UNIVERSITÄTSKLINIKUM DER RHEINISCH-WESTFÄLISCHEN TECHNISCHEN HOCHSCHULE AACHEN
INSTITUT FÜR MEDIZINISCHE INFORMATIK
GESCHÄFTSFÜHRENDER DIREKTOR: UNIVERSITÄTS-PROFESSOR DR. DR. KLAUS SPITZER

Studienarbeit

Texturanalyse von Farbbildern mit Gaborfiltern

16. Juni 1999

Daniel Keysers
Matr.-Nr. 205354

Betreuer:
Dipl.-Inform. Christoph Palm

Inhaltsverzeichnis

1 Einleitung

Texturen können bei der Analyse von Bildmaterial sowohl bei der Segmentierung von Bildteilen als auch bei der Klassifizierung von Bildern eine wesentliche Rolle spielen. Zur (automatischen) Unterscheidung von Texturen wird in jedem Fall eine quantitative Beschreibung benötigt. Dabei existieren im Wesentlichen zwei Modellierungsarten. Einerseits strukturelle Modelle, die die räumliche Organisation von einfachen Strukturen in der Textur betrachten. Andererseits werden statistische Modelle wie Co-Occurrence-Analysen oder texturbeschreibende Verteilungen benutzt. Ferner wird bei der Unterscheidung von Texturen ein modellabhängiges Abstandsmaß eingesetzt, durch das die quantifizierten Größen verglichen werden können.

Bei der Segmentierung der morphologischen Strukturen in medizinischen Bildern spielt die Texturanalyse eine wichtige Rolle [11]. Diese Studienarbeit gliedert sich ein in ein Projekt zur Früherkennung von Tumoren auf Stimmlippen bei Rezidivpatienten. Ein solcher Tumor geht einher mit Veränderungen der sichtbaren Textur der Stimmlippen in laryngoskopischen Aufnahmen. Die Textur ändert sich allerdings auch schon durch die Behandlung des Tumors mit einem Laser. Es könnte jedoch möglich sein, in Zukunft durch eine Texturanalyse der Bilder solche krankhaften Veränderungen zu erkennen, sie grob zu lokalisieren und dadurch die Häufigkeit von Gewebeentnahmen zu reduzieren. Die hier betrachtete Methode dient jedoch nicht nur der Analyse von laryngoskopischen Bildern, sondern bezieht sich auf Farbbilder im Allgemeinen.

In dieser Studienarbeit wird die Implementierung eines Verfahrens zur Analyse von Texturen mit Gaborfiltern vorgestellt. Die Gabortransformation ist ein Spezialfall der Fensterfouriertransformation und daher den strukturellen Methoden zuzuordnen. Da in der medizinischen Bildverarbeitung verstärkt Multispektralbilder ausgewertet werden und oft reine Farbtexturen auftreten, die in der Helligkeitsdarstellung nicht erkennbar sind, wird hier besonders die Anwendung auf Farbtexturen betont. Dies stellt eine Erweiterung bekannter Verfahren dar [3, 8, 5, 7] und lässt sich durch Transformation der Buntheitsinformation in eine komplexwertige Darstellung elegant verwirklichen.

Texturen lassen sich als Helligkeitsmuster modellieren, die eine Anzahl von Ortsfrequenzen mit bestimmten Orientierungen aufweisen. Dabei unterscheiden sich verschiedene Texturen wesentlich in ihren charakteristischen Frequenzen. Bei der Betrachtung von Farbtexturen ist es sinnvoll, auch die Informationen, die die farbigen Anteile bieten, zu analysieren. Es ist jedoch anschaulich klar, dass die Bedeutung der Buntheit für eine Textur nicht in allen Fällen die gleiche ist. Man kann sich leicht Texturen vorstellen, die allein aufgrund der Helligkeitsverteilung beziehungsweise der Farbverteilung zu charakterisieren sind.

Die Benutzung von Gaborfiltern erlaubt es, aus einem Bild mit relativ geringem

Rechenaufwand getrennte Informationen über die Anwesenheit von festzulegenden Frequenzen und Orientierungen in der Umgebung eines Pixels zu extrahieren. Sie können als Beschreibungsvektoren für die umgebende Textur verwendet werden. Die berechneten Werte besitzen Amplitude und Phase, wobei letztere dazu benutzt werden kann, unstetige Übergänge zwischen gleichartigen Texturen zu entdecken. Mit dieser in Kanäle aufgeteilten Beschreibung kann man Texturen, die in einem Bild vorkommen, segmentieren oder klassifizieren. Gaborfilter haben hierfür geeignete Eigenschaften, da sie einstellbare Orientierungs- und Frequenzbandbreiten besitzen und die optimale gleichzeitige Auflösung in Orts- und Frequenzraum erreichen. Das Ergebnis der Gabortransformation besitzt sowohl Orts- als auch Frequenzinformation, man spricht auch vom Orts-/Frequenzraum. So ergeben sich zwei mögliche Sichtweisen: Die Region um ein beliebiges, aber festes Pixel wird durch die Antworten einer Menge von um dieses Pixel örtlich zentrierten Gaborfiltern verschiedener Frequenzen und Orientierungen beschrieben. Andererseits kann man die sich ergebenden Teilbilder zu einem beliebigen, aber festen Gaborfilter, als komplex moduliertes Bild betrachten, dessen Amplitude die räumliche Verteilung der festen Frequenz und Orientierung im Bild beschreibt.

In den folgenden Abschnitten werden zunächst einige theoretische Grundlagen der benutzten Verfahren vorgestellt. Anschließend wird die Implementierung beschrieben, wobei detailliert auf die einzelnen Bestandteile eingegangen wird. Ferner werden zwei Versuche zur Anwendung des Verfahrens vorgestellt: Die Segmentierung eines Farbbildes in verschiedene Bereiche anhand von Texturen sowie die Klassifizierung farbiger Aufnahmen von Baumrinden.

2 Theoretische Grundlagen

In diesem Abschnitt sollen die theoretischen Grundlagen des benutzten Verfahrens beschrieben werden. Dazu wird zunächst der verwendete Farbraum vorgestellt und dann auf Fourier- und Gabortransformation eingegangen.

2.1 HSV-Farbraum

In der vorliegenden Studienarbeit wurden die zu untersuchenden Farbbilder aus der sonst vielfach üblichen RGB-Darstellung mit den (Farb-)Kanälen Rot, Grün und Blau in den HSV-Farbraum mit den Kanälen Hue (Farbton), Saturation (Sättigung) und Value (Dunkelstufe, Helligkeit) transformiert. Die Bezeichnungen der Kanäle sind dabei eher intuitiv, als dies bei einer Darstellung durch zu mischende Grundfarben - wie zum Beispiel im RGB-Farbraum - der Fall ist.

Die Buntheit eines Pixels lässt sich in diesem Farbraum gut in einem Polarkoordinatensystem darstellen, wobei die Sättigung als Betrag und der Farbton als Winkel bzw. Phase interpretiert wird (siehe Abbildung 1). Die Interpretation als Winkel hat die angenehme Eigenschaft, dass es keine „Sprünge“ in der Farbtonskala bei kleinen Farbänderungen gibt (wie es bei einer direkten Benutzung von $h \in [0°, 360°)$ (Hue) der Fall wäre). Dabei liegen Komplementärfarben zum Beispiel maximal entfernt (um 180° versetzt). Diese Darstellung erlaubt die Interpretation als komplexe Zahl. Die Buntheit b in einem Punkt ergibt sich dann aus der Sättigung s und dem Farbton h zu [6]:

$$\begin{aligned} b(x,y) &= s(x,y) \cdot e^{i \cdot h(x,y)} \\ &= s(x,y) \cdot \cos\left[h(x,y)\right] + i \cdot s(x,y) \cdot \sin\left[h(x,y)\right] \end{aligned}$$

Die Buntheitsinformation lässt sich als mit der Sättigung gewichteter Farbton interpretieren. Dies führt dazu, dass Farbsprünge mit gering gesättigten Farben weniger ins Gewicht fallen. Man erhält so aus einem Farbbild ein komplexwertiges Buntheitsbild b und ein reellwertiges Helligkeitsbild v. Diese lassen sich selbstverständlich wieder zurücktransformieren. Das Buntheitsbild enthält nun nur helligkeitsunabhängige Merkmale und ist aufgrund der komplexen Darstellung gut geeignet, mittels Fouriertransformation manipuliert zu werden. Die Fouriertransformierte eines solchen Bildes ist dann wegen der komplexen Eingabe i.A. nicht konjugiertsymmetrisch, wie es bei reellwertigen Bildern der Fall ist [9].

