest, dass die Wedgefilter mit den hier benutzten 20 Basisfilterpaaren, eine genügende Winkelauflösung besitzen, um die vier Verbindungspunkte genau wiederzugeben. Desweiteren spielt der Kontrast eine Rolle bei der Ausprägung von Maxima.

Strukturtensor Bei der Anwendung des Strukturtensors wurde für die Richtungsableitungen der optimierte Sobeloperator, wie ihn Jähne in [14] vorgestellt hat, eingesetzt. Zur Glättung der punktweise multiplizierten Richtungsableitungen für den Tensor, wird ein zweidimensionaler Binomialfilter eingesetzt. Er hat eine Filtermaskenbreite von 13 Pixel. Die Beträge der zugehörigen Orientierungsvektoren (siehe 3.57) geben ein Bestimmtheitsmaß für die Orientierung wieder.

Am Orientierungsbild der Abbildung 4.13 kann man deutlich eine horizontale Struktur erkennen. Schwarze Punkte (Grauwerte nahe Null) und ganz helle Punkte (Grauwerte nahe bei 255) geben eine Orientierung für 0° bzw. 180° wieder. Der aus dem Strukturtensor berechnete Winkel bestätigt dies.

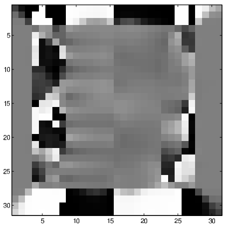
Die vertikale Kante $blox$(B) wird in ihrer Orientierung ebenfalls detektiert. Die leichte Abweichung des Winkels von dem in Tabelle 4.1 ist wiederum mit dem geringen Kontrast in der Umgebung des Verbindungspunktes zu begründen (siehe Abbildung 4.14).

Die Abbildungen 4.15 und 4.16 zeigen, dass bei nicht-Einhaltung einer einfachen Nachbarschaft (intrinsische Dimension ist ungleich eins), der Orientierungsvektor entlang des maximalen Gradienten zeigt. Der L-Verbindungspunkt $blox$(C) hat für die erste Richtung den maximalen Gradienten, da für die zweite Richtung der Kontrast durch den leichten Schatten des Bauklotzes



$$\begin{pmatrix} 0.2322 & -6.2556 \\ -6.2556 & 304..4420 \end{pmatrix} \equiv 1.1^\circ$$

Abbildung 4.13: Orientierungsbild der horizontalen Kante $blox$(A) analysiert mit dem Strukturtensor und Angabe des Strukturtensors sowie des Orientierungswinkels. Das Bestimmtheitsmaß liefert einen großen Wert.

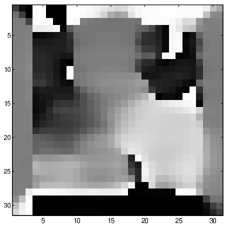


$$\begin{pmatrix} 12.7909 & 1.7436 \\ 1.7436 & 0.3160 \end{pmatrix} \equiv 97.8^\circ$$

Abbildung 4.14: Orientierungsbild der vertikalen Kante $blox$(B) analysiert mit dem Strukturtensor und Angabe des Strukturtensors sowie des Orientierungswinkels. Hier ist das Bestimmtheitsmaß relativ klein.

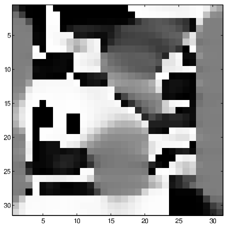
herabgesetzt ist. Bei $blox$(D) liegt der maximale Gradient in der zweiten Richtung. Die aus den jeweiligen Strukturtensoren berechneten Winkel bestätigen diesen Sachverhalt. Das Verfahren des Strukturtensor erlaubt, bein einer vorhanden einfachen Nachbarschaft, Orientierungen zu erkennen.

Monogenes Signal Das Monogene Signal wird mittels radialer Bandpassfilter erzeugt. Hier wurde die vierte Oktave mit einer Bandbreite von 0.55 verwendet. Aus dem Monogenen Signal wurde dann die lokale Orientierung gemäß Gleichung (3.69) berechnet. Für die ersten beiden



$$\begin{pmatrix} 476.7664 & 37.7778 \\ 37.7778 & 17.9393 \end{pmatrix} \equiv 94.6755^\circ$$

Abbildung 4.15: Orientierungsbild des L-Verbindungspunktes $blox(\mathrm{C})$ analysiert mit dem Strukturtensor und Angabe des Strukturtensors sowie des Orientierungswinkels.



$$\begin{pmatrix} 28.9871 & 85.6184 \\ 85.6184 & 773.2222 \end{pmatrix} \equiv 173.5^\circ$$

Abbildung 4.16: Orientierungsbild des Y-Verbindungspunktes $blox(\mathrm{D})$ analysiert mit dem Strukturtensor und Angabe des Strukturtensors sowie des Orientierungswinkels.

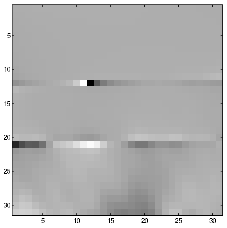


Abbildung 4.17: Orientierungsbild der horizontalen Kante $blox(\mathrm{A})$ analysiert mit dem Monogenen Signal. Der gemessene Orientierungswinkel beträgt 178.7°.

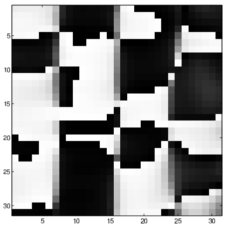


Abbildung 4.18: Orientierungsbild der vertikalen Kante *blox*(B) analysiert mit dem Monogenen Signal. Der gemessene Orientierungswinkel beträgt 97.7°.

Verbindungspunkte *blox*(A) und *blox*(B), die auf Kanten, also eine intrinsisch eindimensionale Struktur besitzten, liegen, liefert das Monogene Signal Winkel, die den vorher bestimmten nahe kommen (die maximale Differenz beträgt 12.7°). Diese Differenz ist zurückzuführen auf die gewählte Oktave, sie bestimmt, welche Strukturen gemäß ihrer Frequenzanteile ausgewählt werden. Die Orientierungsbilder für Umgebungen der Punkte sind in Abbildung 4.17 bzw. 4.18 angegeben.

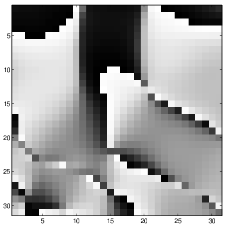


Abbildung 4.19: Orientierungsbild des L-Verbindungspunktes *blox*(C) analysiert mit dem Monogenen Signal. Der gemessene Orientierungswinkel beträgt 77.3°.

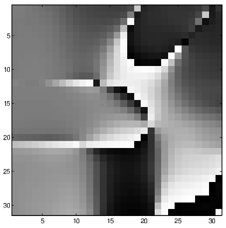


Abbildung 4.20: Orientierungsbild des Y-Verbindungspunktes *blox*(D) analysiert mit dem Monogenen Signal. Der gemessene Orientierungswinkel beträgt 7.0°.

Die Verbindungspunkte $blox$(C) und $blox$(D) können durch keine lokale Orientierung dargestellt werden. Das Monogene Signal liefert deshalb für beide keine aussagekräftigen Winkel.

Kanten

Für die Kantendetektion können alle hier betrachteten Verfahren eingesetzt werden. Die Verfahren: Polarpyramide, Steuerbare Filter und Wedgefilter erlauben es, einen Verbindungspunkt, der auf einer Kante liegt, als einen Kantenpunkt zu identifizieren. Dieses gilt auch für den Strukturtensor und das Monogene Signal (eingeschränkt auf den lokalen Orientierungswinkel). Hier müssen jedoch noch zusätzlich Nachbarschaften betrachtet werden, um eine Kante (zusammenhängender Rand) zu identifizieren. Das Orientierungsbild des Strukturtensors in Abbildung 4.21 für das Bild *tuxhell* zeigt deutlich, dass man Kanten als zusammenhängende Flächen identifizieren kann. Die vom Strukturtensor auf den Kanten erzeugten Flächen bereiten bei der Lokalisation Probleme, da sie eine Ausdehnung besitzen. Man kann also eine Kante nur innerhalb einer Umgebung identifzieren.

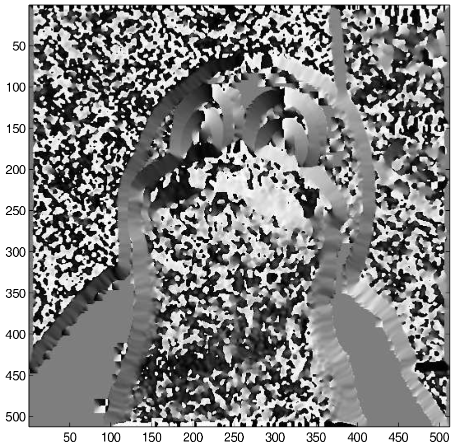
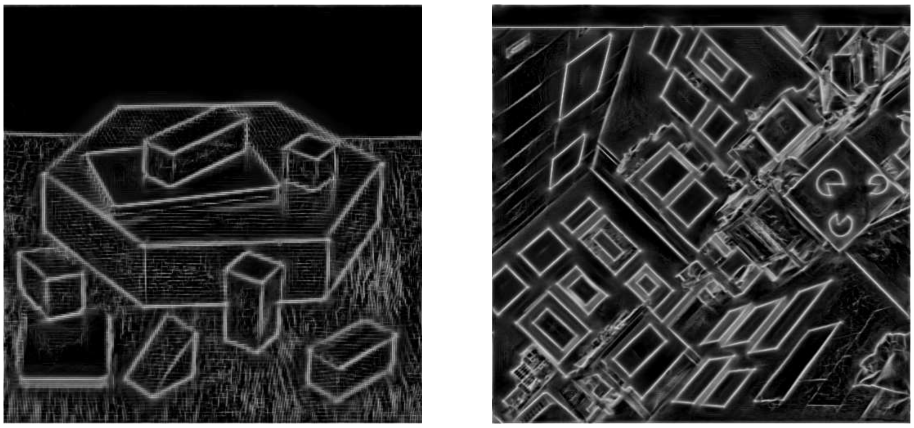


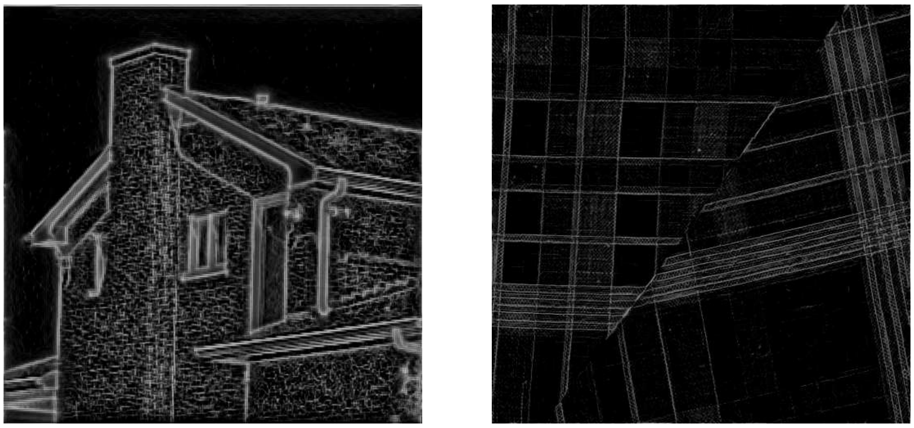
Abbildung 4.21: Orientierungsbild erzeugt durch den Strukturtensor für das Bild *tuxhell*

Es müssen noch die Verfahren: Phasenkongruenz und Monogenes Signal (eingeschränkt auf die lokale Phaseninformation) betrachtet werden. Wendet man das Verfahren der Phasenkongruenz auf die verschiedenen Bilder an, so erhält man die folgenden Bilder, in denen das Maß für jeden Bildpunkt kodiert ist (benutzt wird das Maß PC_2, siehe 3.45). Für die Anwendung der Phasenkongruenz werden keine weiteren Parameter benötigt, die Anwendung erfolgt in genau der von Kovesi (siehe 3.5) vorgeschlagen Art und Weise.

Abbildung 4.22 zeigt die ersten vier Bilder, in denen das Maß für die Phasenkongruenz



blox lab



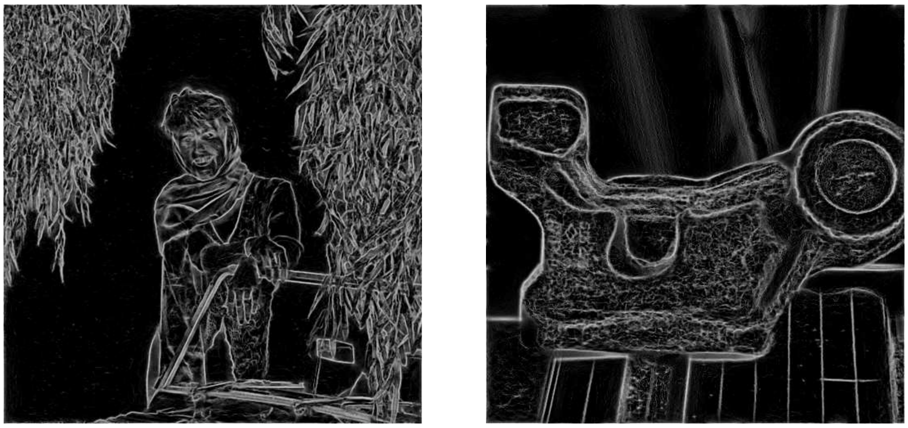
house hemdgrau

Abbildung 4.22: Bilder analysiert mit der Methode der Phasenkongruenz.

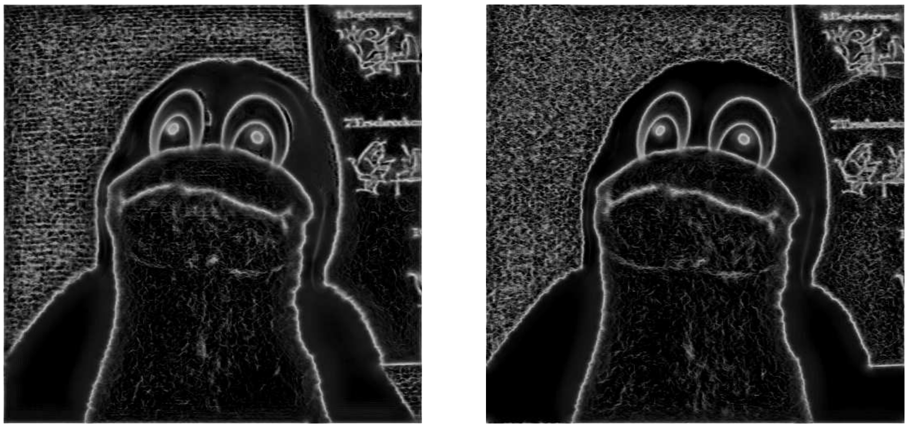
kodiert ist. Bei dieser Art der Darstellung gilt, dass je größer der Grauwert eines Pixel ist, desto größer ist das Maß der Phasenkongruenz an dieser Stelle. Das Bild für *blox* zeigt deutlich die Konturen der Bauklötze. Die bei der Bilderbeschreibung beschriebenen regelmäßigen Zeilenstörungen werden auf den Oberflächen der Klötze sichtbar. Desweiteren werden große Werte für die Phasenkongruenz auf der hölzernen Tischplatte angezeigt, was die Unterscheidung zu Objektkanten problematisch gestaltet.