2.2 Fourier- und Fensterfouriertransformation

Die Fouriertransformation (FT) zerlegt ein Signal f in Sinus- und Cosinusschwingungen, aus denen sich das Signal zusammensetzt. Der Graph $F(\omega)$ des fouriertransformierten Signals wird als *Frequenzspektrum* bezeichnet. Das Signal liegt

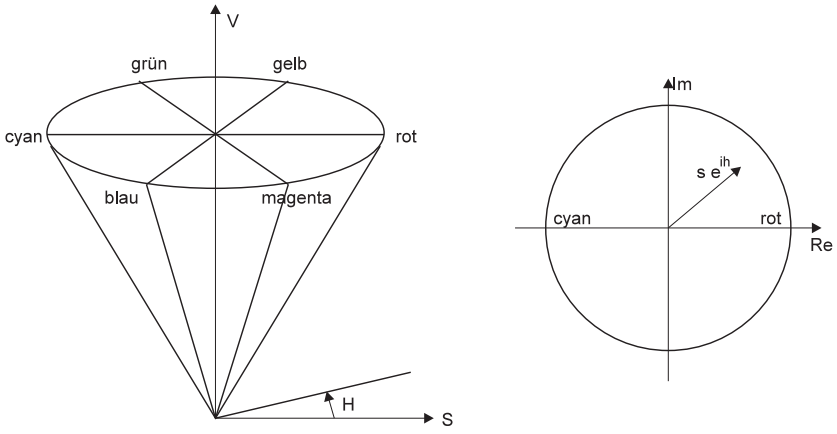

Abbildung 1: Schematische Darstellung des HSV-Farbraumes und komplexe Darstellung der Buntheit

dann im sog. Ortsfrequenz- oder Frequenzraum vor. Die inverse Fouriertransformation bewirkt die entsprechende Umkehrung [9].

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{+\infty} f(t) e^{-i\omega t} dt \\ &= \int_{-\infty}^{+\infty} f(t)(\cos(-\omega t) + i \sin(-\omega t)) dt \\ f(t) &= \frac{1}{2\pi} \int_{-\infty}^{+\infty} F(\omega) e^{i\omega t} d\omega \end{aligned}$$

Die zweite Zeile ist hier angegeben, um zu verdeutlichen, dass es sich um die Extraktion von harmonischen Schwingungsanteilen der Funktionen $\cos \omega t$ und $i \sin \omega t$ handelt, da sich die komplexe Exponentialfunktion unter Anwendung der Euler-Relation in diese zerlegen lässt.

Diese kontinuierliche Fouriertransformation ordnet einem kontinuierlichen Signal ein kontinuierliches Spektrum zu. Einem diskreten Signal wird ein ebenfalls kontinuierliches aber periodisches Spektrum zugeordnet. Die diskrete Abtastung des Spektrums führt auf die *diskrete Fouriertransformation* (DFT), wie sie in der Bildverarbeitung von zu einem Pixelraster dikretisierten Bildern eingesetzt wird. Hierzu existiert ein schneller Algorithmus, der bei Bildern (oder Signalen im Allgemeinen), die ganzzahlige Zweierpotenzen als Kantenlänge besitzen, eine effiziente Berechnung erlaubt und *Fast Fourier Transformation* (FFT) genannt wird.

Die DFT für eindimensionale Signale ist definiert durch

$$F(k) = \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} f(m) w^{-mk} \qquad k = 0, 1, \ldots, M-1$$

wobei $w = e^{i2\pi/M}$ die erste M-te Einheitswurzel ist. Die inverse DFT ergibt sich dann zu

$$f(m) = \frac{1}{\sqrt{M}} \sum_{k=0}^{M-1} F(k) w^{km} \qquad m = 0, 1, \ldots, M-1$$

Da die Gaborfilterung im Ortsraum (bis auf eine Phasenverschiebung, s. u.) eine Faltung darstellt, wird hier kurz die Definition der Faltung angegeben (kontinuierlich/diskret):

$$g(t) * h(t) = (g * h)(t) = \int_{-\infty}^{+\infty} g(\tau) \cdot h(t - \tau) d\tau$$

$$g(m) * h(m) = (g * h)(m) = \frac{1}{M} \sum_{p=0}^{M-1} g(p) h(m - p)$$

Im zweidimensionalen Fall sind die Zusammenhänge entsprechend, sowohl bei der FT, als auch bei der Faltung.

Einige für diese Arbeit wesentliche Eigenschaften der FT sind Faltungstheorem, Superpositionsprinzip, Verhalten bei Stauchung und Frequenzverschiebung [9]:

$$f(t) = (g * h)(t) \quad \Leftrightarrow \quad F(\omega) = G(\omega) \cdot H(\omega) \tag{1}$$

$$f(t) = a \cdot g(t) + b \cdot h(t) \quad \Leftrightarrow \quad F(\omega) = a \cdot G(\omega) + b \cdot H(\omega) \tag{2}$$

$$f(t) = g(b \cdot t) \quad \Leftrightarrow \quad F(\omega) = \frac{1}{|b|} G(\frac{\omega}{b}) \tag{3}$$

$$f(t) = g(t) \cdot e^{i\omega_0} \quad \Leftrightarrow \quad F(\omega) = G(\omega - \omega_0) \tag{4}$$

Außerdem ist die FT rotationsvariant, das heißt, das Frequenzspektrum eines im Ortsraum gedrehten Signals ergibt sich durch Drehung des Spektrums des ursprünglichen Signals.

Jede lokale Veränderung des Signals f bewirkt eine Änderung der FT über der gesamten Frequenzachse. So überdeckt zum Beispiel der Graph der FT der Diracfunktion den gesamten Frequenzbereich. Die FT enthält daher keine lokalen Informationen des Signals f. Dies bedeutet andererseits, dass die Information des Frequenzspektrums den Ortsbereich, in dem die Frequenz auftritt, nicht unmittelbar angibt. Eine Möglichkeit der Lokalisierung der FT im Ortsraum ist die *Fensterfouriertransformation* (WFT), die den lokalen Frequenzinhalt in einem

Fenster g um den Punkt τ beschreibt. Dabei wird für g üblicherweise eine schnell auf 0 abfallende Funktion gewählt, damit sie als Fenster wirkt.

$$F^{\text{Fen}}(\omega, \tau) = \int_{-\infty}^{+\infty} f(t)g(t-\tau)e^{-i\omega t}dt$$

Die Fensterfouriertransformation ist somit von zwei Parametern abhängig, der Frequenz ω und dem Zentrum der Lokalisierung τ. Man spricht deshalb auch von einer Darstellung im Orts-/Frequenzraum. Die Fensterfouriertransformation wird auch als short-time Fourier transform (STFT) bezeichnet