Das Verfahren der Phasenkongruenz besitzt eine gute lokale Auflösung. Dies wird deutlich im Bild *lab*. Auf dem Boden und auf den Wänden befinden sich Strukturen, die nur einen geringen Abstand zueinander haben. Weiterhin kann man deutlich erkennen, dass hier die Helligkeit und der Kontrast einer Region kaum eine Rolle bei der Detektion von Kanten spielen. Selbst die Kante der auf dem Boden vordersten Platte (unter rechts im Bild kaum zu sehen) sind noch zu erkennen. Dies zeigt wieder, dass die Phase entscheidend für die Bildinformation ist.

Ein Problem der Phasenkongruenz tritt bei Bildern mit sehr vielen Strukturinformationen auf. Dies sieht man z.B. an den Phasenkongruenz Bildern von *house* und *hemdgrau*, dort gibt es viele Texturen, auf die das Maß der Phasenkongruenz reagiert. Es werden viele Kanten innerhalb der texturierten Flächen detektiert. Im Bild *hemdgrau* werden aber dennoch einige Grenzkanten zwischen verschiedenen Texturen gefunden.



fraumitbusch reno



tuxhell tuxdunkel

Abbildung 4.23: Bilder analysiert mit der Methode der Phasenkongruenz.

Für die restlichen Bilder sind die Ergebnisse der Phasenkongruenz in Abbildung 4.23 angegeben. Das Ergebnis für das Bild *fraumitbusch* zeigt, dass das leichte Moiré sich hier nicht auf die Kantendetektion auswirkt. Feine Strukturen, wie sie in den Blättern vorhanden sind, werden einwandfrei wiedergegeben.

Im Bild *reno* sind im Hintergrund des metallischen Objektes Kanten größerer Strukturbreite vorhanden (siehe dazu *reno*(A) und *reno*(B)). Wie man in den entsprechenden Bildern der Phasenkongruenz sieht, werden diese ebenfalls dargestellt. Es ist lediglich das Maß an diesen Stellen geringer im Verhältnis zu Kanten kleinerer Strukturbreite.

Die Bilder *tuxhell* und *tuxdunkel* zeigen erneut, dass die Helligkeit (Beleuchtung) und der Kontrast keine große Rolle bei der Phasenkongruenz spielen. Es wird hier deutlich, dass das Rauschen der CCD-Kamera in *tuxdunkel* zunimmt, was man am Hintergrund des Stoff-Pinguins erkennt. Rauschen zeichnet sich im Allgemeinen durch hohe Frequenzanteile aus, die möglicherweise als Kanten interpretiert werden können.

Die Auswertung der lokalen Phaseninformation, die durch das Monogene Signal geliefert wird, musste an dieser Stelle aus Zeitgründen weggelassen werden. Eine Analyse des Monogenen Signals bezüglich einem der Phasenkongruenz ähnlichen Verfahren wird von Felsberg in [7] betrachtet. Dort wird festgestellt, dass die Berechnung der Phasenkongruenz mittels der lokalen Phase ähnlich gute Ergebnisse liefert. Besonders vermerkt ist dort das gute Laufzeitverhalten.

Texturen

Das Verhalten der Verfahren hinsichtlich der Segmentierung von Texturen konnte leider nicht mehr im Rahmen dieser Arbeit überprüft werden.

4.2.2 Stabilität

In diesem Abschnitt werden die Verfahren hinsichtlich ihrer Stabilität gegenüber äußeren Einflüssen analysiert. Entscheidend ist dabei des Verhalten der Verfahren gegenüber leichten Variationen der Eingabebilder. Führt eine leichte Veränderung des Eingabebildes zu einer leichten Veränderung des Ausgabebildes, so spricht man von einem *bounded input - bounded output*-System (BIBO). Bei den Variationen der Eingabebilder handelt es sich um Rauschen und Helligkeitsvariationen. Das hier betrachtete Rauschen ist ein mittelwertfreies, normalverteiltes Rauschen. Das im Folgenden betrachtete Bild *tuxhell* enthält natürlich, bedingt durch den Abbildungsprozeß, ein eigenes Grundrauschen. Es wird ein neues Bild *tuxrausch* erzeugt, das sich aus *tuxhell* durch zusätzliches Verrauschen ergibt. Das so entstandene Bild hat mit dem zusätzlichen Rauschen einen geschätzten Signal-Rauschabstand von kleiner-gleich $5dB$ (bedingt durch das hinzugefügte Rauschen). Helligkeits- bzw. Kontrastvariationen werden sinnvoller Weise in den Bildern *tuxhell* und *tuxdunkel* betrachtet, da diese bei unterschiedlichen Helligkeitsbedingungen aufgenommen wurden. Zu erwarten ist, dass die Verfahren, die lineare Operatoren enthalten, ein gutes Verhalten gegenüber äußeren Einflüssen zeigen. Dieser Sachverhalt ergibt sich aus der Tatsache, dass das Rauschen additiv ist und es sich bei den Faltungen um lineare Operationen handelt.

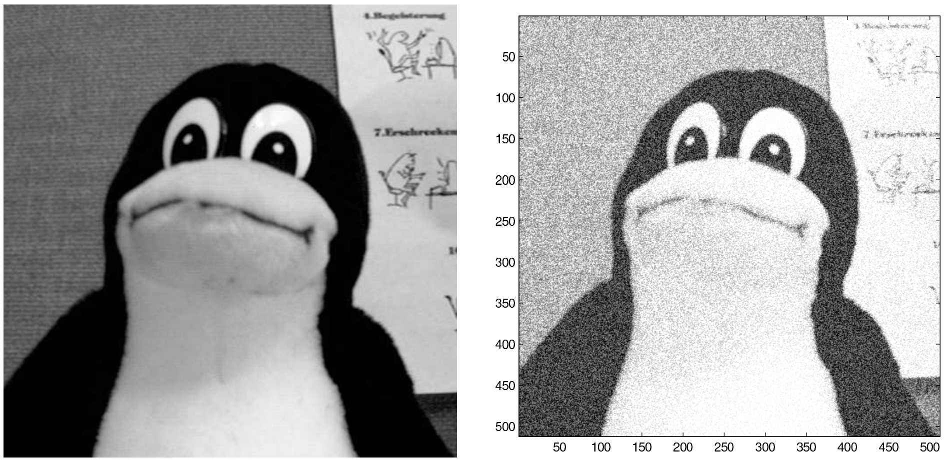


Abbildung 4.24: linkses Bild: *tuxhell*, rechts Bild: *tuxrausch* entstanden durch Hinzufügen eines zusätzlichen mittelwertfreien, gleichverteilten Rauschens (SNR ≤ 5 dB).

Verbindungspunkte, Orientierungen und Richtungen

Für die weitere Analyse sind wieder die Orientierungen der Verbindungspunkte *tuxhell*(A-D) gemessen worden.

Verbindungspunkt	Richtung 1	Richtung 2	Richtung 3
tuxhell(A)	90°	190°	
tuxhell(B)	25°	205°	
tuxhell(C)	95°	170°	350°
tuxhell(D)	80°	260°	

(4.2)

Bei den Bildern *tuxdunkel* und *tuxrausch* gelten diese Winkelangaben genauso, da diese Bilder lediglich andere Abbildungseinflüsse für das Bild *tuxhell* darstellen. Um nicht unnötig viele Grafiken für die Analyse zeigen zu müssen, beschränkt sich hier die Analyse jeweils auf einen markierten Punkt. Sofern es nicht anders vermerkt ist, wird immer der Punkt *tuxhell*(C) benutzt, der einen T-Verbindungspunkt darstellt.

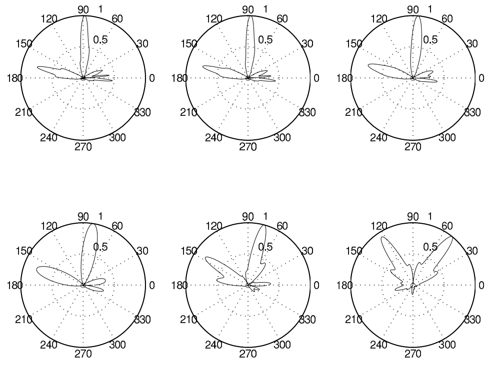


Abbildung 4.25: *tuxhell*(C) analysiert mit der Methode der Polarpyramiden.

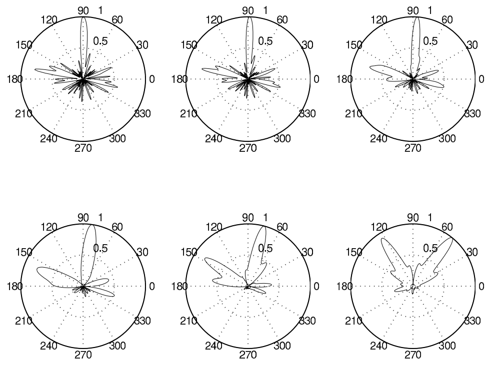


Abbildung 4.26: *tuxrausch*(C) analysiert mit der Methode der Polarpyramiden.

Polarpyramide Die verschiedenen Polarpyramiden des markierten Punktes C sind in den Abbildungen 4.25, 4.26 und 4.27 zu finden. Es kann hier festgestellt werden, dass die Maxima bei allen Bildern entsprechend denen der in Tabelle 4.2 angegebenen liegen. Auffällig ist für *tuxrausch*(C), dass zu den wesentliche drei Maxima viele kleinere lokale Maxima entstehen, die auf das Rauschen zurückzuführen sind. Die leichte Abdunkelung in *tuxdunkel* verändert die Richtungsdarstellungen der Polarpyramiden nur unwesentlich.

Steuerbare Filter Wie man in den Abschnitten zuvor gesehen hat, ist das Verfahren der Steuerbaren Filter für die Orientierungsanalyse geeignet. Da es sich bei *tuxhell*(C) um einen T-Verbindungspunkt handelt, kann man also keine Richtungen analysieren. Dennoch ist es möglich etwas über das Verhalten bezüglich der Veränderungen auszusagen. Betrachtet man die drei

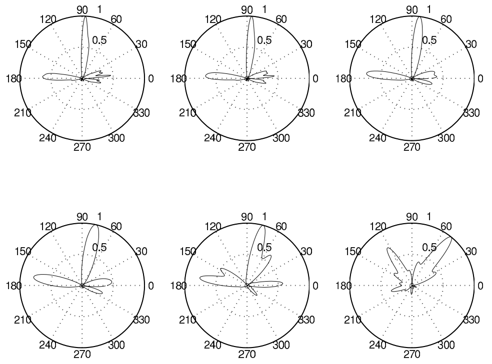


Abbildung 4.27: *tuxdunkel*(C) analysiert mit der Methode der Polarpyramiden.

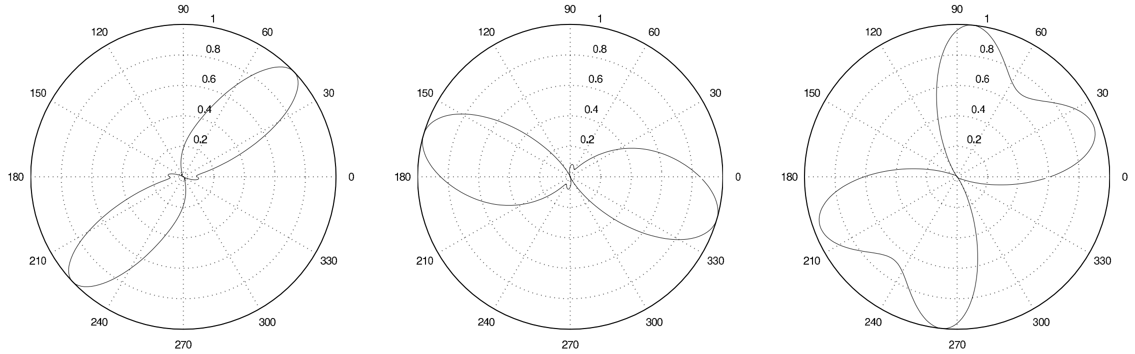


Abbildung 4.28: *tuxhell*(C), *tuxrausch*(C) und *tuxdunkel*(C) analysiert mit der Methode der Steuerbaren Filter.

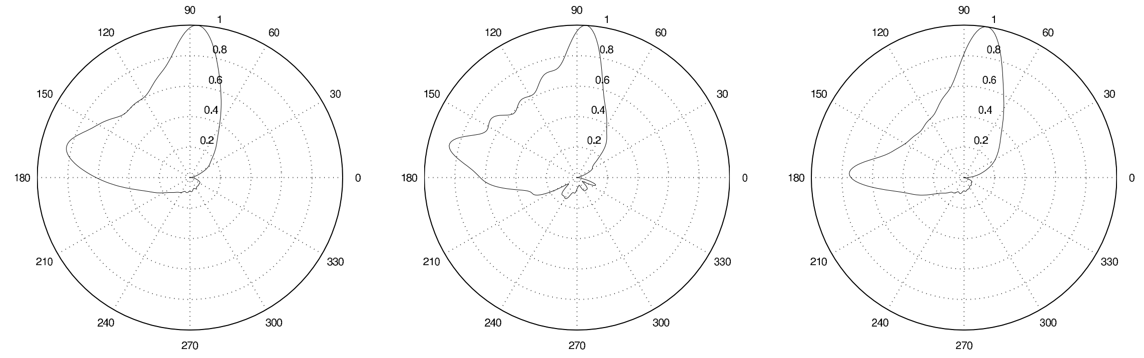
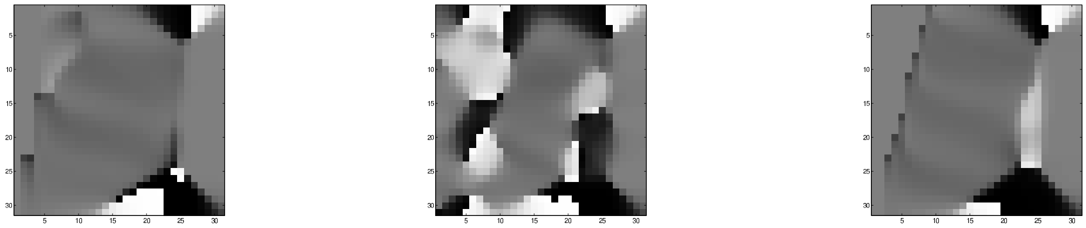


Abbildung 4.29: *tuxhell*(C), *tuxrausch*(C) und *tuxdunkel*(C) analysiert mit der Methode der Wedgefilter.

Energieplots der Abbildung 4.28, so werden deutliche Veränderungen festgestellt. Im Fall des Rauschens wird sogar eine völlig andere Orientierung detektiert, dies ist eine Konsequenz der hohen Ableitungsstufe, es gilt im Allgemeinen: Je höher die Stufe der Ableitung desto größer ist die Anfälligkeit auf Rauschen. Für *tuxdunkel*(C) werden die vorher eindeutigen Maxima aufgelöst, dies lässt den Schluß zu, dass das Rauschen in *tuxdunkel* deutlich geringer sein muss, als das Rauschen in *tuxrausch*.

Wedgefilter Der Wedgefilter hat bei dieser T-Verbindung leichte Probleme bei der Detektion. Man erkennt dieses daran, dass der rechte Übergang einen geringen Kontrast aufweist, dort wird eine geringere Energie festgestellt. Ein drittes Maximum ist kaum auszumachen. Die Maxima in den verschiedenen Richtungen bleiben bei den Veränderungen des Bildes erhalten.