2.3 Gabortransformation

Die WFT mit einer Gaußfunktion $g_\alpha(t)$ als Fensterfunktion wurde von D. Gabor 1946 verwendet:

$$g_\alpha(t) = \frac{1}{2\sqrt{\pi\alpha}} e^{-\frac{t^2}{4\alpha}}$$

Diese spezielle WFT heißt *Gabortransformation*. Bezeichnet man das Ergebnis der Gabortransformation von f mit G_f so ergibt sich wegen der Symmetrie von g_α

$$\begin{aligned} G_f(\omega, \tau) &= \int_{-\infty}^{+\infty} f(t)g_\alpha(t-\tau)e^{-i\omega t}dt \\ &= e^{-i\omega\tau} \int_{-\infty}^{+\infty} f(t)g_\alpha(\tau-t)e^{i\omega(\tau-t)}dt \\ &= e^{-i\omega\tau}(f(\tau) * (g_\alpha(\tau)e^{i\omega\tau})) \\ &= e^{-i\omega\tau}(f(\tau) * h(\tau)) \end{aligned}$$

Im Ortsraum stellt die Gaborfilterung daher bis auf den Faktor $e^{-i\omega\tau}$ eine Faltung dar. Dieser Faktor bewirkt jedoch lediglich eine Phasenverschiebung und kann daher bei Anwendungen, die nur die Amplitude des Ergebnisses berücksichtigen, vernachlässigt werden.

Ein zweidimensionaler Gaborfilter zur Faltung im Ortsraum berechnet sich folgendermaßen als Produkt zweier eindimensionaler Gaborfilter:

$$\begin{aligned} h(x, y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} e^{2\pi i(u_0 x + v_0 y)} \\ &= \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} (\cos(2\pi(u_0 x + v_0 y)) + i\sin(2\pi(u_0 x + v_0 y))) \end{aligned}$$

Hierbei bezeichnet σ die Standardabweichung ($\alpha = \frac{\sigma^2}{2}$), $\tau = (x, y)$ stellt die Ortskoordinate dar und $(u_0, v_0) = (f\cos\theta, f\sin\theta)$ mit Frequenz f und Orientierung

θ ist der Frequenzmittelpunkt des Filters im Frequenzraum. Bei Verwendung der Frequenz f an Stelle der Kreisfrequenz ω ergibt sich wegen des Zusammenhanges $\omega = 2\pi f$ der Faktor 2π. Dabei wird der Cosinusanteil als gerader und der Sinusanteil als ungerader Bestandteil des Filters bezeichnet. Wegen des Faltungstheorems (1) kann eine Faltung im Ortsraum auch als eine Multiplikation im Frequenzraum durchgeführt werden. Dies ist meist schon ab einer kleinen Bildgröße vorteilhaft, da die Faltung bei einer Bildkantenlänge von n die Komplexität $O(n^4)$ besitzt. Dagegen ist die Komplexität von FFT, Multiplikation und inverser FFT nur $O(n^2 \log n + n^2 + n^2 \log n) = O(n^2 \log n)$. Daher bietet es sich an, auch die Gaborfilterung als Multiplikation der Fouriertransformierten im Frequenzraum zu betrachten. Die Fouriertransformierte $H(u,v)$ des Filters $h(x,y)$ ergibt sich durch Anwendung von (2, 3, 4), FT und $g(t) = e^{-\pi t^2} \Leftrightarrow G(f) = e^{-\pi f^2}$ zu

$$\begin{aligned}
H(u,v) &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} e^{2\pi i(u_0 x + v_0 y)} e^{-2\pi i(ux+vy)} dx\, dy \\
&= \frac{1}{2\pi\sigma^2} \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{-\pi\frac{x^2+y^2}{2\pi\sigma^2}} e^{-2\pi i((u-u_0)x+(v-v_0)y)} dx\, dy \\
&= \frac{1}{2\pi\sigma^2} \sqrt{2\pi\sigma^2}^2 e^{-\pi\sqrt{2\pi\sigma^2}^2((u-u_0)^2+(v-v_0)^2)} \\
&= e^{-2\pi^2\sigma^2((u-u_0)^2+(v-v_0)^2)}
\end{aligned}$$

Betrachtet man auch nicht-isotrope Gaußfunktionen, so ergeben sich zwei weitere Freiheitsgrade. Einerseits ein Faktor λ zur Angabe des Verhältnisses der Hauptachsen zueinander und andererseits die Orientierung ϕ der Hauptachsen im Raum. Dazu werden die Gleichungen für rotierte Koordinaten $(u', v') = (u\cos\phi + v\sin\phi, -u\sin\phi + v\cos\phi)$ und analog (u'_0, v'_0) sowie (x', y') (Rotationsinvarianz der FT) angegeben. Dies ergibt eine Drehung der für $\lambda \neq 1$ ellipsenförmigen Gaußglocke um den Winkel ϕ.

$$\begin{aligned}
h(x,y) &= \frac{1}{2\pi\sigma^2\lambda} e^{-\frac{1}{2}\frac{x'^2/\lambda^2+y'^2}{\sigma^2}} e^{2\pi i(u_0 x' + v_0 y')} \\
H(u,v) &= e^{-2\pi^2\sigma^2\left[(u'-u'_0)^2\lambda^2+(v'-v'_0)^2\right]} \\
&= H_g(u,v) + i\cdot H_u(u,v) \\
H_g(u,v) &= \frac{1}{2}\left(e^{-2\pi^2\sigma^2\left[(u'-u'_0)^2\lambda^2+(v'-v'_0)^2\right]} + e^{-2\pi^2\sigma^2\left[(u'+u'_0)^2\lambda^2+(v'+v'_0)^2\right]}\right) \\
H_u(u,v) &= \frac{1}{2i}\left(e^{-2\pi^2\sigma^2\left[(u'-u'_0)^2\lambda^2+(v'-v'_0)^2\right]} - e^{-2\pi^2\sigma^2\left[(u'+u'_0)^2\lambda^2+(v'+v'_0)^2\right]}\right)
\end{aligned}$$

Für den Filter im Frequenzraum wurde hier noch die entsprechende Zerlegung in geraden und ungeraden Anteil dargestellt (Indizes g und u), die sich unter Anwendung der Gleichungen $\cos\beta = 1/2(e^{i\beta} + e^{-i\beta})$ und $\sin\beta = 1/2(e^{i\beta} - e^{-i\beta})$ ergibt.

Die Gaborfilter sind im Ortsraum also komplexe harmonische Funktionen, die durch zweidimensionale, um den Ursprung zentrierte Gaußfunktionen moduliert sind. Im Frequenzraum sind sie verschobene zweidimensionale Gaußfunktionen. Da die Fouriertransformation einer Gaußfunktion wieder eine Gaußfunktion ergibt, stellt das Ergebnis der Gabortransformation sowohl im Orts- als auch im Frequenzraum lokale Information dar. Der Filter kann jede beliebige elliptische Region des Frequenz- oder des Ortsraums überdecken. Ferner erzielt die Gabortransformation – unabhängig von der Anordnung – maximale gleichzeitige Auflösung im Orts- und Frequenzraum, das heißt die Gaußfunktion erreicht als (einzige) Fensterfunktion das Minimum der Unschärferelation $\sigma_g^2 \cdot \sigma_G^2 \geq \frac{\pi}{2}$, wobei σ_g^2 die Varianz der Fensterfunktion im Ortsraum (Ortsunschärfe) und σ_G^2 entsprechend die im Frequenzraum (Frequenzunschärfe) angibt [9, 4]. Daraus ergibt sich direkt der reziproke Zusammenhang zwischen den Unschärfen und damit ein wichtiger trade-off. Das heißt um die Auflösung im Ortsraum zu verdoppeln, muss eine halbierte Auflösung im Frequenzraum in Kauf genommen werden, und umgekehrt. Filter mit geringer Bandbreite im Frequenzraum sind erwünscht, da sie eine feine Unterscheidung zwischen verschiedenen Texturen erlauben. Andererseits sind für eine genaue Erkennung von Texturgrenzen Filter nötig, die im Ortsraum eine geringe Bandbreite aufweisen.