Strukturtensor Bei der Analyse des Strukturtensors hinsichtlich seines Stabilitätsverhaltens, wird der Punkt *tuxhell*(D) benutzt. Er liegt auf einer vertikalen Kante, die leicht im Uhrzeigersinn gedreht ist. In der Abbildung 4.30 kann man an den berechneten Winkeln sehen, dass die Veränderungen des Bildes *tuxhell* nur eine geringe Wirkung auf die Orientierung haben. Es wird eine Winkelvarianz von 4° festgestellt. Die Bestimmtheitswerte bzw. die Beträge der zugehörigen Orientierungsvektoren sind, im Verhältnis zu den für *blox*(A-D) aufgetretenen, sehr groß. Lediglich bei *tuxdunkel* nimmt die Bestimmtheit der Orientierung ein wenig ab. Diese Betrachtung läßt den Schluß zu, dass der Strukturtensor sehr stabil bezüglich leichten Bildveränderungen ist.



$$\begin{pmatrix} 1070.8 & -198.08 \\ -198.08 & 49.5 \end{pmatrix} \equiv 79.9^\circ \quad \begin{pmatrix} 1493.3 & -355.5 \\ -355.5 & 153.7 \end{pmatrix} \equiv 76.0^\circ \quad \begin{pmatrix} 285.812 & -57.7156 \\ -57.7156 & 17.5801 \end{pmatrix} \equiv 78.3^\circ$$

Abbildung 4.30: Strukturtensoren für den Punkt *tuxhell*(D) in den Bildern (v.l.n.r.) *tuxhell*, *tuxrausch* und *tuxdunkel*. Die zugehörigen Orientierungswinkel stehen jeweils rechts neben dem Tensor.

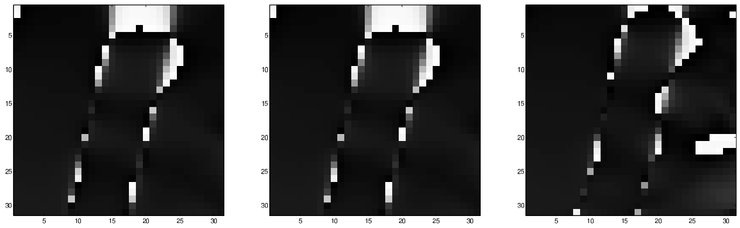


Abbildung 4.31: *tuxhell*(C), *tuxrausch*(C) und *tuxdunkel*(C) analysiert mit der Methode des Monogenen Signals. Die ermittelten Winkel sind: 86.3°, 82.3° und 89.7°.

Monogenes Signal Das Verhalten des Monogenen Signals auf die verschiedenen Einflüsse für den Punkt *tuxhell*(D) ist in Abbildung 4.31 dargestellt. Die Winkel variieren nur gering um die in Bild *tuxhell* gefundene Orientierung, der Unterschied beträgt maximal 4.0°.

Kanten

Phasenkongruenz Die Methode der Phasenkongruenz liefert, wie man in Abbildung 4.32 sieht, ein passables Ergebnis in Bezug auf das Rauschen. Es werden noch alle Kanten identifiziert, lediglich an einigen Stellen erkennt man, dass das Maß für die Phasenkongruenz abnimmt und eine Kante unterbrochen wird (rechtes Bild im linken Bildteil). Auch hier findet man die Aussage bestätigt, dass die Struktur eines Bildes im Wesentlichen durch die Phase bestimmt ist, denn das Phasenkongruenz-Bild von *tuxdunkel* unterscheidet sich kaum von dem von *tuxhell*. Man schließt, dass das Verfahren der Phasenkongruenz stabil bezüglich der hier betrachteten

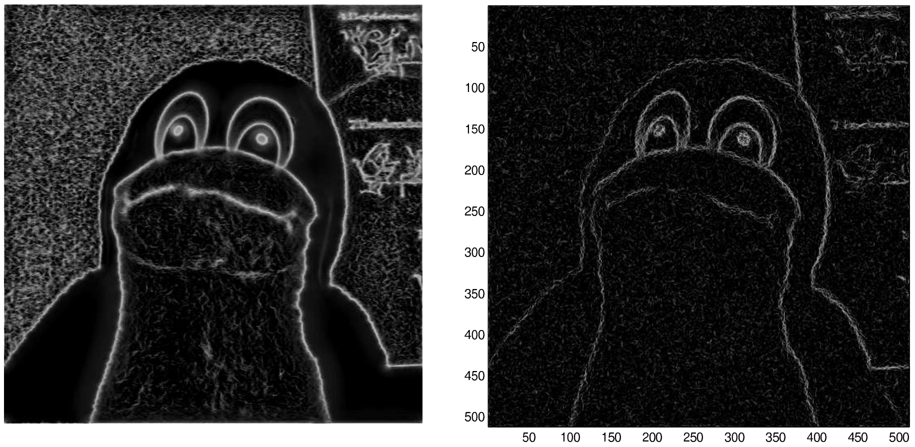


Abbildung 4.32: linkes Bild: *tuxdunkel* , rechtes Bild: *tuxrausch*. Beide Bilder sind mit dem Verfahren der Phasenkongruenz behandelt worden.

Veränderungen des Eingabebildes ist.

4.2.3 Lokalisation

Unter der Lokalisation von Merkmalen wird hier das Verhalten der Verfahren gegenüber Verschiebungen der markierten Punkte verstanden. Es ist also von Interesse, wie sich die Verfahren verhalten, wenn z.B. die vorgegebenen Koordinaten eines markierten Punktes nur ungefähr mit den tatsächlichen übereinstimmen. Einige Verfahren benötigen als Parameter einen Aufsetzpunkt, dieser muss durch ein davorgestelltes Verfahren ermittelt werden, welches in der Regel nur ungenaue Koordinaten liefern wird. Es werden alle Verfahren ausser das der Phasenkongruenz betrachtet. Für die Analyse wird der Verbindungspunkt $blox(\mathrm{A})$ verwendet, für diesen sind alle Verfahren sinnvoll anwendbar.

Polarpyramide Beim Verfahren der Polarpyramide werden zwei Richtungen detektiert (siehe Abbildung 4.33), die aber nicht mehr um 180° auseinanderliegen. Die Richtungen, die angegeben werden, zeigen jeweils in Richtung der horizontalen Kante. Eine qualitative Aussage für den verschobenen Aufpunkt kann deshalb nur dahingehend getroffen werden, dass es sich bei dem Punkt $blox(\mathrm{A})$ um einen L-Verbindungspunkt handelt.

Steuerbare Filter Abbildung 4.34 zeigt, dass die Methode der Steuerbaren Filter die korrekte Orientierung detektiert. Das ist darauf zurückzuführen, dass die verwendeten Ableitungsoperato-

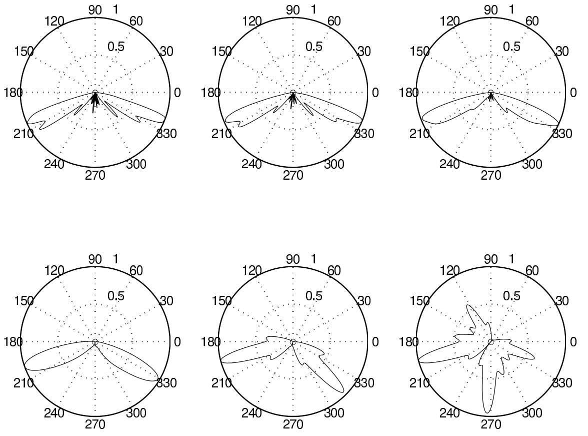


Abbildung 4.33: Von der Polarpyramide detektierte Richtungen bei Verschiebung des zu analysierenden Aufpunktes um vier Pixel nach oben.

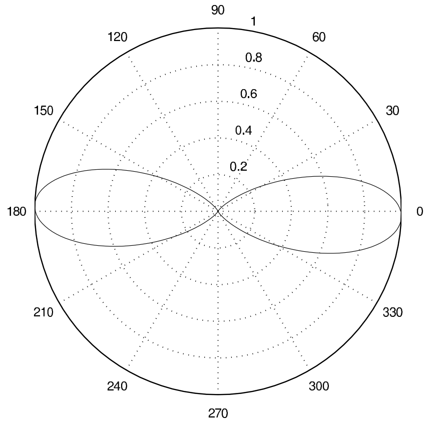


Abbildung 4.34: Von den Steuerbaren Filtern gelieferte Orientierung bei Verschiebung des zu analysierenden Aufpunktes um vier Pixel nach oben.

ren eine gewisse Filterbreite besitzen. Bei der hier verwendeten vierten Gaussfunktionsableitung wird eine Filterbreite von 13 Pixeln erzeugt, in der die Filter eine glättende Wirkung haben.

Wedgefilter Für die Wedgefilter (Abbildung 4.35) gilt Ähnliches wie für die Polarpyramide. Die detektierten Richtungen zeigen ebenfalls in Richtung der horizontalen Kante. Die horizontale Kante kann nicht mehr als solche identifiziert werden, dennoch sind zwei wesentliche Maxima

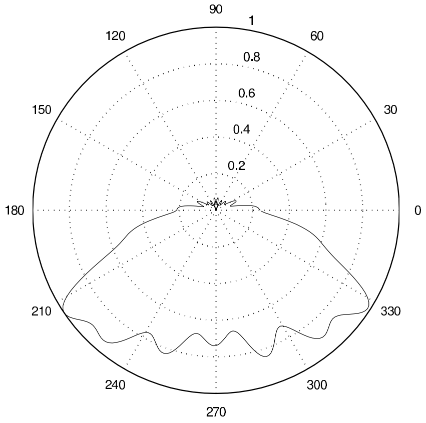
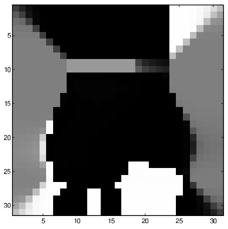


Abbildung 4.35: Von den Wedgefiltern gelieferte Richtungen bei Verschiebung des zu analysierenden Aufpunktes um vier Pixel nach oben.

zu erkennen. Diese Maxima klassifizieren den Verbindungspunkt *blox*(A) ebenfalls als einen L-Verbindungspunkt.



$$\begin{pmatrix} 0.0030 & -0.0594 \\ -0.0594 & 1.9100 \end{pmatrix} \equiv 1.7^\circ$$

Abbildung 4.36: Ergebnis des Strukturtensors in Bezug auf die horizontale Kante *blox*(A) mit einer Verschiebung um vier Pixel nach oben. Zu beachten ist hier, dass der Betrag des zugehörigen Orientierungsvektors gering ist, d.h. das Bestimmtheitsmaß für die berechnete Orientierung ist ebenfalls gering.

Strukturtensor Betrachtet man den Betrag des zugehöringen Orientierungsvektors des Strukturtensors aus Abbildung 4.36, so stellt man fest, dass dieser im Verhältnis zu dem aus 4.16 rela-

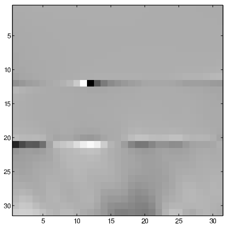


Abbildung 4.37: Orientierungsbild der Umgebung des um vier Pixel nach oben verschobenen Aufpunktes. Die Orientierungen wurden mittels des Monogenen Signals berechnet. Der gemessene Winkel für den verschobenen Aufpunkt beträgt 179.7°

tiv gering ist. Die Bestimmtheit der Orientierung ist also geringer. Trotzdem gibt der berechnete Orientierungswinkel eine korrekte Orientierung der Kante wieder. Bei der Anwendung der Strukturtensors werden Binomialfilter zur Glättung eingesetzt. Diese bewirken eine Homogenisierung der Orientierungen in einer Umgebung, die von der Größe dieser Filter abhängig ist. Die hier benutzten Binomialfilter haben eine Größenordnung von 13^2 Pixel, es ist also anzunehmen, dass innerhalb einer Umgebung mit dem Radius 13 Pixel, die Orientierungen ähnlich sind.

Monogenes Signal Die Ermittlung der Orientierungen durch das Monogene Signal verhält sich ähnlich der des Strukturtensors (siehe 4.37). Da die Berechnung des Signals mittels Bandpassfiltern im Frequenzraum erfolgt, und diese im Ortsraum, aufgrund des Ähnlichkeitstheorems der Fouriertransformation, eine unendliche Ausdehnung besitzen, werden ebenfalls Ähnlichkeiten in Umgebungen erzeugt. Für die hier verwendete Oktave und Bandbreite, verhält sich das Monogene Signal gutartig. Es liefert einen Orientierungswinkel, der sich lediglich um 1° von dem des nichtverschobenen Aufpunktes unterscheidet.

4.2.4 Komplexität

Unter der Komplexität eines Verfahrens versteht man den zur Berechnung nötigen Aufwand an Betriebsmitteln wie Rechenzeit und Speicherplatz. Da im Rahmen dieser Arbeit lediglich einzelne Bilder bzw. Ausschnitte aus diesen Bildern betrachtet werden, ist die Betrachtung des Speicherplatzverbrauches nicht von besonderem Interesse. Deshalb wird im Folgenden nur die Rechenzeit bzw. das Laufzeitverhalten betrachtet.

Bei einigen Verfahren muss zwischen der Konstruktionsphase und der Anwendungsphasen unterschieden werden. Beispielsweise ist für die Wedgefilter der Aufwand für die Konstruktion und für die Anwendung der erzeugten Filterbasis zu betrachten.

Polarpyramide Bei der Polarpyramide muss für jeden gegebenen Punkt eine Polartransformation durchgeführt werden. Da die Polartransformation eine Koordinatentransformation ist, kann diese vorher berechnet werden, um dann die Ergebnisse in einer Tabelle zu speichern. Für die Komplexitätsbetrachtung kann deshalb dieser Aufwand vernachlässigt werden, da das Nachschauen in der Tabelle als Einmalaufwand quasi vernachlässigbar ist. Geht man davon aus, dass man S verschiedene Winkel darstellen kann, so benötigt man für die Herstellung der ersten Stufe der Pyramide mit einer Filterbreite W (Breite der Gaußfiltermaske) und den Radien R_{max} und R_{min} die Anzahl von $SW(R_{max} - R_{min})$ Schleifendurchläufen. Die erste Stufe der Pyramide ist ein eindimensionales Feld der Länge S. Für die Erstellung der Pyramide werden in jedem Schritt $W\log(S)$ Durchläufe gebraucht. Für das Interpolieren auf die vollständige Größe wird derselbe Aufwand benötigt. Es ergibt sich also insgesamt ein Aufwand von:

$$SW(R_{max} - R_{min}) + 2W\log(S) \in O(SWR), \tag{4.3}$$

für ein Bild mit der Größe N^2 kommt noch ein Aufwand von $O(N^2)$ hinzu.

Steuerbare Filter Um die hier verwendeten Filter der Größenordnung A^2 zu erzeugen, werden für die Basisfilter von G_4 genau $5A^2$ Schleifendurchläufe und für H_4 genau $6A^2$ benötigt. Es ergibt sich also ein Konstruktionsaufwand von:

$$5A^2 + 6A^2 = 11A^2 \in O(A^2). \tag{4.4}$$

Für jeden Punkt eines gegebenen quadratischen Bildes der Größenordnung N^2 werden bei der Anwendung genau 11 Faltungen durchgeführt, welche einen Aufwand von

$$11N^2A^2 \in O(N^2A^2) \tag{4.5}$$

bedeuten. Werden für jeden Bildpunkt S Winkel betrachtet, so wird ein Aufwand von

$$11SN^2 \in O(SN^2) \tag{4.6}$$

für die Erzeugung aller gerichteten Energien gebraucht. Es ergibt sich ein Anwendungsaufwand von

$$11N^2A^2 + 11SN^2 = 11N^2(A^2 + S) \in O(N^2(A^2 + S)). \tag{4.7}$$

Wedgefilter Für den Wedgefilter gilt ähnliches wie für die Steuerbaren Filter. Sollen M Filterpaare mit der Filtergröße A^2 konstruiert werden, so entsteht ein Aufwand von

$$(2M)A^2 \in O(MA^2). \tag{4.8}$$

Der Aufwand für die Anwendung bestimmt sich wieder aus der Anzahl der Faltungen, es ergibt sich ein Aufwand von

$$(2M)N^2 \in O(MN^2). \tag{4.9}$$

Mit der Bestimmung der gerichteten Energie über S Winkel für ein quadratisches Bild der Größe N^2 wird ein gesamter Anwendungsaufwand von

$$(2M)N^2 + (2M)SN^2 \in O((MS)N^2) \tag{4.10}$$

gebraucht.

Strukturtensor Sei ein quadratisches Bild der Größenordnung N^2 gegeben. Sei A^2 die Größe der Ableitungsfilter. Durch zwei Faltungen mit dem Aufwand $2N^2A^2$ werden zwei Richtungsableitungen des Bildes berechnet. Für die Konstruktion des Strukturtensors werden $3N^2$ nichtlineare Punktoperationen benötigt (Multiplikationen). Die so entstandenen drei Bilder werden durch einen Glättungsfilter (Binomialfilter) der Größe G^2 mit einem Aufwand von $3N^2G^2$ geglättet. Für den gesamten Aufwand ergibt sich

$$2N^2A^2 + 3N^2 + 3N^2G^2 \in O(N^2(A^2 + G^2)). \tag{4.11}$$

Die Berechnung der Orientierungswinkel erfolgt mit einem Aufwand von N^2 aus den Orientierungsvektoren.

Phasenkongruenz Bei der Phasenkongruenz wird ein quadratisches Bild der Größe N^2 durch eine FFT in den Frequenzraum transformiert. Dies geschieht mit einem Aufwand von $N^2 \log_2 N$. Für die Berechnung der Phasenkongruenz werden insgesamt sechs Orientierungen und vier verschiedene Waveletskalen betrachtet. Es ergibt sich für die Berechnung der Phasenkongruenz gemäß Kovesi ein Gesamtaufwand, der hier aus Gründen der Übersicht nicht aufgeschlüsselt werden soll, von

$$N^2 \log_2 N + 24(2N^2 \log_2 N + N^2) + 6N^2 \in O(N^2). \tag{4.12}$$

Monogenes Signal Sei N die Breite des zu betrachtenden quadratischen Bildes. Für eine Fouriertransformation des Bildes mittels der FFT werden $N^2 \log_2 N$ Operationen benötigt. Die Vorbereitung des radialen Bandpassfilters wird mit einem Aufwand von N^2 im Frequenzraum erreicht. Insgesammt werden drei Faltungen berechnet, die im Frequenzraum mittels der komplexen Multiplikation durchgeführt werden. Dies geschieht mit einem Aufwand von $3N^2$. Um nun das Monogene Signal im Ortsraum zu erzeugen, werden zwei inverse Fouriertransformationen mit dem Aufwand $2N^2\log_2 N$ verwendet. Insgesamt erhält man einen Aufwand von

$$4N^2\log_2 N + 4N^2 \in O(N^2). \tag{4.13}$$

Für die Berechnung der Orientierungswinkel werden zusätzlich N^2 Operationen benötigt. Die Ergebnisse der Komplexitätsbetrachtung sind in der Tabelle 4.38 zusammengefaßt.

Verfahren	Konstruktionsaufwand	Anwendungsaufwand
Polarpyramide		$O(SWR)$
Steuerbare Filter	$5A^2 + 6A^2 = 11A^2 \in O(A^2)$	$11N^2A^2 + 11SN^2 = 11N^2(A^2 + S) \in O(N^2(A^2 + S))$
Wedgefilter	$(2M)A^2 \in O(MA^2)$	$(2M)N^2 + (2M)SN^2 \in O((MS)N^2)$
Strukturtensor		$2N^2A^2 + 3N^2 + 3N^2G^2 \in O(N^2(A^2 + G^2))$
Phasenkongruenz		$N^2 \log_2 N + 24(2N^2 \log_2 N + N^2) + 6N^2 \in O(N^2)$
Monogenes Signal		$4N^2 \log_2 N + 4N^2 \in O(N^2)$

Abbildung 4.38: Übersicht über die Komplexität der einzelnen Verfahren in Bezug auf ein Bild der Größenordnung N^2. A^2 bezeichnet die Größe eines Ableitungsfilters, G^2 die Größe eines Glättungsfilters, M die Anzahl von Basisfunktionen, W die Breite eines Filters und R die Höhe eines Filters und S die Anzahl der betrachteten Winkel.

Kapitel 5

Diskussion

5.1 Mathematische Klassifikation

Die einzelnen Verfahren bedienen sich zum Teil verschiedener mathematischer Ansätze, die im Folgenden erläutert werden, sie können in Klassen eingeteilt werden,

5.1.1 Verfahren zur Steuerung von Filtern

Das Verfahren der Steuerbaren Filter kann als ein grundlegendes Verfahren für die Steuerbarkeit von Filterfunktionen benutzt werden. Die Idee des Verfahrens ist, einen Filter, der für einen gewissen Winkel einen Energiewert liefert, zu steuern. Das Steuern setzt voraus, dass man eine Filterbasis hat, mit der man für fehlende Richtungen, einen geeigneten Filter durch Linearkombination der Basisfilter erzeugen kann. Hat man nur eine *kleine* Basis, so lässt sich der Aufwand für die sonst zahlreichen Filteranwendungen reduzieren. Diese Methodik wird ebenfalls bei den Wedgefiltern benötigt. Sie sind jedoch schon mit dem Gedanken der Richtungsdetektion entworfen worden. Ausgehend von der *Keilform* eines Wedgefilters, wird dieser mittels einer Fourierreihe dargestellt. Es hat sich gezeigt, dass Winkel-Funktionen, die sich als Fourierreihe darstellen lassen, sich für die Steuerbarkeit eignen. Bei den Wedgefiltern ist die größe der Filterbasis abhängig von der Winkelauflösung. Für beide Verfahren wird zusätzlich die Steuerbarkeit der Hilberttransformierten benötigt. Diese wird für die gerichtete Energieberechnung gebraucht.

5.1.2 Optimierte Verfahren

Die Polarpyramide bedient sich verschiedener Mechanismen zur Beschleunigung der Berechnung von Richtungen. Zunächst wird eine Umgebung polartransformiert, was Vorteile bringt, da die weiteren Verarbeitungsschritte lediglich in einem rechteckigen Fenster stattfinden. Ein weiterer Schritt veringert die Dimension des Fensters durch eine geeignete Mittelwertbildung. Das so entstandene eindimensionale Signal wird dann in den einzelnen Stufen einer Gaußpyramide geglättet. Das Konzept der Pyramide ist wiederum ein Verfahren, welches Geschwindigkeitsvorteile bei der Faltung mit großen Glättungsmasken bringt.

5.1.3 Gradientenbasiertes Verfahren

Bei der Betrachtung von einfachen Nachbarschaften, wie es beim Strukturtensor der Fall ist, wird diejenige Orientierung, die die geringste Abweichung von der Gradientenrichtung hat, als optimal angesehen. Wählt man einen beliebig orientierten Vektor, so entsteht ein Maximierungsproblem, welches in einer Tensoralgebra gelöst werden kann. Es zeigt sich, dass man die lokale Struktur in einem Tensor (dem Strukturtensor) kodieren kann. Die Orientierungsbestimmung reduziert sich dann auf die Analyse von Eigenwerten, welche im zweidimensionalen Fall trival ist.

5.1.4 Phasenbasierte Verfahren

Die Verfahren der Phasenkongruenz und des Monogenen Signals nutzen die lokale Phaseninformation von Bildern aus, um Aussagen über Merkmale wie Kanten oder Orientierungen zu treffen. Dazu werden verschiedene Transformationen verwendet, die es ermöglichen, die Phaseninformation lokal zu betrachten. Bei der Phasenkongruenz wird die Wavelettransformation benutzt, die als lokale Fouriertransformation verstanden werden kann. Beim Monogenen Signal wird die Riesztransformation verwendet, sie liefert den sogenannten *split of identity*, welcher die Trennung der Bildinformation in Amplitude und Phase für zweidimensionale Bilder leistet.

5.2 Anwendungsbezogene Klassifikation

In der Tabelle 5.2 spiegeln sich die Verwendungszwecke der einzelnen Verfahren wider. Dort sind die Verwendungzwecke eingetragen, die sich im Rahmen dieser Arbeit herausgestellt haben. Sind bei den Merkmalen die Sterne geklammert, so wird das betroffene Merkmal erst durch eine geeignete Weiterverarbeitung festgestellt. Die Merkmalsanalyse ist leider nicht ganz vollständig, der Bereich der Segmentierung musste leider ausgelassen werden. Weiterhin sind dort die Voraussetzungen, die benötigten Parameter und die Wahl der Parameter beschrieben. Ebenso ist das Verhalten bezüglich der verschiedenen Störungen und Ungenauigkeiten dargestellt. Die Klassifikation in *gut* und *schlecht* ist als relatives Maß unter den Verfahren zu verstehen. Ist bei den Eigenschaften ein *gut* geklammert, so bedeutet dieses, dass ein Teil der qualitativen Aussage des Verfahrens bei der Analyse verlorengegangen ist. Die Beurteilung der Verfahren hinsichtlich ihrer Geschwindigkeit ist der Komplexitätsanalyse entnommen worden.

5.3 Ausblick

Die Analyse von Verfahren hinsichtlich ihres Verhaltens der lokalen Strukturanalyse ist ein umfangreiches Gebiet. Ein Problem was sich bei der Bearbeitung dieses Themas ergeben hat, ist die Berücksichtigung der verschiedenen Konfigurationen der einzelnen Verfahren. Um die Verfahren *vergleichbar* zu machen, sind für alle Verfahren ähnliche Bearbeitungsvoraussetzungen zu schaffen. Dies gelingt aber nur zum Teil, da einige Verfahren nicht sehr variabel sind. Beispielsweise ergeben sich Probleme bei den Steuerbaren Filtern durch Einbeziehung des Aufpunktes bei der Analyse der Orientierung. Dagegen kann man bei den Verfahren Wedgefilter und Polarpyramide die Umgebung des Aufpunktes ausschließen. Desweiteren macht die Vielzahl der möglichen Parametereinstellungen Probleme, es wurden zahlreiche Experimente gebraucht, um geeignete Parametereinstellungen für die in dieser Arbeit benutzten Bilder zu finden. Die Qualität der Verfahren ist deshalb auch abhängig von der Art der Bilder, dies begründet auch die Wahl der umfangreichen Bildermenge (4.1).

Der Umfang der Analyse hat sich ebenfalls als problematisch herausgestellt, jedes Verfahren für sich sollte mit einer entsprechend ausführlichen Analyse begutachtet werden. Die in dieser Arbeit vorgestellten Analyseergebnisse geben nur einen Teil der benötigten Analysen wieder. Für weitere Arbeiten, die sich mit anderen Merkmalen oder Verfahren beschäftigen, sollte die anwendungsbezogene Klassifikation aus 5.2 erweitert werden. Möglicherweise kann man eine Datenbank aufbauen, die die Zuordnung von Merkmalen zu den bisher bekannten Verfahren wiedergibt.

	Polar	Steuer	Wedge	Tensor	Phase	Monogen
Voraussetzungen						
Keypoint	$\star$	$\star$	$\star$	$\star$		$(\star)$
Bildbereich					$\star$	$(\star)$
Parameter 1	Radien, Filterbreite	Ableitungstufe	Anzahl Stützstellen	Glättungsfilter		Oktave
Wahl Para. 1	$R_{min} = 5$, R_{max}=15,	4-te Ableitug	20 Basispaare	Binomial(13)		4-te Oktave
Parameter 2	Filterbreite		minimaler Radius			Bandbreite
Wahl Para. 2	$W = 5$		$R_{min} = 5$			0.55
Merkmale						
Richtung	$\star$		$\star$			
Orientierung	$(\star)$	$\star$	$(\star)$	$\star$		$\star$
Kanten/Linien	$(\star)$	$(\star)$	$(\star)$	$(\star)$	$\star$	$\star$
Segmentierung						
Skala	$\star$				$\star$	
Eigenschaften						
Rauschen	gut	schlecht	gut	gut	gut	gut
Kontrast	gut	schlecht	gut	gut	gut	gut
Lokalistation	(gut)	gut	(gut)	gut		gut
Geschwindigkeit	schnell	mittelmäßig	mittelmäßig	schnell	langsam	schnell

Abbildung 5.1: Klassifikation und Ergebnisse der Analyse bezüglich der Verfahren: Polarpyramiden (Polar), Steuerbare Filter (Steuer), Wedgefilter (Wedge), Strukturtensor (Tensor), Phasenkongruenz (Phase) und Mongenes Signal (Monogen). Geklammerte $(\star)$ bei den Voraussetzungen bedeuten, dass das jeweilige Verfahren entsprechend auf diesen Verarbeitungsbereich eingeschränkt wird. Bei den Merkmalen bedeutet $(\star)$, dass das Verfahren dieses Merkmal nicht direkt, sondern erst nach einer Weiterverarbeitung liefert.