Eine weitere interessante Eigenschaft von Gaborfiltern ist, dass sie eine gute Annäherung an die Empfindlichkeitsprofile von Neuronen im visuellen Kortex zu sein scheinen, in der Art, dass sie frequenz- und richtungsspezifische Signale verarbeiten [3, 8, 7]. Dies muss jedoch nicht in Zusammenhang mit einer guten Eignung für die digitale Texturanalyse stehen.

Für den Zusammenhang zwischen Standardabweichung σ im Ortsraum, der Frequenz f und den Bandbreiten halber Intensität in Frequenz B (in Oktaven) und in Orientierung Ω gelten die Gleichungen [8, 3]

$$\begin{aligned} B &= \log_2 \frac{\pi f \lambda \sigma + \gamma}{\pi f \lambda \sigma - \gamma} \\ \Omega &= 2 \arctan \frac{\gamma}{\pi f \sigma} \end{aligned}$$

mit $\gamma = \sqrt{(\ln 2)/2}$. Damit ergeben sich als freie Parameter eines zweidimensionalen Filters beispielsweise Bandbreite B in Frequenz, Bandbreite Ω in Orientierung, Frequenz f und Orientierung θ.

2.4 Gaborexpansion

Bei entsprechender Verteilung der Filter im Frequenzraum ist es möglich, das ursprüngliche Signal aus den gefilterten Signalen näherungsweise zurückzugewinnen [2, 3, 8, 12, 7]. Die Zerlegung wird dann auch *Gaborexpansion* genannt. Die

Rekonstruktion kann folgendermaßen als inverse FT der für konstantes ω über τ integrierten gefilterten Signale hergeleitet werden [12]:

$$\begin{aligned}
\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} G_f(\omega,\tau)e^{i\omega t}d\tau d\omega &= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(t')g(t'-\tau)e^{-i\omega t'}dt'd\tau e^{i\omega t}d\omega \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{i\omega(t-t')}d\omega f(t')g(t'-\tau)dt'd\tau \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} 2\pi\delta(t-t')f(t')g(t'-\tau)dt'd\tau \\
&= 2\pi f(t)\int_{-\infty}^{+\infty} g(t-\tau)d\tau \\
&= 2\pi f(t)
\end{aligned}$$

In der praktischen Durchführung erwies sich jedoch eine andere Methode als günstiger, deren Begründung im folgenden dargestellt ist. Sie besteht darin, die gefilterten Signale, die sich bei der Berechnung unter Vernachlässigung des Faktors $e^{-i\omega\tau}$ ergeben, über ω zu integrieren:

$$\begin{aligned}
\int_{-\infty}^{+\infty} e^{i\omega\tau}G_f(\omega,\tau)d\omega &= \int_{-\infty}^{+\infty} e^{i\omega\tau}\int_{-\infty}^{+\infty} f(t)g(t-\tau)e^{-i\omega t}dtd\omega \\
&= \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} e^{i\omega(\tau-t)}d\omega f(t)g(t-\tau)dt \\
&= \int_{-\infty}^{+\infty} 2\pi\delta(\tau-t)f(t)g(t-\tau)dt \\
&= 2\pi f(\tau)g(0) \\
&= \frac{\sqrt{2\pi}}{\sigma}f(\tau)
\end{aligned}$$

Mit beiden Methoden lässt sich also das ursprüngliche Signal (bis auf einen konstanten Faktor) wieder zurückgewinnen. In der Anwendung ergeben sich an Stelle der Integrationen Summationen. Dies erklärt das bessere Resultat der zweiten Methode in der Praxis. Bei einer Diskretisierung von τ in N und von ω in M Werte ergibt sich bei der zweiten Methode eine Reduktion der Information der gefilterten Bilder durch Summation über ω auf N Werte. Bei der ersten Methode wird zunächst auf M Werte reduziert und dann eine Darstellung mit N Werten durch die inverse FT erzeugt. Da in der Implementierung M im Allgemeinen bedeutend kleiner als N ist (z.B. $N = 128 \times 128 = 16384, M = 80$) erweist sich diese

Vorgehensweise als nachteilig. Um eine vollständige[1] Rückgewinnung zu ermöglichen muss gelten $\Delta\omega\Delta\tau = 2\pi$ bzw. $\Delta f\Delta\tau = 1$ [7, 12, 3]. Dies würde bedeuten, dass zu einem Bild mit n^2 Pixeln auch n^2 Gaborfilterungen durchgeführt werden müssten, da wegen der Diskretisierung $\Delta\tau \geq 1, \Delta f \geq 1$.

2.5 Spezifika

In der Implementierung wurden die Gaborfilter einerseits in einer Anordnung über einem gleichmäßigen Gitter mit konstanten Bandbreiten angeboten. Andererseits können sie an die Frequenz, die betrachtet werden soll, angepasst werden. In dieser adaptiven Form ähnelt die hier verwendete Filterung der Wavelet-Transformation, denn das Fenster hat im Frequenzraum konstante Oktaven-Bandbreite und Oktaven-Abstand. Anders ausgedrückt wird dadurch das Fenster im Ortsraum um so kleiner, je höher die betrachtete Frequenz ist. Die Gaborfunktionen sind zulässige Wavelets, erreichen jedoch keine orthogonale Dekomposition, was bedeutet, dass in der Zerlegung redundante Information enthalten ist [8, 12].

Die Effekte der Diskretisierung der Filter müssen bei der Verwendung in der digitalen Bildverarbeitung berücksichtigt werden. Die wesentliche Einschränkung ist dabei, dass die benutzte Frequenz kleiner als die halbe Bildbreite sein sollte, um keine Fehler durch die Diskretisierung zu erhalten [3]. Dies ergibt sich rein anschaulich jedoch schon dadurch, dass der Mittelpunkt eines solchen ungünstigen Filters außerhalb des Bildes der diskreten Fouriertransformation liegt.

Ein nicht-triviales Problem bei der Berechnung der Merkmalsvektoren ist die Auswahl der Filterparameter. In [3] wird vorgeschlagen, zur Bestimmung von für die Textur beziehungsweise das Bild charakteristischen Frequenzen das Frequenzspektrum nach Maxima zu durchsuchen. Hier wurde jedoch eine gleichmäßige Verteilung der Filter im Frequenzraum, wie in [8] verwendet.

[1] Genaugenommen ergeben sich nicht die exakten Koeffizienten β_{rm} der Gaborzerlegung durch die Gaborfilterung. Die exakte Gaborzerlegung erfüllt die Gleichung $f(t) = \sum_r \sum_m \beta_{rm} h_{rm}(t)$ für geeignete Gaborfilter im Ortsraum h_{rm}. Es handelt sich daher trotzdem nur um eine Näherung.

3 Implementierung

Die Implementierung des Verfahrens wurde in der Entwicklungsumgebung KHOROS mit der Programmiersprache C durchgeführt. Im Folgenden werden die Bestandteile der Reihe nach vorgestellt. KHOROS ist eine visuelle Programmierumgebung, die es erlaubt, Informationen – wie zum Beispiel Bilder – in einem einheitlichen Format zwischen Programmen auszutauschen. Diese Ein-/Ausgabeobjekte sind fünfdimensional mit den Dimensionen Breite, Höhe, Tiefe, Zeit und „Elemente“. Letztere wird zur Angabe von vektorwertigen Informationen benutzt. Ein RGB-Bild hat dann zum Beispiel die Größe $(128, 128, 1, 1, 3)$. KHOROS stellt schon eine Vielzahl von Funktionen zur Bearbeitung von Bildern bereit, so zum Beispiel die FFT oder arithmetische Operationen. Programme werden auf der Oberfläche (`cantata`) durch sogenannte Glyphs dargestellt, die es erlauben die nötigen Parameter des zugehörigen Programms anzugeben. Eine als K-Routine bezeichnetes Programm übernimmt den Aufruf der Library-Funktion, die die eigentliche Funktionalität des Programms enthält. Im folgenden werden deshalb nur die Library-Funktionen betrachtet.

3.1 Farbraumtransformation

Zur Zerlegung von Bildern in komplexwertige Buntheits- und reellwertige Helligkeitsbilder und die entsprechende Rücktransformation wurden zwei Routinen implementiert, `hsv2cplx` und `cplx2hsv`. Routinen zur Umwandlung zwischen den Farbräumen RGB und HSV existieren in KHOROS bereits, so dass von Daten im HSV-Raum ausgegangen werden kann.

Die Routine `hsv2cplx` erwartet ein Eingabeobjekt in HSV-Darstellung, das heißt mit 3 Eingabebändern. Zwei optionale Ausgabeobjekte (signalisiert durch `flag`-Variablen) können angegeben werden, wobei in das erste die komplexe Darstellung der Buntheitsdaten (H/S, wie oben erläutert) und die Maskierungsdaten geschrieben werden, und in das zweite die Helligkeitsdaten (V). Daraus ergibt sich der Funktionstyp

```
int lhsv2cplx(kobject eingabe,
              int o_flag,
              kobject ausgabe,
              int ov_flag,
              kobject ausgabe2)
```

Die Implementierung ist direkt, da lediglich eine Umrechnung der Eingabedaten (H/S) nötig ist:

```
/*calculate complex output from H and S*/
for (i=0; i<gesamt; i++)
     hsdaten[i]=kdcp2r(kdccomp(sdaten[i],KPI*hdaten[i]/180.0));
```

(`kdcp2r` : KHOROS double complex polar to rectangular , `kdccomp` : KHOROS double complex composition (from two real)) Die V-Daten müssen lediglich übertragen werden, da sie schon in der Eingabe getrennt vorliegen.

Die Routine `cplx2hsv` realisiert die umgekehrte Transformation. Dabei ist ein Objekt, das die komplexe Darstellung der Buntheitsdaten enthält, als Eingabe notwendig. Weiterhin kann ein Objekt mit den Helligkeitsdaten angegeben werden. Andernfalls wird für die Helligkeit ein konstanter Wert angenommen, der vom Benutzer spezifiziert werden kann. Falls die Maskendaten nicht im ersten Eingabeobjekt zu finden sind, kann ein Objekt, das diese enthält, angegeben werden. Schließlich kann ein Ausgabeobjekt festgelegt werden. Es ergibt sich hier der Funktionstyp

```
int lcplx2hsv(kobject eingabe1,
              int iv_flag,
              kobject eingabe2,
              int im_flag,
              kobject eingabem,
              double defval,
              int o_flag,
              kobject ausgabe)
```

Auch hier ist die Implementierung direkt:

```
  /*calculate H and S data*/
  for (i=0; i<gesamt; i++)
     {
      hdaten[i]=kdcang(hsdaten[i])*180.0/KPI;
      if (hdaten[i]<0.0) hdaten[i]+=360.0;
      sdaten[i]=kdcmag(hsdaten[i]);
     }
```

(`kdcang`: KHOROS double complex angle, `kdcmag`: KHOROS double complex magnitude)

In Abbildung 2 ist ein RGB Bild zu sehen, sowie der HS-Anteil und der V-Anteil, das heißt die Helligkeit. Der Buntheitskanal ist komplexwertig und muss daher zur Darstellung ins Reelle übertragen werden. Dazu wurde hier und im weiteren jeweils der Logarithmus des um 1 inkrementierten Betrages der komplexen Zahl verwendet, also $z \mapsto \log(|z| + 1)$.

3.2 Gabortransformation in Orts- und Frequenzraum

Gaborfilter wurden sowohl im Ortsraum als auch im Frequenzraum implementiert, um einen Vergleich zwischen beiden Methoden zu ermöglichen und so Fehler

Abbildung 2: Umwandlung von RGB in HS und V

erkennen zu können (Die Formeln aus [5, 3] wurden verwendet). Die entsprechenden Routinen heißen `csgabor` (complex spatial domain gabor filter) und `cfgabor` (complex frequency domain gabor filter). Da die Filterung im Ortsraum (näherungsweise, s. o.) eine Faltung darstellt, ist die Filterung im Frequenzraum einschließlich der durchgeführten FFT erwartungsgemäß deutlich schneller. Für alle weiteren Anwendungen wurde deshalb nur die Filterung im Frequenzraum benutzt. Das gefilterte Bild enthält für jedes Pixel Information darüber, wie stark die gegebene Frequenz mit der angegebenen Orientierung in der Umgebung des Pixels im ursprünglichen Bild vertreten ist.

Die Routine `csgabor` hat die folgenden Parameter:

```
int lcsgabor(kobject eingabe,
             int o_flag,
             kobject ausgabe,
             int of_flag,
             kobject ausgabef,
             float freq,
             float deviation,
             int phase,
             int filtertype)
```

Der Parameter `eingabe` bezeichnet dabei das komplexe Eingabeobjekt. `o_flag` gibt an, ob eine Ausgabe des gefilterten Bildes erfolgen soll, und zwar in das Objekt `ausgabe`. Analoges gilt für `of_flag` und `ausgabef` in Bezug auf den Filter. Der Parameter `freq` gibt die Frequenz des Filters in Zyklen pro Bildbreite an. Sinnvollerweise sollte die Frequenz die halbe Bildbreite nicht überschreiten (Nyquist, vgl. 2.5). Der folgende Wert `deviation` gibt die Standardabweichung des Filters im Ortsraum an. Je größer dieser Wert ist, desto ungenauer wird einerseits die Auflösung im Ortsraum, andererseits wird aber gleichzeitig die Auflösung im Frequenzraum genauer. `phase` gibt die Phase des Filters in Grad an, das heißt die Richtung, in der die entsprechende Frequenz untersucht wird. Dabei werden

die Winkel im Uhrzeigersinn von der positiven x-Achse aus angegeben, da die y-Achse der Bilder im Gegensatz zur sonst üblichen mathematischen Orientierung nach unten zeigt. Diese drei zuletzt beschriebenen Parameter bestimmen dabei das Verhalten des Filters. Sie sind unabhängig voneinander, wie im theoretischen Teil beschrieben. Im Ortsraum wurden nur isotrope Filter betrachtet, weshalb der Parameter λ entfällt. Der Parameter `filtertype` schließlich gibt an, welche Art von Filter angewendet werden soll. Die zulässigen Werte sind die folgenden:

1: Es wird der gerade Anteil (cos) als reeller Anteil verwendet:
$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} \cos(2\pi(u_0 x + v_0 y))$$

2: Der ungerade Anteil (sin) wird als reeller Anteil benutzt:
$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} \sin(2\pi(u_0 x + v_0 y))$$

3: Der ungerade Anteil (sin) wird als imaginärer Anteil benutzt:
$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} i \sin(2\pi(u_0 x + v_0 y))$$

4: Beide Filter werden gleichzeitig (als komplexe Zahl) verwendet:
$$h(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{x^2+y^2}{\sigma^2}} (\cos(2\pi(u_0 x + v_0 y)) + i \sin(2\pi(u_0 x + v_0 y)))$$

Die Implementierung berechnet zunächst den entsprechenden Filter, der dann mit dem Bild gefaltet wird. Die Faltung hat einen Aufwand der Ordnung $O(n^4)$ für eine Bildbreite und -höhe von n und bestimmt daher die Laufzeit[2]. Auf eine Angabe des Quelltextes wurde im Folgenden aufgrund der erheblichen Länge verzichtet, lediglich die Funktionsköpfe werden dargestellt. Abbildung 3 zeigt ein Beispiel eines Filters im Ortraum.