Kapitel 6

Anhang Beweise

Beweise für Lemmata, die zur Erzeugung der Wedgefilter nötig sind Im Folgenden gilt immer:

- $w = (w_n)$ sind die Gewichte
- $\Lambda = diag(\lambda(\phi_1), \ldots, \lambda(\phi_m))$, mit $\phi_m = 2\pi m/M$
- $m \times n -$ Matrix $\quad S = (S_{mn}) = (\sin(2\pi nm/M))$
- $m \times n -$ Matrix $\quad C = (C_{mn}) = (\cos(2\pi nm/M))$

Lemma 6

$$\Lambda C w = \begin{pmatrix} \lambda(\phi_1) \sum_{n=1}^{N} w_n \cos(2\pi n \cdot 1/M) \\ \vdots \\ \lambda(\phi_M) \sum_{n=1}^{N} w_n \cos(2\pi n \cdot M/M) \end{pmatrix} \tag{6.1}$$

Beweis

$$\begin{aligned}
\Lambda C w &= \begin{pmatrix} \lambda(\phi_1) & & \\ & \ddots & \\ & & \lambda(\phi_M) \end{pmatrix} \begin{pmatrix} \cos(2\pi \cdot 1 \cdot 1/M) & \cdots & \cos(2\pi \cdot N \cdot 1/M) \\ \vdots & \ddots & \vdots \\ \cos(2\pi \cdot 1 \cdot M/M) & \cdots & \cos(2\pi \cdot N \cdot M/M) \end{pmatrix} \begin{pmatrix} w_1 \\ \vdots \\ w_N \end{pmatrix} \\
&= \begin{pmatrix} \lambda(\phi_1) & & \\ & \ddots & \\ & & \lambda(\phi_M) \end{pmatrix} \begin{pmatrix} \sum_{n=1}^{N} w_n \cos(2\pi n \cdot 1/M) \\ \vdots \\ \sum_{n=1}^{N} w_n \cos(2\pi n \cdot M/M) \end{pmatrix} \\
&= \begin{pmatrix} \lambda(\phi_1) \sum_{n=1}^{N} w_n \cos(2\pi n \cdot 1/M) \\ \vdots \\ \lambda(\phi_M) \sum_{n=1}^{N} w_n \cos(2\pi n \cdot M/M) \end{pmatrix}
\end{aligned}$$

•

Lemma 7

$$\sum_{m=1}^{M} \lambda^2(\phi_m) h_e^2(\phi_m) = |\Lambda C w|^2 \tag{6.2}$$

Beweis

$$\begin{aligned}
\sum_{m=1}^{M} \lambda^2(\phi_m) h_e^2(\phi_m) &= \sum_{m=1}^{M} \lambda^2(\phi_m) \left[\sum_{n=1}^{N} w_n \cos(n\phi_m) \right]^2 \\
&= \sum_{m=1}^{M} \lambda^2(\phi_m) \left[\sum_{n=1}^{N} w_n \cos(2\pi nm/M) \right]^2 \\
&= < \Lambda C w, \Lambda C w > \\
&= |\Lambda C w|^2
\end{aligned}$$

•

Lemma 8

$$\sum_{m=1}^{M} \lambda^2(\phi_m) E(\phi_m) = |\Lambda C w|^2 + |\Lambda S w|^2 \tag{6.3}$$

Beweis

$$\begin{aligned}
\sum_{m=1}^{M} \lambda^2(\phi_m) E(\phi_m) &= \sum_{m=1}^{M} \lambda^2(\phi_m) \left[h_e^2(\phi_m) + h_o^2(\phi_m) \right] \\
&= \sum_{m=1}^{M} \lambda^2(\phi_m) h_e^2(\phi_m) + \sum_{m=1}^{M} \lambda^2(\phi_m) h_o^2(\phi_m) \\
&= |\Lambda C w|^2 + |\Lambda S w|^2
\end{aligned}$$

•

Kapitel 7

Anhang Programme

In diesem Kapitel werden die Programme, die für die experimentelle Ausarbeitung gebraucht worden sind, vollständig dargelegt. Die Programme laufen unter Matlab 5.3 mit der Visulisierungs-Toolbox für Bilder.

7.1 Polarpyramiden

Das Programm *erzeugepyramide* beinhaltet die Aufrufe der entsprechenden Unterprogramme zur Erzeugung einer Polarpyramide bis zur fünften Stufe. Die Polartransformaiton und die Erzeugung der ersten Stufe für die Pyramide wird durch die Routine *binomialaveraging* durchgeführt. Die einzelnen Schichten der Pyramide und die Anwendung findet in *gausspyramid* statt. Die weiteren Routinen sind kleine Hilfsprogramme.

```
function erzeugepyramide(pic, rmin, rmax)
%%%
%%% erzeugt eine Polarpyramide bis zur fuenften Stufe
%%%

[Th, OrigX, X, DX, num, radmin, radmax]=
           binomialaveraging(double(pic),rmin,rmax);
gausspyramid(Th, X, DX);

function [Th, Origx, X, DX,Num, radmin, radmax] = binomialaveraging(in,radmin,radmax)
%%%%%
%%%%% This function uses Statistics to find out the orientations with
%%%%% local maximal  values.
%%%%% in      : input (spectrum) image
%%%%% [radmin:radmax]: The area in which the statistics should be taken.
%%%%% angstep      : step of rotation
%%%%% width      : Mask angle width in degree
%%%%%
%%%%%
```

```
%%%%% Feb-13-98 NOTE: At the beginning we do not multiply the Num with
%%%%%                 the weighting coefficients. It seems worse than
%%%%%                 the result of rectangle filter. After the
%%%%%                 multiplication it is O.K. The possible reason is
%%%%%                 that weighted Num works better in normalization.
%%%%%
%%%%%
angstep = pi/180;

width4  = 4;
weight4 = [1 4 6 4 1]./16;

width6  = 6;
weight6 = [1  6  15  20  15  6  1]./64;

width10  = 10;
weight10 = [1 10 45 120 210 252 210 120 45 10 1]./1024;

width20  = 20;
weight20 = [1 20 190 1140 4845 15504 38760 77520 125970 167960 184756 ...
167960 125970 77520 38760 15504 4845 1140 190 20 1]./1048576;

width  = width6 ;
weight = weight6 ;

[row, col] = size(in);
     if row ~= col
  ' image must have the same size'
  return;
     end
  win2 = floor(row/2)+1;
  if win2 + radmax > row
  ' image is too small or radius is too big'
  return;
  end

      X   = zeros(1,floor(2*pi/angstep)+1);
      Num = zeros(1,floor(2*pi/angstep)+1);

Th = 0: angstep : 2*pi;
Origx = zeros(size(Th));

%%%%
%%%% First quadrant
%%%%
     for x = win2+1: win2+radmax,
         for y = win2+1: win2+radmax,
             r = sqrt((x-win2).^2 + (y-win2).^2);
     if r <= radmax & r > radmin
```

```
                phi = round(acos((x-win2)/r)*180/pi);
                th =  [phi-width/2 : phi+width/2];
                ind = floor(mod(th,(2*pi/angstep+1)))+1;
                X(ind) =  X(ind) + in(x,y).*weight;
                Num(ind) =  Num(ind)+1.*weight;
             end
     end
     end

%%%%
%%%% Second quadrant
%%%%
     for x = win2-radmax: win2-1,
     for y = win2+1: win2+radmax,
             r = sqrt((x-win2).^2 + (y-win2).^2);
     if r <= radmax & r > radmin
                phi = round(acos((x-win2)/r)*180/pi);
                th =  [phi-width/2 : phi+width/2];
                ind = floor(mod(th,(2*pi/angstep+1)))+1;
                X(ind) =  X(ind) + in(x,y).*weight;
                Num(ind) =  Num(ind)+1.*weight;
              end
     end
     end
%%%%
%%%% Third quadrant
%%%%
     for x = win2-radmax: win2-1,
         for y = win2-radmax: win2-1,
             r = sqrt((x-win2).^2 + (y-win2).^2);
     if r <= radmax & r > radmin
                phi = 360 - round(acos((x-win2)/r)*180/pi) ;
                th =  [phi-width/2 : phi+width/2];
                ind = floor(mod(th,(2*pi/angstep+1)))+1;
                X(ind) =  X(ind) + in(x,y).*weight;
                Num(ind) =  Num(ind)+1.*weight;
             end
     end
     end
%%%%
%%%% Fourth quadrant
%%%%
     for x = win2+1: win2+radmax,
         for y = win2-radmax: win2-1,
             r = sqrt((x-win2).^2 + (y-win2).^2);
     if r <= radmax & r > radmin
                phi = 360 - round(acos((x-win2)/r)*180/pi);
                th =  [phi-width/2 : phi+width/2];
                ind = floor(mod(th,(2*pi/angstep+1)))+1;
```

```
                X(ind) =  X(ind) + in(x,y).*weight;
                Num(ind) =  Num(ind)+1.*weight;
             end
     end
     end
%%%
%%% Four axes
%%%
    y = win2;
    for x = win2+1: win2+radmax,
        r = abs(x-win2);
if r <= radmax & r > radmin
            th =  [0 - width/2 : 0 + width/2];
            ind = floor(mod(th,(2*pi/angstep+1)))+1;
            X(ind) =  X(ind) + in(x,y).*weight;
            Num(ind) =  Num(ind)+1.*weight;
        end
    end
    for x = win2-radmax: win2-1,
        r = abs(x-win2);
if r <= radmax & r > radmin
            th =  [180 - width/2 : 180 + width/2];
            ind = floor(mod(th,(2*pi/angstep+1)))+1;
            X(ind) =  X(ind) + in(x,y).*weight;
            Num(ind) =  Num(ind)+1.*weight;
        end
    end
    x = win2;
    for y = win2+1: win2+radmax,
        r = abs(y-win2);
if r <= radmax & r > radmin
            th =  [90 - width/2 : 90 + width/2];
            ind = floor(mod(th,(2*pi/angstep+1)))+1;
            X(ind) =  X(ind) + in(x,y).*weight;
            Num(ind) =  Num(ind)+1.*weight;
        end
    end
    for y = win2-radmax: win2-1,
        r = abs(y-win2);
if r <= radmax & r > radmin
            th =  [270 - width/2 : 270 + width/2];
            ind = floor(mod(th,(2*pi/angstep+1)))+1;
            X(ind) =  X(ind) + in(x,y).*weight;
            Num(ind) =  Num(ind)+1.*weight;
        end
    end


%%%
```

```
%%% Averaging
%%%
    for index = 1:floor(2*pi/angstep)+1,
        if Num(index) ~= 0
               X(index) =  X(index)/Num(index);
        end
    end

XXX = [ X X X];

Orig  = conv2(XXX,weight,'same');
Origx = Orig(size(X,2)+1:size(X,2)+size(X,2));

if 1
    ramp = [-2:0.25:2];
    Gauss = exp(-ramp.^2);
    Derivat = -2.*ramp.*Gauss;
    XXX = [X X X];
    DDX = conv2( XXX, Derivat, 'same' );
%
%  We need only the local absolute maximal values for finding the orientations.
%
    DX = abs(DDX(size(X,2)+1:size(X,2)+size(X,2)));
end


function  [ ] = gausspyramid(Th,X,DX)
%%%
%%% Feb-10-98 This programm  builds a Gauss oktave pyramid
%%%           given the sampling averaging values from the
%%%           programm of ''binomialaveraging''.
%%%
%%%  Because we use degrees as angular unit the whole number of
%%%  angles are 360 or 361 (if we want to have a whole circle).
%%%  This is not the optimal situation for constructing pyramid.
%%%  (According to \cite{Burt83comm} p533 line 4 from bottom:
%%%  the size should be M \times 2^{n} +1 , where M \in intergel)
%%%
%%%  But we can build the pyramid partly without error.
%%%

% set the generating kernel
Weight5 =    [1  4  6  4  1] ./16;   % for pyramid with factor 2

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% for pyramid with factor 3 we can not simply further apply the
% binomial mask with 6th. order, because the 6.th binomial mask:
% [ 1  6  15  20  15  6  1 ]
% [ a  b   c  d   c  b  a ]
```

```
% does not satisfy the constraint of equal contribution
% (\cite{burt83comm} p.533 line 15 from bottom),  which means
%  a+d+a = b+c  (a+b+c+d+c+b+a = 1)
% But we still want to have a mask similar to Gauss filter
% Thus, we define the following mask.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Weight7 = [3 22 66 82 66 22 3]./264;

% cut the input down to exact one periods. (Maybe unnessisary)
if 1
Th    = Th(1:360);
X    =  X(1:360);
DX    = DX(1:360);
end

%% We have no circular convolution. We try to build a 3 periodic signal
%% to get a short way with the price of some more compuatation.

% Build 3 periods
TTTh   = [Th Th Th];
XXX    = [X X X];
DDDX   = [DX DX DX];

% employ normal linear convolution
XX1    =  conv2(XXX, Weight5, 'same');
DDX1   =  conv2(DDDX,Weight5, 'same');

% cut down only a period to have the result of circular convolution
% subsample them with factor 2 to build the 1th.  level of the pyramid.

Th1    =  TTTh(size(Th,2)+1:2:size(Th,2)+size(Th,2));
X1    =  XX1(size(X,2)+1:2:size(X,2)+size(X,2));
DX1    =  DDX1(size(DX,2)+1:2:size(DX,2)+size(DX,2));

% Using generationg kernel again to expand the signal.
[EX1, ETh1]    =  gaussexpand1d2(X1,Th1) ;
[EDX1,EDTh1]   =  gaussexpand1d2(DX1,Th1);

%% Build 2th. levels of the pyramid.

% Build 3 periods
TTTh1   = [Th1 Th1 Th1];
XXX1   =  [ X1  X1  X1];
DDDX1  =  [DX1 DX1 DX1];

% employ normal linear convolution
XX2    =  conv2(XXX1, Weight5, 'same');
DDX2   =  conv2(DDDX1,Weight5, 'same');
```

```
% cut down only a period to have the result of circular convolution
% subsample them with factor 2 to build the 2th.  level of the pyramid.

Th2    =  TTTh1(size(Th1,2)+1:2:size(Th1,2)+size(Th1,2));
X2    =  XX2(size(X1,2)+1:2:size(X1,2)+size(X1,2));
DX2    =  DDX2(size(DX1,2)+1:2:size(DX1,2)+size(DX1,2));

% Using generationg kernel again to expand the signal.
[EX2,ETh2]      =  gaussexpand1d2(X2,Th2);
[EX2,ETh2]      =  gaussexpand1d2(EX2,ETh2);
[EDX2,EDTh2]    =  gaussexpand1d2(DX2,Th2);
[EDX2,EDTh2]    =  gaussexpand1d2(EDX2,EDTh2);

%% Build 3th. levels of the pyramid.

% Build 3 periods
TTTh2   = [Th2 Th2 Th2];
XXX2   =  [ X2  X2  X2];
DDDX2  =  [DX2 DX2 DX2];

% employ normal linear convolution
XX3    =  conv2(XXX2, Weight5, 'same');
DDX3   =  conv2(DDDX2,Weight5, 'same');

% cut down only a period to have the result of circular convolution
% subsample them with factor 2 to build the 3th.  level of the pyramid.

Th3    =  TTTh2(size(Th2,2)+1:2:size(Th2,2)+size(Th2,2));
X3    =  XX3(size(X2,2)+1:2:size(X2,2)+size(X2,2));
DX3    =  DDX3(size(DX2,2)+1:2:size(DX2,2)+size(DX2,2));

% Using generationg kernel again to expand the signal.
[EX3,ETh3]    =  gaussexpand1d2(X3,Th3);
[EX3,ETh3]    =  gaussexpand1d2(EX3,ETh3);
[EX3,ETh3]    =  gaussexpand1d2(EX3,ETh3);

[EDX3,EDTh3]  =  gaussexpand1d2(DX3,Th3);
[EDX3,EDTh3]  =  gaussexpand1d2(EDX3,EDTh3);
[EDX3,EDTh3]  =  gaussexpand1d2(EDX3,EDTh3);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%% Build 4th. levels of the pyramid.
%% Note: This level is subsampled from the level with 45 elements.
%%       45 can not be divided exactly by 2. That means if we subsample
%%       the sequence every two point, then due to the period of
%%       the angle the first and the last subsampling values have only
```

```
%%       the half sampling interval as the normal interval. But by
%%       the expanding process they are regarded as nomal interval. This
%%       leads an error. In order to avoid such error, we have tried
%%       not to use the
%%       last subsample by the first reexpanding. After the first
%%       expanding we then append the last subsample and expand the
%%       appended sequence further. But this method does not work.
%%       Now we  try to use pyramid with factor 3.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% Build 3 periods
TTTh3   = [Th3 Th3 Th3];
XXX3    =  [ X3  X3  X3];
DDDX3   =  [DX3 DX3 DX3];

% employ normal linear convolution
XX4     =  conv2(XXX3, Weight7, 'same');
DDX4    =  conv2(DDDX3,Weight7, 'same');

% cut down only a period to have the result of circular convolution
% subsample them with factor THREE to build the 4th.  level of the pyramid.

Th4     =  TTTh3(size(Th3,2)+1:3:size(Th3,2)+size(Th3,2));
X4     =  XX4(size(X3,2)+1:3:size(X3,2)+size(X3,2));
DX4     =  DDX4(size(DX3,2)+1:3:size(DX3,2)+size(DX3,2));

% Using generationg kernel again to expand the signal.
% The first expanding applies factor 3
[EX4,ETh4]     =  gaussexpand1d3(X4,Th4);
% Further expandings apply factor 2
[EX4,ETh4]     =  gaussexpand1d2(EX4,ETh4);
[EX4,ETh4]     =  gaussexpand1d2(EX4,ETh4);
[EX4,ETh4]     =  gaussexpand1d2(EX4,ETh4);

% The first expanding applies factor 3
[EDX4,EDTh4]   =  gaussexpand1d3(DX4,Th4);
% Further expandings apply factor 2
[EDX4,EDTh4]   =  gaussexpand1d2(EDX4,EDTh4);
[EDX4,EDTh4]   =  gaussexpand1d2(EDX4,EDTh4);
[EDX4,EDTh4]   =  gaussexpand1d2(EDX4,EDTh4);

%% Build 5th. levels of the pyramid applying further factor 3.

% Build 3 periods
TTTh4   = [Th4 Th4 Th4];
XXX4    =  [ X4  X4  X4];
DDDX4   =  [DX4 DX4 DX4];
```

```
% employ normal linear convolution
XX5    =  conv2(XXX4, Weight7, 'same');
DDX5   =  conv2(DDDX4,Weight7, 'same');

% cut down only a period to have the result of circular convolution
% subsample them with factor THREE to build the 5th.  level of the pyramid.

Th5    =  TTTh4(size(Th4,2)+1:3:size(Th4,2)+size(Th4,2));
X5    =  XX5(size(X4,2)+1:3:size(X4,2)+size(X4,2));
DX5    =  DDX5(size(DX4,2)+1:3:size(DX4,2)+size(DX4,2));

% Using generationg kernel again to expand the signal.
[EX5,ETh5]    =  gaussexpand1d3(X5,Th5);
[EX5,ETh5]    =  gaussexpand1d3(EX5,ETh5);

[EX5,ETh5]    =  gaussexpand1d2(EX5,ETh5);
[EX5,ETh5]    =  gaussexpand1d2(EX5,ETh5);
[EX5,ETh5]    =  gaussexpand1d2(EX5,ETh5);

[EDX5,EDTh5]    =  gaussexpand1d3(DX5,Th5);
[EDX5,EDTh5]    =  gaussexpand1d3(EDX5,EDTh5);

[EDX5,EDTh5]    =  gaussexpand1d2(EDX5,EDTh5);
[EDX5,EDTh5]    =  gaussexpand1d2(EDX5,EDTh5);
[EDX5,EDTh5]    =  gaussexpand1d2(EDX5,EDTh5);

figure; subplot(2,3,1),polar(Th,shift(energieableitung(X),90));
        subplot(2,3,2),polar(ETh1,shift(energieableitung(EX1),90));
        subplot(2,3,3),polar(ETh2,shift(energieableitung(EX2),90));
        subplot(2,3,4),polar(ETh3,shift(energieableitung(EX3),90));
        subplot(2,3,5),polar(ETh4,shift(energieableitung(EX4),90));
        subplot(2,3,6),polar(ETh5,shift(energieableitung(EX5),90));


function ableitung=energieableitung(X)
%%%
%%% berechnet die erste Ableitung einer winkelabhaengigen
%%% Energiefunktion und normiert diese
%%%

ramp = [-2:0.25:2];
Gauss = exp(-ramp.^2);
Derivat = -2.*ramp.*Gauss;
XXX = [X X X];
DDX = conv2( XXX, Derivat, 'same' );
DX = abs(DDX(size(X,2)+1:size(X,2)+size(X,2)));

minimum=min(min(DX));
maximum=max(max(DX));
```

```
ableitung=(DX-minimum)./(maximum-minimum);
return


function [Expand, Exth]   =  gaussexpand1d2(in,Th) ;
%%%
%%% Feb-11-98 This program expand the higher level of the Gauss pyramid
%%%           to lower level using the generating kernel.
%%%           The difference between the traditional expanding and our
%%%           expanding is that in the traditional expanding
%%%           if we regard the size of the input as N+1
%%%           then the size of the expanded output is 2N+1.
%%%           While in our method we expand the size of the input with
%%%           factor 2. namely N --> 2N
%%%
%%%           In order to avoid the boundary problem we simply repeat the
%%%           input three times: [in in in]
%%%
%%%
%%%           in must be a vector.

% Define the generating kernel again.
Weight = [1 4 6 4 1] ./16;

% Check if the input is a vector.
[m,n] = size(in);
if m ~= 1 & n ~= 1
  'Error!  The input must be a 1D signal.'
  return;
end
M = max(m,n);

if M ~= max(size(Th))
  'Error!  The input Angle must be the same as the signal.'
  return;
end

% To avoid exceeding the subordinate boundary.
% M2 is a integer with factor 2
% In order to cut sufficient subordinate, we set the minimum of
% the cutting part 4. This is only such an experience:
% To reconstruct the element i in lower level we need the element
% (i+2)/2 in higher level. Taking into our account that the
% subordinate  in Matlab beginns from not 0, but 1. we need further
% (i+4)/2 for higher matrix boundary and (i-4)/2 lower boundary.
% Thus we set the minimum 4.

M2= max(round(M/4)*2,4);

%producing the double densitive sampling angles
```

```
Exth   = zeros(1,2*M);
i      = 1:M;
Exth(2.*i-1) =  Th(i);
slope  = (Th(2)-Th(1));
Exth(2.*i) =  Th(i)+slope/2;

Expand = zeros(1,2*M);

rin = [in in in];
rexpand = zeros(1,6*M);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand the input using the expression (2) in \cite{burt83comm}.
% According to the constraint of equal contribution, each node will
% give the same total weight (here 1/2: (1+6+1)/16 or (8+8)/16)
% to the next higher level.  By the expanding we must use the
% reciprocal of this weight (here 2)
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
for i = 2+M2:2:6*M-M2;
 index1 =mod((i-2)./2, 6*M)+1;
 index3 =mod((i+0)./2, 6*M)+1;
 index5 =mod((i+2)./2, 6*M)+1;
 rexpand(i) = 2.*Weight(1).*rin(index1) ...
    + 2.*Weight(3).*rin(index3) ...
            + 2.*Weight(5).*rin(index5) ;
end
for  j = 1+M2:2:6*M-1-M2;
 index2 =mod((j-1)./2, 6*M)+1;
 index4 =mod((j+1)./2, 6*M)+1;
 rexpand(j) = 2.*Weight(2).*rin(index2) ...
            + 2.*Weight(4).*rin(index4) ;
end

Expand = rexpand(2*M+1:4*M);


function [Expand, Exth]   =  gaussexpand1d3(in,Th) ;
%%%
%%% Feb-11-98 This program expand the higher level of the Gauss pyramid
%%%           to lower level using the generating kernel.
%%%           The difference between the traditional expanding and our
%%%           expanding is that in the traditional expanding
%%%           if we regard the size of the input as N+1
%%%           then the size of the expanded output is 2N+1.
%%%           While in our method we expand the size of the input with
%%%           factor 3. namely N --> 3N
```

```
%%%           The corresponding reason is the periodic property of the
%%%           angles. We need an exact period to expand/subsampling,
%%%           while in the traditional pyramid method the boundary is
%%%           not  so important as in the angular pyramid.
%%%
%%%           In order to lose the periodic  problem of the angle we
%%%           simply repeat the  input three times: [in in in]
%%%
%%%
%%%
%%%           in must be a vector of angles.

% Define the generating kernel again.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% for pyramid with factor 3 we can not simply further apply the
% binomial mask with 6th. order, because the 6.th binomial mask:
% [ 1  6  15  20  15  6  1 ]
% [ a  b   c  d    c  b  a ]
% does not satisfy the constraint of equal contribution
% (\cite{burt83comm} p.533 line 15 from bottom),  which means
%  a+d+a = b+c  (a+b+c+d+c+b+a = 1)
% But we still want to have a mask similar to Gauss filter
% Thus, we define the following mask.
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
Weight7 = [3 22 66 82 66 22 3]./264;

% Check if the input is a vector.
[m,n] = size(in);
if m ~= 1 & n ~= 1
  'Error!  The input must be a 1D signal.'
  return;
end
M = max(m,n);

if M ~= max(size(Th))
  'Error!  The input Angle must be the same as the signal.'
  return;
end



% To avoid exceeding the subordinate boundary.
% M3 is a integer with factor 3
% In order to cut sufficient subordinate, we set the minimum of
% the cutting part 6.  This is only such an experience:
% To reconstruct the element i in lower level we need the element
% (i+3)/3 in higher level. Taking into our account that the
% subordinate  in Matlab beginns from not 0, but 1. we need further
```

```
% (i+6)/3 for higher matrix boundary and (i-6)/3 lower boundary.
% Thus we set the minimum 6.

M3= max(round(M/9)*3,6);

%producing the TRIPPLE densitive sampling angles
Exth   = zeros(1,3*M);
i      = 1:M;
Exth(3.*i-2) =  Th(i);
slope  = (Th(2)-Th(1));
Exth(3.*i-1) =  Th(i) + slope/3;
Exth(3.*i) =  Th(i)+2.*slope/3;

Expand = zeros(1,3*M);

rin = [in in in];
rexpand = zeros(1,9*M);

%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% Expand the input using the generalized expression (2) in
% \cite{burt83comm} (subsampling every three points) expressed as:
%
%               3
%  g_n(i) = 3  sum   w(m)*g_{n-1}((i-m)/3)
%              m=-3
%
% g_{n-1}((i-m)/3) is defined only when (i-m)/3 \in integer
%
% The constant 3 comes from the weight contribute as explained in
% gaussexpand1d2.m
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

for i = 3+M3:3:9*M-M3;
 index1 =mod((i-3)./3, 9*M)+1;
 index4 =mod((i+0)./3, 9*M)+1;
 index7 =mod((i+3)./3, 9*M)+1;
 rexpand(i) = 3.*Weight7(1).*rin(index7) ...
    + 3.*Weight7(4).*rin(index4) ...
            + 3.*Weight7(7).*rin(index1) ;
end

for i = 2+M3:3:9*M-1-M3;
 index2 =mod((i-2)./3, 9*M)+1;
 index5 =mod((i+1)./3, 9*M)+1;
 rexpand(i) = 3.*Weight7(3).*rin(index5) ...
            + 3.*Weight7(6).*rin(index2) ;
```

```
end

for i = 1+M3:3:9*M-2-M3;
 index3 =mod((i-1)./3, 9*M)+1;
 index6 =mod((i+2)./3, 9*M)+1;
 rexpand(i) = 3.*Weight7(2).*rin(index6) ...
            + 3.*Weight7(5).*rin(index3) ;
end

Expand = rexpand(3*M+1:6*M);
```