Das Ergebnis des Filterns kann dann durch Faltung mit der Eingabe berechnet werden, wobei die Bilder als wrap-around betrachtet werden (modulo n):

$$(f * h)(x,y) = \sum_{p=0}^{n-1} \sum_{q=0}^{n-1} f(x-p, y-q) h(p,q)$$

Die Routine `cfgabor` hat die folgenden Parameter:

```
int lcfgabor(kobject eingabe,
             int o_flag,
             kobject ausgabe,
             int of_flag,
             kobject ausgabef,
             float freq,
```

[2]Wegen der Vergleichbarkeit mit der die Fouriertransformation benutzende Routine sind die Anwendungen auf quadratische Bilder mit ganzzahligen Zweierpotenzen als Kantenlänge beschränkt.

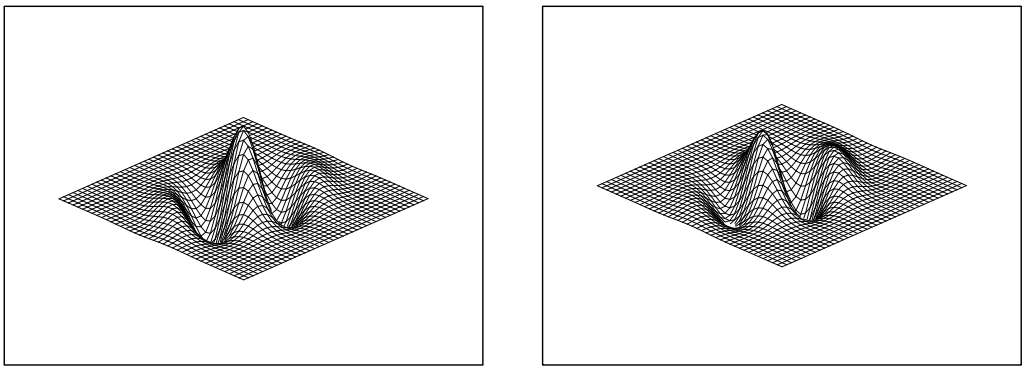

Abbildung 3: Gaborfilter im Ortsraum, gerader Anteil (reell) und ungerader Anteil (imaginär)

```
float ratio,
float deviation,
int phase,
int ordiff,
int filtertype,
int expfac)
```

Dabei haben alle Parameter, die auch bei `csgabor` auftreten, die gleiche Bedeutung, insbesondere gibt der Parameter `deviation` auch die Standardabweichung im Ortsraum an, um die Ausgaben vergleichen zu können. Zusätzlich sind hier noch weitere Angaben möglich. Der Parameter `ratio` gibt das Verhältnis der beiden Hauptachsen des (für Werte ungleich 1 ellipsenförmigen) Filters an. `ordiff` gibt die Differenz in der Orientierung der Hauptachsen und der Phase des Filters an und wird meist gleich 0 gesetzt. Mit `expfac` kann ausgewählt werden, ob die Phasenkorrektur mit dem Faktor $e^{i\omega\tau}$ durchgeführt werden soll (vgl. den theoretischen Teil). Diese Parameter sind Erweiterungen der grundlegenden Funktionalität und daher nur in der weiterverwendeten Version, die im Frequenzraum arbeitet, verfügbar.

Die Implementierung berechnet auch hier zunächst den entsprechenden Filter und transformiert dann die Eingabe mittels FFT in den Frequenzraum, wo sie mit dem Filter multipliziert wird. Das Ergebnis wird mittels inverser FFT in den Ortsraum zurücktransformiert und nach eventueller Korrektur ausgegeben. Die Ausgabe des Filters geschieht im Frequenzraum. Die Filter berechnen sich wie im theoretischen Teil angegeben.

In Abbildung 4 ist ein Gaborfilter im Orts- und im Frequenzraum dargestellt. Die verwendeten Parameter sind Orientierung 0°, Frequenz 15 Zyklen/Bildbreite,

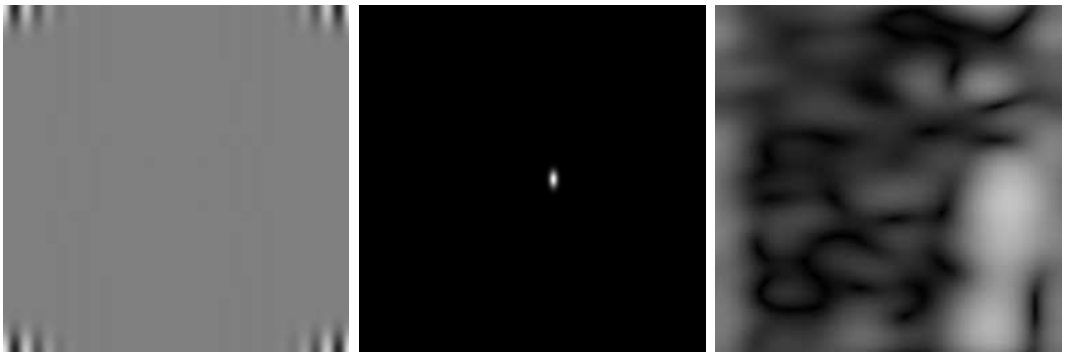

Abbildung 4: Gaborfilter im Ortsraum (Realteil) sowie im Frequenzraum und gefiltertes Bild

Standardabweichung 10 und Achsenverhältnis 2. Der Filter im Ortsraum ist nicht in der Mitte des Bildes zentriert, sondern hat den Koordinatenursprung in der oberen linken Ecke. Wegen der wrap-around Sichtweise kann man sich den Filter aber auch entsprechend verschoben vorstellen. Das dritte Bild der Abbildung zeigt das Ergebnis der Filterung des Bildes aus Abbildung 2. Man erkennt gut die großen Werte, die beim Übergang von Blau nach Rot an der Nase entstehen. Weitere Stellen mit starken Antworten sind die horizontalen Übergänge der Augen und der Bildrand in x-Richtung, der wegen der „wrap-around“-Sichtweise erkannt wird.

3.3 Texturmerkmalsextraktion

Zur Zerlegung eines Bildes in verschiedene Anteile mittels Gabor-Filtern wurde die Routine `decompose` implementiert. Dabei wird die Filterung wegen der besseren Performanz im Frequenzraum durchgeführt. Um die Güte der Dekomposition zu testen, wurde auch eine Routine `recompose` zur Rekombinierung der gewonnenen Kanäle geschrieben. „Güte“ bezieht sich dabei auf die Möglichkeit der Rückgewinnung des ursprünglichen Bildes aus der Zerlegung. Es zeigt sich, dass eine gute Rückgewinnung noch einer Korrektur des Frequenzspektrums bedarf (siehe 3.5). Dies hängt auch mit der oben erwähnten Redundanz der Gaborzerlegung zusammen.

Das Ergebnis der Zerlegung ist ein Bild (im Orts-/Frequenzraum), das für jedes Pixel einen Vektor von komplexen Werten enthält, die die Stärke und Phase der jeweiligen Frequenz und Orientierung in der Umgebung darstellen. Dieser Vektor lässt sich daher zur Klassifizierung oder Segmentierung von Texturen benutzen. Durch Konkatenation können sowohl die Informationen aus der Buntheit als auch die der Helligkeit benutzt werden.

Die Routine `decompose` hat den folgenden Kopf:

```
int ldecompose(int i_flag, kobject eingabe,
               int iv_flag, kobject eingabev,
               kobject ausgabe,
               kobject ausgabef,
               int expfac, int pos,
               int numpha, int numphalow,
               float dcfac, int begin,
               int numfil)
```

Dabei geben die `flag`-Variablen an, welche Eingaben vorhanden sind. Mindestens eine der Eingaben `eingabe` (für die komplexe Eingabe, hier also die Buntheit) oder `eingabev` (für die reellwertige Eingabe, hier die Helligkeit) muss vorhanden sein. Die Ausgaben enthalten die Vektoren der Zerlegung (`ausgabe`) und die benutzten Filter im Frequenzraum (`ausgabef`). Der Parameter `expfac` gibt an, ob der Phasenkorrekturfaktor $e^{i\omega\tau}$ benutzt werden soll. Dies ist nur nötig, wenn auch die Phasen verwendet werden, zum Beispiel zur Segmentierung von verschobenen Texturen. Der folgende Eingabewert `pos` wählt zwischen einer Positionierung der Filter auf konzentrischen Kreisen (Wert 1) mit Anpassung der Filterdimensionen an die Position oder einer Positionierung auf einem quadratischen Gitter (Wert 2). Für den zweiten Fall gibt der Parameter `numfil` die Anzahl der Filter pro Seite des Gitters an, für den ersten Fall bestimmen die weiteren Angaben die Positionierung. Der Parameter `numpha` gibt die Anzahl der zu verwendenden Phasen für die hohen Frequenzen an, `numphalow` die Anzahl der Phasen für die niedrigen Frequenzen. Um eine gute Rekonstruktion zu erreichen sollte der zweite Wert niedrig sein (4 hat sich dazu als brauchbar erwiesen), da sich sonst eine zu starke Überlappung der Filter ergibt. Für Klassifizierungsaufgaben könnten auch höhere Werte sinnvoll sein, um auch bei niedrigen Frequenzen andere Orientierungen, zum Beispiel die Richtungen der Winkelhalbierenden ($45° \ldots$) zu berücksichtigen. Nach [10] ist die Relevanz für die Unterscheidung von Texturen generell größer für hohe Frequenzen. Die Anzahl der Phasen für eine bestimmte Frequenz wird mittels linearer Interpolation zwischen den gegebenen Extremwerten errechnet. Die Anzahl verschiedener Frequenzen wird durch die Breite der Eingaben bestimmt. Der Parameter `begin` gibt dabei an, bei welcher der ermittelten Frequenzen begonnen werden soll. Die ermittelten Frequenzen sind $1\sqrt{2}, 2\sqrt{2}, 4\sqrt{2}, 8\sqrt{2}, \ldots \frac{n}{4}\sqrt{2}$, was eine Gesamtzahl von Frequenzen von maximal $\log_2(n) - 1$ ergibt. Benutzt werden die Filter ab der Frequenz $2^{\texttt{begin}-1}\sqrt{2}$. Damit sind die Filter jeweils eine Oktave voneinander entfernt, da ihr Frequenzverhältnis genau 2 beträgt. Der Faktor $\sqrt{2}$ ergibt sich aus der (willkürlichen) Forderung, die äußersten Filter sollen am Bildrand eine Intensität von 50% erreichen. Die Filter werden auf die gleiche Art bestimmt, wie dies in der Routine `cfgabor` geschieht. Die benutzten Frequenzen und Orientierungen werden dabei zur Kontrolle über die Standardausgabe ausgegeben. Die Filter werden im Mit-

telpunkt („Gleichstromkomponente“) auf 0 gesetzt, damit die konstanten Anteile die Filterausgabe nicht beeinflussen. Zusätzlich wird die Gleichstromkomponente als eigener Kanal betrachtet und an den Bildvektor angehängt. Der Wert `dcfac` gibt an, mit welchem Faktor der „Gleichstromanteil“ (DC component) des Bildes multipliziert werden soll. Für eine gute Rekonstruktion sollte dieser Wert 1 betragen, bei der Klassifikation mit der Nearest-Neighbor-Methode scheint teilweise ein größerer Wert sinnvoll zu sein, damit er im Vergleich zu den anderen Vektoranteilen eine signifikante Bedeutung bekommt. Die Dimension der entstehenden Vektoren ist also abhängig von der Größe des Bildes und der Anzahl der betrachteten Phasen und Frequenzen. Die verbleibenden Parameter der Filter ergeben sich wie folgt: Die Verhältnisse der Filter und die Standardabweichungen werden so gewählt, dass sie sich an den Stellen halber Intensität berühren, um eine möglichst gute Abdeckung bei geringer Überlappung (und damit geringer Redundanz) zu gewährleisten. Die Berechnung der Parameter ergibt sich aus den Formeln für die Bandbreiten halber Intensität in Frequenz B und in Orientierung Ω, wie sie im theoretischen Teil angegeben sind. Die Orientierung der Hauptachsen schließlich wird gleich der Orientierung des Filters selbst gewählt. Für die reellwertige Eingabe (Helligkeit) wird die Hälfte der Filter benutzt, da die FFT in diesem Fall konjugiert-symmetrische Ergebnisse liefert. Für die Zerlegung werden jeweils Filter mit reellem (geradem) und imaginärem (ungeradem) Anteil benutzt (vgl. `cfgabor`). Auf den komplexwertigen Eingaben führen die anderen Filter zu Interferenzen zwischen den beiden Anteilen, die die Ergebnisse weniger aussagekräftig machen. Für rein reellwertige Eingaben sollten jedoch auch z.B. gerade Filter zu guten Ergebnissen führen.

Die Routine `recompose` hat den folgenden Kopf.

```
int lrecompose(kobject eingabe,
               int f_flag,
               kobject eingabef,
               int method,
               kobject ausgabe)
```

Dabei gibt `eingabe` das Eingabeobjekt an, bestehend aus den (z.B. von `decompose` getrennten) verschiedenen Kanälen in der Elementebene. Der Parameter `f_flag` gibt an, ob auch die bei der Dekomposition benutzten Filter als Eingabe zur Verfügung stehen. Diese sind dann als `eingabef` anzugeben. Der Schalter `method` wählt zwischen einfacher Addition (Wert 1) oder komplizierterer Rekonstruktion (Wert 2) aus. Die zweite Methode ist erheblich laufzeitintensiver als die erste. Schließlich wird das Ausgabeobjekt angegeben.

Bei der einfachen Methode werden lediglich alle Kanäle der Eingabe addiert; die Filter sind dazu nicht nötig. Diese Methode lässt sich dadurch rechtfertigen, dass gilt

$$\int_{-\infty}^{+\infty} e^{i\omega\tau} G_f(\omega, \tau) d\omega = \frac{\sqrt{2\pi}}{\sigma} f(\tau)$$

für alle τ (vgl. theoretischen Teil). Im diskreten Fall und bei der Beschränkung auf relativ wenige Filter entspräche dies einer Summation.

Abbildung 5: Ursprüngliche HS-Komponente eines Teilbildes und mit Methode 1 bzw. 2 zurückgewonnene.

Methode zwei ist die Implementierung einer anderen Rekonstruktionsformel aus [3], die jedoch deutlich schlechtere Ergebnisse brachte (Abbildung 5), und zudem als Faltung erheblich mehr Rechenzeit benötigt. Ein ausgiebiges Testen der zweiten Methode scheitert an der extremen Laufzeit[3].

Es existiert außerdem eine weitere Routine `invgabor`, die die Rekonstruktion nach der ersten im theoretischen Teil angegebenen Methode implementiert. Sie liefert jedoch schlechtere Ergebnisse, was mit den dort angeführten Argumenten erklärt werden kann.