7.2 Steuerbare Filter

Um die Filterbasis für ein Bild anzuwenden, wird das Programm *EG4H4* benötigt. Dieses benutzt die restlichen Unterroutinen zur Erzeugung der Filterbasis.

```
function EG4H4=EG4G4(pic,xpos,ypos)
%%%
%%% EG4H4 = (pic*G4)^2+(pic*H4)^2
%%% liefert Engergiefunktion fuer Winkel [0..359]
%%% an der Stelle (xpos,ypos) des Bildes pic
%%%

% erzeuge Basisfilter
boxsize = 15;
for x=-boxsize:boxsize,
   for y=-boxsize:boxsize,
      kx=x/2;
      ky=y/2;
      fG4a (x+boxsize+1, y+boxsize+1) = G4a(kx,ky);
      fG4b (x+boxsize+1, y+boxsize+1) = G4b(kx,ky);
      fG4c (x+boxsize+1, y+boxsize+1) = G4c(kx,ky);
      fG4d (x+boxsize+1, y+boxsize+1) = G4d(kx,ky);
      fG4e (x+boxsize+1, y+boxsize+1) = G4e(kx,ky);

      fH4a (x+boxsize+1, y+boxsize+1) = H4a(kx,ky);
      fH4b (x+boxsize+1, y+boxsize+1) = H4b(kx,ky);
      fH4c (x+boxsize+1, y+boxsize+1) = H4c(kx,ky);
      fH4d (x+boxsize+1, y+boxsize+1) = H4d(kx,ky);
      fH4e (x+boxsize+1, y+boxsize+1) = H4e(kx,ky);
      fH4f (x+boxsize+1, y+boxsize+1) = H4f(kx,ky);
   end
end

% fuehre Faltungen mit Bild durch
falt = conv2 (pic, fG4a, 'same');
CpicG4a = falt(xpos,ypos);
falt = conv2 (pic, fG4b, 'same');
CpicG4b = falt(xpos,ypos);
falt = conv2 (pic, fG4c, 'same');
CpicG4c = falt(xpos,ypos);
falt = conv2 (pic, fG4d, 'same');
CpicG4d = falt(xpos,ypos);
falt = conv2 (pic, fG4e, 'same');
CpicG4e = falt(xpos,ypos);

falt = conv2 (pic, fH4a, 'same');
CpicH4a = falt(xpos,ypos);
falt = conv2 (pic, fH4b, 'same');
CpicH4b = falt(xpos,ypos);
falt = conv2 (pic, fH4c, 'same');
```

```
CpicH4c = falt(xpos,ypos);
falt = conv2 (pic, fH4d, 'same');
CpicH4d = falt(xpos,ypos);
falt = conv2 (pic, fH4e, 'same');
CpicH4e = falt(xpos,ypos);
falt = conv2 (pic, fH4f, 'same');
CpicH4f = falt(xpos,ypos);

% berechne Energiefunktion mit Hilfe der Interpolationsfunktionen
for theta=0:359,
   rad = theta * 2 * pi / 360;
   G4theta = cos(rad)^4 * CpicG4a - 4*cos(rad)^3*sin(rad) * CpicG4b + 6*cos(rad)^2*sin(rad)^2 * CpicG4c -
             4*cos(rad)*sin(rad)^3 * CpicG4d + sin(rad)^4 * CpicG4e;
   H4theta = cos(rad)^5 * CpicH4a - 5*cos(rad)^4*sin(rad) * CpicH4b + 10*cos(rad)^3*sin(rad)^2 * CpicH4c -
             10*cos(rad)^2*sin(rad)^3 * CpicH4d + 5*cos(rad)*sin(rad)^4 * CpicH4e - sin(rad)^5 * CpicH4f ;
   EG4H4(theta+1) = G4theta^2 + H4theta^2;
end
return

function G4a=G4a(x,y)
%%%
%%%  G4a = 1.246*(0.75-3*x^2+x^4)*exp(-(x^2+y^2))
%%%

G4a = 1.246*(0.75-3*x^2+x^4)*exp(-(x^2+y^2));
return

function G4b=G4b(x,y)
%%%
%%%  G4b = 1.246*(-1.5*x+x^3)*y*exp(-(x^2+y^2))
%%%

G4b = 1.246*(-1.5*x+x^3)*y*exp(-(x^2+y^2));
return

function G4c=G4c(x,y)
%%%
%%%  G4c = 1.246*(x^2-0.5)*(y^2-0.5)*exp(-(x^2+y^2))
%%%

G4c = 1.246*(x^2-0.5)*(y^2-0.5)*exp(-(x^2+y^2));
return

function G4d=G4d(x,y)
%%%
%%%  G4d = 1.246*(-1.5*y+y^3)*x*exp(-(x^2+y^2))
%%%

G4d = 1.246*(-1.5*y+y^3)*x*exp(-(x^2+y^2));
return
```

```
function G4e=G4e(x,y)
%%%
%%%  G4e = 1.246*(0.75-3*y^2+y^4)*exp(-(x^2+y^2))
%%%

G4e = 1.246*(0.75-3*y^2+y^4)*exp(-(x^2+y^2));
return

function H4a=H4a(x,y)
%%%
%%%  H4a = 0.3975*(7.189*x-7.501*x^3+x^5)*exp(-(x^2+y^2))
%%%

H4a = 0.3975*(7.189*x-7.501*x^3+x^5)*exp(-(x^2+y^2));
return

function H4b=H4b(x,y)
%%%
%%%  H4b = 0.3975*(1.438-4.501*x^2+x^4)*y*exp(-(x^2+y^2))
%%%

H4b = 0.3975*(1.438-4.501*x^2+x^4)*y*exp(-(x^2+y^2));
return

function H4c=H4c(x,y)
%%%
%%%  H4c = 0.3975*(x^3-2.225*x)*(y^2-0.6638)*exp(-(x^2+y^2))
%%%

H4c = 0.3975*(x^3-2.225*x)*(y^2-0.6638)*exp(-(x^2+y^2));
return

function H4d=H4d(x,y)
%%%
%%%  H4d = 0.3975*(y^3-2.225*y)*(x^2-0.6638)*exp(-(x^2+y^2))
%%%

H4d = 0.3975*(y^3-2.225*y)*(x^2-0.6638)*exp(-(x^2+y^2));
return

function H4e=H4e(x,y)
%%%
%%%  H4e = 0.3975*(1.438-4.501*y^2+y^4)*x*exp(-(x^2+y^2))
%%%

H4e = 0.3975*(1.438-4.501*y^2+y^4)*x*exp(-(x^2+y^2));
return

function H4f=H4f(x,y)
%%%
```

```
%%%  H4f = 0.3975*(7.189*y-7.501*y^3+y^5)*exp(-(x^2+y^2))
%%%

H4f = 0.3975*(7.189*y-7.501*y^3+y^5)*exp(-(x^2+y^2));
return
```

7.3 Wedgefilter

Die Anwendung der Wedgefilter erfolt durch das Programm *Ewedge*, die einzelnen Filter werden von den Unteroutinen *wedgeeven* und *wedgeodd* erzeugt.

```
function Ewedge=Ewedge(pic, xpos, ypos, N)
%%%
%%% Ewedge=Ewedge(pic, xpos, ypos, N)
%%% liefert Engergiefunktion fuer Winkel [0..359]
%%% an der Stelle (xpos,ypos) des Bildes pic
%%% N ist die Anzahl der Filterpaare
%%%

% berechne Faltungen aller Filter mit dem Bild
for i=1:N,
  wedgee=imresize(wedgeeven(i*2*pi/N,N), [7 7], 'bilinear');
  wedgeo=imresize( wedgeodd(i*2*pi/N,N), [7 7], 'bilinear');
  heven(:,:,i) = conv2(pic, wedgee, 'same');
  hodd(:,:,i)  = conv2(pic, wedgeo, 'same');
end;

% berechne die inverse Interpolations-Matrix
K = inv(kmatrix(N));

% berechne h-Vektor
for i=1:N,
  hevencur=heven(:,:,i);
  hoddcur=hodd(:,:,i);
  h(i*2-1) = hevencur(xpos,ypos);
  h(i*2  ) = hoddcur(xpos,ypos);
end

% berechne die Energiefunktion
for theta=0:359,
  rad=theta*2*pi/360;

% berechne ke-Vektor und ko-Vektor
  for i=1:N,
     ke(i*2-1) = cos(i*rad);
     ke(i*2  ) = sin(i*rad);
     ko(i*2-1) = -sin(i*rad);
     ko(i*2  ) = cos(i*rad);
  end;
  hevensum=ke*K*h';
  hoddsum=ko*K*h';
  Ewedge(theta+1)=hevensum^2+hoddsum^2;
end
return

function wedgeeven=wedgeeven(theta, N)
```

```
%%%
%%%  wedgeeven = wedgeeven (theta, N)
%%%  berechnet den geraden wedge-filter fuer den Winkel theta
%%%

boxsize=31;

% berechne Gewichte fuer Filterpaare
W=weights(180, N);

% berechne wedge-filter
for x=-boxsize:boxsize,
   for y=-boxsize:boxsize,
      phi=atan2(y,x);
      r = sqrt(x^2+y^2);
      z=he(phi+theta,W,N);
      f=radialfunction(r,2,3,30,3);
      wedgeeven(x+boxsize+1,y+boxsize+1)=f*z;
   end
end
return

function wedgeodd=wedgeodd(theta, N)
%%%
%%%  wedgeodd = wedgeodd (theta, N)
%%%  berechnet den ungeraden wedge-filter fuer den Winkel theta
%%%

boxsize=31;

% berechne Gewichte fuer Filterpaare
W=weights(180, N);

% berechne wedge-filter
for x=-boxsize:boxsize,
   for y=-boxsize:boxsize,
      phi=atan2(y,x);
      r = sqrt(x^2+y^2);
      z=ho(phi+theta,W,N);
      f=radialfunction(r,2,3,30,3);
      wedgeodd(x+boxsize+1,y+boxsize+1)=f*z;
   end
end
return

function radial=radialfunction(r,r1,d1,r2,d2)
%%%
%%% radial=radialfunction(r1,d1,r2,d2)
%%% berechnet die Radial-Funktion
```

```
%%% r, r1, r2 sind Radien
%%% d1, d2 sind Rand-Faktoren
%%%

radial=0;

if (r < r1)
  radial=0;
end;
if ((r1 <= r) & (r < r1+d1))
  radial=0.5*(1-cos(pi*(r-r1)/d1));
end;
if ((r1+d1 <= r) & (r <= r2-d2))
  radial=1;
end;
if ((r2-d2 <= r) & (r < r2))
  radial=0.5*(1+cos(pi*(r+d2-r2)/d2));
end;
if (r2 <= r)
  radial=0;
end
return

function kmatrix=kmatrix(N)
%%%
%%% kmatrix=kmatrix(N)
%%% berechnet Interpolationsmatrix fuer wedge-filter
%%% N Anzahl der Filterpaare
%%%

for i=1:N,
   for j=1:N,
      rad = 2*pi*j/N;
      kmatrix(2*i-1,2*j-1) = cos(i*rad);
      kmatrix(2*i  ,2*j-1) = sin(i*rad);
      kmatrix(2*i  ,2*j  ) = cos(i*rad);
      kmatrix(2*i-1,2*j  ) = -sin(i*rad);
   end
end
return

function w=weights(M,N)
%%%
%%%  weights(M,N)= eig(C'L'LC+S'L'L'S)
%%%

C=cosbas(M,N);
S=sinbas(M,N);
L=lambda(M);
```

```
w=eig(C'*L'*L*C+S'*L'*L*S);
return

function L=lambda(M)
%%% Berechnung der Diagonalmatrix lambda(phi)=phi
%%%    M Zeilen
%%%    M Spalten

for i=1:M,
      L(i,i)= 2*(i-1)*pi/M;
end
return

function S=sinbas(M,N)
%%% Berechnung der sinusoidalen Basis
%%%    M Zeilen
%%%    N Spalten

for i=1:M,
   for j=1:N,
      S(i,j)=(sin(2*pi*i*j/M));
   end
end
return

function C=cosbas(M,N)
%%% Berechnung der cosinusoidalen Basis
%%%    M Zeilen
%%%    N Spalten

for i=1:M,
   for j=1:N,
      C(i,j)=(cos(2*pi*i*j/M));
   end
end
return
```

7.4 Strukturtensor

Für die Berechnung des Strukturtensors wird lediglich ein Programm benötigt.

```
function strukturtensor=strukturtensor(pic, xpos, ypos, level)
%%%
%%% strukturtensor=strukturtensor(pic, xpos, ypos)
%%% berechnet den Strukturtensor an der Stelle (xpos,ypos)
%%% Ergebnis ist eine 2x2-Matrix
%%%

% berechne die Richtungsableitungen mit optimiertem Sobeloperator
dx=[0.08382441 0.33235118 0 -0.33235118 -0.08382441];
gy=[0.02342211 0.2415544 0.47004698 0.2415544 0.02342211]';
dx=gy*dx;
dy=-dx';

sox=conv2(pic, dx, 'same');
soy=conv2(pic, dy, 'same');

% erstelle isotrope Glaettungsfilter
B = binomial(level);

% berechne Strukturtensor-Elemente
Ixx = conv2(sox.*sox, B, 'same');
Iyy = conv2(soy.*soy, B, 'same');
Ixy = conv2(sox.*soy, B, 'same');

strukturtensor = [Ixx(xpos,ypos) Ixy(xpos,ypos); Ixy(xpos,ypos) Iyy(xpos,ypos)];

for i=1:size(pic,1),
  for j=1:size(pic,2),
% Orientierungsbild
    orientbild(i,j)=(atan2(2*Ixy(i,j), Ixx(i,j) - Iyy(i,j))+pi)*255/(2*pi);
% Kohaerenzmass gemaess Jaehne
%     orientbild(i,j)=((Iyy(i,j)-Ixx(i,j))^2 + 4 * (Ixy(i,j)^2))/((Ixx(i,j)+Iyy(i,j))^2)*255;;
  end
end

figure;
imagesc (orientbild), axis image;
set_gray256;
return
```

7.5 Phasen Kongruenz

Der Sourcecode ist unter folgender Adresse aus dem Internet bezogen worden: *http://www.cs.uwa.edu.au/ pk/Research/phasecong.m.* Es sind Anpassungen an die vorhandene Problemstellung vorgenommen worden.

```
%
% PHASECONG - function for computing phase congruency on an image
%
% Usage: [phaseCongruency orientation] = phasecong(image)
%
% This function calculates the PC_2 measure of phase congruency.
% For maximum speed the input image should be square and have a
% size that is a power of 2, but the code will operate on images
% of arbitrary size.
%
%
% Return values:
%                 PhaseCongruency   - phase congruency image
%                                     (values between 0 and 1)
%                 orientation       - orientation image.
%                                     (orientation in which local energy
%                                      is a maximum, radians)
%
% Peter Kovesi, "Image Features From Phase Congruency".
% Videre: A Journal of Computer Vision Research. MIT Press.
% Volume 1, Number 3, Summer 1999
% http://mitpress.mit.edu/e-journals/Videre/001/v13.html
%
% Author: Peter Kovesi   pk@cs.uwa.edu.au
% Department of Computer Science
% The University of Western Australia
%
% Original Version written April 1996
% Noise compensation corrected. August 1998
% Noise compensation corrected. October 1998  - Again!!!
% Modified to operate on non-square images of arbitrary size. September 1999

function[phaseCongruency,orientation]=phasecong(image)

sze = size(image);

nscale          = 4;     % Number of wavelet scales.
norient         = 6;     % Number of filter orientations.
minWaveLength   = 3;     % Wavelength of smallest scale filter.
mult            = 2;     % Scaling factor between successive filters.
sigmaOnf        = 0.55;  % Ratio of the standard deviation of the Gaussian
                         % describing the log Gabor filter's transfer function
                         % in the frequency domain to the filter center frequency.
dThetaOnSigma   = 1.2;   % Ratio of angular interval between filter orientations
```

```
                        % and the standard deviation of the angular Gaussian
                        % function used to construct filters in the freq. plane.
k               = 2.0;  % No of standard deviations of the noise energy beyond the
                        % mean at which we set the noise threshold point.
                        % standard deviation to its maximum effect on Energy.
cutOff          = 0.4;  % The fractional measure of frequency spread below which
                        % phase congruency values get penalized.
g               = 10;   % Controls the sharpness of the transition in the sigmoid
                        % function used to weight phase congruency for frequency
                        % spread.
epsilon         = .0001; % Used to prevent division by zero.


thetaSigma = pi/norient/dThetaOnSigma;  % Calculate the standard deviation of the
                                        % angular Gaussian function used to
                                        % construct filters in the freq. plane.

imagefft = fft2(image);                 % Fourier transform of image
sze = size(imagefft);
rows = sze(1);
cols = sze(2);
zero = zeros(sze);

totalEnergy = zero;                     % Matrix for accumulating weighted phase
                                        % congruency values (energy).
totalSumAn  = zero;                     % Matrix for accumulating filter response
                                        % amplitude values.
orientation = zero;                     % Matrix storing orientation with greatest
                                        % energy for each pixel.
estMeanE2n = [];

% Pre-compute some stuff to speed up filter construction

x = ones(rows,1) * (-cols/2 : (cols/2 - 1))/(cols/2);
y = (-rows/2 : (rows/2 - 1))' * ones(1,cols)/(rows/2);
radius = sqrt(x.^2 + y.^2);       % Matrix values contain *normalised* radius from centre.
radius(round(rows/2+1),round(cols/2+1)) = 1; % Get rid of the 0 radius value in the middle
                                             % so that taking the log of the radius will
                                             % not cause trouble.
theta = atan2(-y,x);              % Matrix values contain polar angle.
                                  % (note -ve y is used to give +ve anti-clockwise angles)
clear x; clear y;      % save a little memory

% The main loop...

for o = 1:norient,                   % For each orientation.
  disp(['Processing orientation ' num2str(o)]);
  angl = (o-1)*pi/norient;           % Calculate filter angle.
  wavelength = minWaveLength;        % Initialize filter wavelength.
```

```
sumE_ThisOrient   = zero;          % Initialize accumulator matrices.
sumO_ThisOrient   = zero;
sumAn_ThisOrient  = zero;
Energy_ThisOrient = zero;
EOArray = [];          % Array of complex convolution images - one for each scale.
ifftFilterArray = [];  % Array of inverse FFTs of filters

% Pre-compute filter data specific to this orientation
% For each point in the filter matrix calculate the angular distance from the
% specified filter orientation.  To overcome the angular wrap-around problem
% sine difference and cosine difference values are first computed and then
% the atan2 function is used to determine angular distance.

ds = sin(theta) * cos(angl) - cos(theta) * sin(angl); % Difference in sine.
dc = cos(theta) * cos(angl) + sin(theta) * sin(angl); % Difference in cosine.
dtheta = abs(atan2(ds,dc));                           % Absolute angular distance.
spread = exp((-dtheta.^2) / (2 * thetaSigma^2));      % Calculate the angular filter component.

for s = 1:nscale,                  % For each scale.

  % Construct the filter - first calculate the radial filter component.
  fo = 1.0/wavelength;                  % Centre frequency of filter.
  rfo = fo/0.5;                         % Normalised radius from centre of frequency plane
                                        % corresponding to fo.
  logGabor = exp((-(log(radius/rfo)).^2) / (2 * log(sigmaOnf)^2));
  logGabor(round(rows/2+1),round(cols/2+1)) = 0; % Set the value at the center of the filter
                                                 % back to zero (undo the radius fudge).

  filter = logGabor .* spread;          % Multiply by the angular spread to get the filter.
  filter = fftshift(filter);            % Swap quadrants to move zero frequency
                                        % to the corners.

  ifftFilt = real(ifft2(filter))*sqrt(rows*cols);  % Note rescaling to match power
  ifftFilterArray = [ifftFilterArray ifftFilt];    % record ifft2 of filter

  % Convolve image with even and odd filters returning the result in EO
  EOfft = imagefft .* filter;           % Do the convolution.
  EO = ifft2(EOfft);                    % Back transform.

  EOArray = [EOArray, EO];              % Record convolution result
  An = abs(EO);                         % Amplitude of even & odd filter response.

  sumAn_ThisOrient = sumAn_ThisOrient + An;     % Sum of amplitude responses.
  sumE_ThisOrient = sumE_ThisOrient + real(EO); % Sum of even filter convolution results.
  sumO_ThisOrient = sumO_ThisOrient + imag(EO); % Sum of odd filter convolution results.

  if s == 1                             % Record the maximum An over all scales
    maxAn = An;
  else
```

```
    maxAn = max(maxAn, An);
  end

  if s==1
    EM_n = sum(sum(filter.^2));          % Record mean squared filter value at smallest
  end                                    % scale. This is used for noise estimation.

  wavelength = wavelength * mult;        % Finally calculate Wavelength of next filter
end                                      % ... and process the next scale

% Get weighted mean filter response vector, this gives the weighted mean phase angle.
XEnergy = sqrt(sumE_ThisOrient.^2 + sumO_ThisOrient.^2) + epsilon;
MeanE = sumE_ThisOrient ./ XEnergy;
MeanO = sumO_ThisOrient ./ XEnergy;

% Now calculate An(cos(phase_deviation) - | sin(phase_deviation)) | by using
% dot and cross products between the weighted mean filter response vector and
% the individual filter response vectors at each scale.  This quantity is
% phase congruency multiplied by An, which we call energy.

for s = 1:nscale,
  Energy_ThisOrient = Energy_ThisOrient ...
      + real(submat(EOArray,s,cols)).*MeanE + imag(submat(EOArray,s,cols)).*MeanO ...
   -abs(real(submat(EOArray,s,cols)).*MeanO - imag(submat(EOArray,s,cols)).*MeanE );
end

clear XEnergy; clear MeanE; clear MeanO;    % save a little more memory

% Compensate for noise
% We estimate the noise power from the energy squared response at the smallest scale.
% If the noise is Gaussian the energy squared will have a Chi-squared 2DOF pdf.
% We calculate the median energy squared response as this is a robust statistic.
% From this we estimate the mean.
% The estimate of noise power is obtained by dividing the mean squared energy value
% by the mean squared filter value

medianE2n = median(reshape(abs(submat(EOArray,1,cols)).^2,1,rows*cols));
meanE2n = -medianE2n/log(0.5);
estMeanE2n = [estMeanE2n meanE2n];

noisePower = meanE2n/EM_n;                       % Estimate of noise power.

% Now estimate the total energy^2 due to noise
% Estimate for sum(An^2) + sum(Ai.*Aj.*(cphi.*cphj + sphi.*sphj))

EstSumAn2 = zero;
for s = 1:nscale
  EstSumAn2 = EstSumAn2+submat(ifftFilterArray,s,cols).^2;
end
```

```
EstSumAiAj = zero;
for si = 1:(nscale-1)
  for sj = (si+1):nscale
    EstSumAiAj = EstSumAiAj + submat(ifftFilterArray,si,cols).*submat(ifftFilterArray,sj,cols);
  end
end

EstNoiseEnergy2 = 2*noisePower*sum(sum(EstSumAn2)) + 4*noisePower*sum(sum(EstSumAiAj));

tau = sqrt(EstNoiseEnergy2/2);                    % Rayleigh parameter
EstNoiseEnergy = tau*sqrt(pi/2);                  % Expected value of noise energy
EstNoiseEnergySigma = sqrt( (2-pi/2)*tau^2 );

T =  EstNoiseEnergy + k*EstNoiseEnergySigma;      % Noise threshold

% The estimated noise effect calculated above is only valid for the PC_1 measure.
% The PC_2 measure does not lend itself readily to the same analysis.  However
% empirically it seems that the noise effect is overestimated roughly by a factor
% of 1.7 for the filter parameters used here.
T = T/1.7;        % Empirical rescaling of the estimated noise effect to
                  % suit the PC_2 phase congruency measure
Energy_ThisOrient = max(Energy_ThisOrient - T, zero);  % Apply noise threshold

% Form weighting that penalizes frequency distributions that are particularly
% narrow.
% Calculate fractional 'width' of the frequencies present by taking
% the sum of the filter response amplitudes and dividing by the maximum
% amplitude at each point on the image.
width = sumAn_ThisOrient ./ (maxAn + epsilon) / nscale;

% Now calculate the sigmoidal weighting function for this orientation.
weight = ones(sze) ./ (1 + exp( (cutOff - width)*g));

% Apply weighting
Energy_ThisOrient =   weight.*Energy_ThisOrient;

% Update accumulator matrix for sumAn and totalEnergy
totalSumAn  = totalSumAn + sumAn_ThisOrient;
totalEnergy = totalEnergy + Energy_ThisOrient;

% Update orientation matrix by finding image points where the energy in this
% orientation is greater than in any previous orientation (the change matrix)
% and then replacing these elements in the orientation matrix with the
% current orientation number.
if(o == 1),
  maxEnergy = Energy_ThisOrient;
else
  change = Energy_ThisOrient > maxEnergy;
```

```
    orientation = (o - 1).*change + orientation.*(~change);
    maxEnergy = max(maxEnergy, Energy_ThisOrient);
  end
end  % For each orientation


disp('Mean Energy squared values recorded with smallest scale filter at each orientation');
disp(estMeanE2n);

% Display results
imagesc(totalEnergy), axis image, title('total energy');
disp('Hit any key to continue '); pause
imagesc(totalSumAn), axis image, title('total sumAn');
disp('Hit any key to continue '); pause

% Normalize totalEnergy by the totalSumAn to obtain phase congruency
phaseCongruency = totalEnergy ./ (totalSumAn + epsilon);
imagesc(phaseCongruency), axis image, title('phase congruency');

% Convert orientation matrix values to degrees
orientation = orientation * (180 / norient);

%
% SUBMAT
%
% Function to extract the i'th sub-matrix 'cols' wide from a large
% matrix composed of several matricies.  The large matrix is used in
% lieu of an array of matricies

function a = submat(big,i,cols)
a = big(:,((i-1)*cols+1):(i*cols));
```

7.6 Monogenes Signal

Die Berechnung des Monogenen Signals erfolgt im Rahmen derr Klasse *vectorarray*. Die hier angegebenen Bestandteile der Klasse sind der Konstruktor *vectorarray* sowie die Methode *orient*, die mit der Hilfe des Konstrukturs die lokale Orientierung berechnet.

```
function va=vectorarray(arg1,octave,bandwidth)

% Various constructors for vectorarray class
% vectorarray class (c) by Michael Felsberg
% rudimentary version 10/2000
% attention: octave-parameter has changed: instead of 0..log_2(size)
% use now log_2(size)..0 ! 1 means Nyquist-frequence, 2 means half of the
% Nyquist-fr. and so on.

if (nargin==0)                  % empty data
    va.r=[];
    va.i=[];
    va.j=[];
    va=class(va,'vectorarray');
elseif isa(arg1,'vectorarray')  % copy constructor
    va=arg1;
elseif (nargin>1)

% This constructor calculates the spherical quadrature filter response of
% an image.
% The parameter octave corresponds to the scale of the filter, i.e.
% the higher the octave the higher the center frequency.
% The (optional) parameter bandwidth specifies the relative bandwidth of
% the filter. Default is 0.55 which means a two octave bandwidth.
% 0.75 corresponds to a one octave bandwidth.

    if (nargin==2)
        bandwidth=0.55;         % default bandwidth
    end
    sx=size(arg1,2);            % image width
    sy=size(arg1,1);            % image height
    sxu=ceil(sx/2)-1;           % biggest x-coordinate after fftshift
    sxl=ceil(-sx/2);            % smallest x-coordinate after fftshift
    syu=ceil(sy/2)-1;           % biggest y-coo.
    syl=ceil(-sy/2);            % smallest y-coo.
    [u,v]=meshgrid(sxl:sxu,syl:syu);
                                % frequency coordinates
    u=u./sx;                    % normalized frequency coordinates
    v=v./sy;    % s.a.
    r=sqrt(u.^2+v.^2);          % length of frequency vector
    r(r==0)=1;                  % workaround singularity part I
    BANDPASS=exp(-(log(r/2^-octave)).^2/(2*log(bandwidth).^2));
                                % radial bandpass (lognormal)
    BANDPASS(u==0 & v==0)=0;    % workaround singularity part II
```

```
     IMAGE=fftshift(fft2(arg1));   % spectrum of image
     IMAGE=IMAGE.*BANDPASS;        % bandpass filtered spectrum
     u(:,1)=0;                     % ideal odd symmetry
     %v(1,:)=0;                    % not necessary because of imag in line (*)
     tmp=ifft2(ifftshift(IMAGE.*(1+u./r)));
                                   % temporary filter result for r and i part
     va.r=real(tmp);               % even filter response
     va.i=-imag(tmp);              % 1st odd filter response
     va.j=-imag(ifft2(ifftshift(IMAGE.*v./r)));
                                   % 2nd odd filter response (*)
     va=class(va,'vectorarray');
else                               % constructor for 3D array (compatibility)
     va.r=arg1(:,:,1);
     va.i=arg1(:,:,2);
     va.j=arg1(:,:,3);
     va=class(va,'vectorarray');
end

function theta=orient(va)

% returns matrix of local orientation

va=vectorarray(va);
theta=atan(va.j./va.i);
theta=theta+(theta<0)*pi;
```

Abbildungsverzeichnis

Literaturverzeichnis

[1] J. Babaud, A.P. Witkin, M. Baudin, and R.O. Duda. Uniqueness of the gaussian kernel for scale-space filtering. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 8, pages 26–33, 1986.

[2] C. Bernard. Einführung in die Wavelettheorie. http://www.cmap.polytecnique.fr/~bernard/.

[3] F. Brackx and F. Sommen. Clifford-hermite wavelets in euclidean space. In *The Journal of Fourier Analysis and Applications*, volume 6, Issue 3, pages 299–310. Birkhauser Boston, 2000.

[4] P.J. Burt. The pyramid as a structure for efficient computation. In A. Rosenfeld, editor, *Multiresolution image processing and analysis*, volume 12. Berlin: Springer, 1984.

[5] P.J. Burt and E.H. Adelson. The laplacian pyramid as a compact image code. In *IEEE Trans. on Communications*, volume 31, pages 532–540, 1983.

[6] J. Canny. A computational approach to edge detection. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume PAMI-8, No. 6, pages 679–698, 1986.

[7] M. Felsberg and G. Sommer. A new extension of linear signal processing for estimating local properties and detecting features. In G. Sommer, N. Krüger, and Ch. Perwass, editors, *22. Symposium für Mustererkennung, DAGM 2000, Kiel*, pages 195–202. Springer-Verlag, 2000.

[8] M. Felsberg and G. Sommer. Structure multivector for local analysis of images. Technical report, Christian-Albrechts-Universität zu Kiel, Institut für Informatik und Praktische Mathematik, Februar 2000.

[9] W. Förstner. A framework for low level feature extraction. In J.O. Eklundh, editor, *European Conference on Computer Vision*, volume 2, pages 383–394. Stockholm, Sweden, May 2-6, LNCS, Vol. 801, Springer–Verlag, 1994.

[10] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 13, No. 9, pages 891–906, 1991.

[11] H. Groemer. *Spherical Harmonics.* Cambridge University Press, 1996.

[12] M. Hahn and N. Krüger. Junction detection and semantic interpretation using hough lines. In *2nd Int. ICSC Symposium on Engineering of Intelligent Systems (EIS'2000), University of Paisley, Scotland*, pages 109–115, 2000.

[13] E. Fourgeau J. Morlet, G. Arens and D. Giard. Wave propagation and sampling theory. In *Geophysics*, volume 47(2), pages 222–236, 1982.

[14] B. Jähne. *Digitale Bildverarbeitung.* Springer–Verlag, 1997.

[15] P. Kovesi. Image features from phase congruency. In *Videre: Journal of Computer Vision Research*, volume Volume 1, Number 3. The MIT Press, 1999.

[16] M. Michaelis and G. Sommer. Junction classification by multiple orientation detection. In J.-O. Eklundh, editor, *Proc. Third European Conference on Computer Vision*, pages 101–108. Stockholm, LNCS, Vol. 800, Springer–Verlag, 1994.

[17] M. Michaelis and G. Sommer. A Lie group approach to steerable filters. *Pattern Recognition Letters*, 16:1165–1174, 1995.

[18] M. C. Morrone and R.A. Owens. Feature detection from local energy. In *Pattern Recognition Letters*, volume Volume 6, pages 303–313, 1987.

[19] P. Perona. Deformable kernels for early vision. In *IEEE Trans. Pattern Analysis and Machine Intelligence*, volume 17, No. 5, pages 488–499, 1995.

[20] K. Rohr. On the precision in estimating the location of edges and corners. In *Journal of Mathematical Imaging and Vision*, pages 7–22. Kluwer Academic Publishers, 1997.

[21] E.P. Simoncelli and H. Farid. Steerable wedge filters for local orientation analysis. In *IEEE Trans. on Image Processing*, volume 5, No. 9, pages 1377–1382, 1996.

[22] E.P. Simoncelli, W. T. Freeman, E.H. Adelson, and D.J. Heeger. Shiftable multiscale transforms. In *IEEE Trans. on Information Theory*, volume 38, No. 2, pages 587–607, 1992.

[23] P.C. Teo and Y. Hel-Or. A computational group-theoretic approach to steerable functions. Technical report, Standford University, Department of Computer Science, April 1996.

[24] S. Venkatesh and R.A. Owens. An energy feature detection scheme. In *The International Conference on Image Processing*, pages 553–557. Singapore, 1989.

[25] W. Yu, K. Daniilidis, and G. Sommer. Junction characterization using polar pyramid. In P. Levi, R.-J. Ahlers, F. May, and M. Schanz, editors, *Mustererkennung 1998, Tagungsband 20. DAGM–Symposium*, pages 69–76. DAGM'98, Springer, 1998.

[26] W. Yu, K. Daniilidis, and G. Sommer. Low-cost junction characterization using polar averaging filters. In *Int. Conf. Image Processing, ICIP'98*, volume 3, pages 41–44, Chicago, Illinois, 1998. IEEE Signal Processing Society.

[27] W. Yu, K. Daniilidis, and G. Sommer. Rotated wedge averaging method for junction characterization. In *Proc. Computer Society Conference on Computer Vision and Pattern Recognition 1998*, pages 390–395. CVPR'98, Santa Barbara, CA, June 23-25, 1998.

Erklärung

Hiermit versichere ich, dass ich die vorliegende Arbeit, selbstständig verfasst und ausschließlich die angebenenen Hilfsmittel und Quellen verwendet habe.

Neumünster, den 11. Juli 2001