3.4 Klassifikation anhand eines Beispielbildes

Um die Klassifikation von Farbtexturen mittels der vorgestellten Programme zu veranschaulichen wurde an dem schon oben verwendeten Beispielbild eine Textursegmentierung mit einem k-Nearest-Neighbor-Klassifikator durchgeführt. Dazu wurden kleine Regionen des Bildes ausgewählt und aus ihnen durch Clustering jeweils 10 Vektoren als Basis für die Klassifikation erzeugt. Zum Vergleich

[3]Die Berechnung brauchte für ein Bild der Größe 128×128 auf einer Sparc Ultra2 Workstation ca. 7 Stunden Rechenzeit.

wurde die Segmentierung auch nur aufgrund der RGB-Daten durchgeführt. Außerdem wurden verschieden Parametereinstellungen der Gaborzerlegung benutzt und Buntheits- sowie Helligkeitsdaten getrennt und kombiniert betrachtet.

Um den in KHOROS vorhandenen k-NN-Klassifikator benutzen zu können, mussten die komplexen Daten in reelle umgewandelt werden. Dazu wurde eine weitere Routine `cplx2real` implementiert. Das Programm trennt die komplexen Daten in Betrag und Phase auf, wobei die Phasen mit einem vom Benutzer anzugebenden Faktor multipliziert werden[4]. Es werden jedoch nicht die Phasen selber, sondern die Phasendifferenzen zwischen den einzelnen Filterebenen berechnet, da sie auch Informationen enthalten, die nicht von der Plazierung der Textur abhängen. Zur Gewinnung von signifikanteren Merkmalen aus den Phasen siehe [7]. Wahlweise kann auch nur der Betrag verwendet werden, was die entstehenden Vektoren etwa um den Faktor 2 verkürzt. Dies wiederum vereinfacht die Klassifikation. In der aktuellen Version wird statt des Betrages das Quadrat des Betrages ausgegeben, was der Benutzung des Energiespektrums entspricht. Dies scheint keinen wesentlichen Einfluss auf die Klassifikation zu haben, ist jedoch konformer mit den in der Literatur erwähnten Verfahren.

Die Routine hat folgenden Kopf

```
int lcplx2real(kobject eingabe,
               kobject ausgabe,
               int phaseoutput,
               float phasefactor)
```

Dabei gibt `phaseoutput` an, ob die Phasendifferenzen ausgegeben werden sollen (Wert 1) oder nicht (Wert 2). Falls sie ausgegeben werden, gibt `phasefactor` den Koeffizienten für die Phasenwerte an.

In Abbildung 6 sind einige der Ergebnisse zu sehen. Zunächst fällt auf, dass das Ergebnis der Klassifizierung anhand der RGB-Daten mehr Ähnlichkeit mit dem Original zu haben scheint, aber keine Segmentierung des Bildes in Regionen liefert. Dies geschieht jedoch bei Verwendung der Daten der Gaborfilter relativ gut, insbesondere, wenn man die geringe Anzahl der Trainingsdaten berücksichtigt. Die Regionen sind hier wegen der wrap-around-Sichtweise als über den Bildrand hinaus fortgesetzt zu betrachten. Die Segmentierungen, die in den Bildern 3, 4 und 5 zu sehen sind unterscheiden sich nicht wesentlich. Hier sind zur Feststellung der besten Parametereinstellung jeweils problemspezifische Tests nötig. Allerdings ist ein deutlicher Vorteil gegenüber Bild 6 zu erkennen, in dem die Segmentierung ohne Verwendung der transformierten Farbdaten gezeigt ist. Dies bestätigt die Eignung der Methode zur Analyse von Farbtexturen.

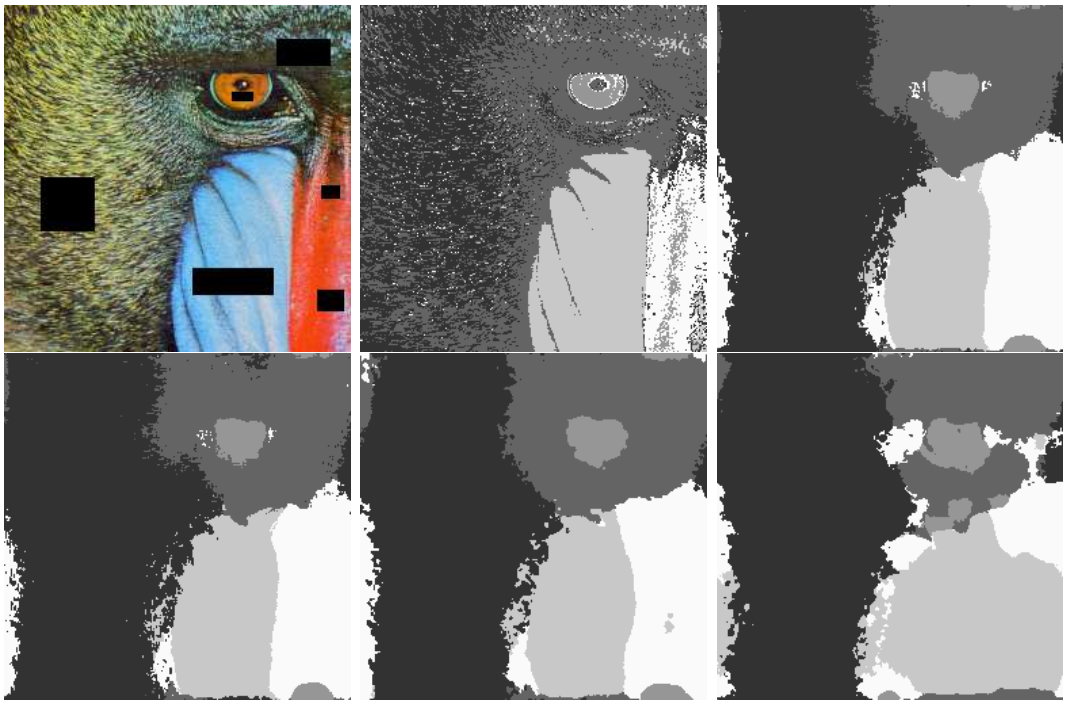

Abbildung 6: (1) Als Basis der Klassifikation benutzte Regionen des Bildes (2) Segmentierung anhand der RGB-Kanäle (k-NN auf den dreielementigen Vektoren) (3) Segmentierung mit 4 Orientierungen und Farb- und Helligkeitsinformationen (4) ebenso, aber nur mit Farbinformationen (5) Segmentierung mit 9 Orientierungen und nur Buntheitsinformationen (6) ebenso, aber nur mit Helligkeitsinformationen

3.5 Beispiel-Workspace

Abschließend wird hier der erstellte Beispiel-Workspace (`beispiel.wk`) vorgestellt. Dieser enthält im wesentliche alle Programme, die im Verlauf der Studienarbeit erstellt wurden, in einer Verknüpfung, die die Verwendung deutlich macht und zu den vorgestellten Ergebnissen führt.

Ein Teil des Workspace (Abbildung 7) stellt die Benutzung der Gabortransformation alleine vor. Hier wird zunächst ein existierendes RGB-Bild auf das Format 128×128 gebracht und in den HSV-Farbraum transformiert (vgl. die Abbildungen der vorangegangenen Abschnitte). Dann wird die HS-Komponente in das komplexe Format umgewandelt und die Maskendaten werden entfernt. Schließlich wird die Gabortransformation angewendet und der verwendete Filter sowie das gefilterte Bild angezeigt (siehe Abbildung 4).

[4] Es scheint, dass sie bei gleicher Gewichtung einen zu starken Einfluss haben.

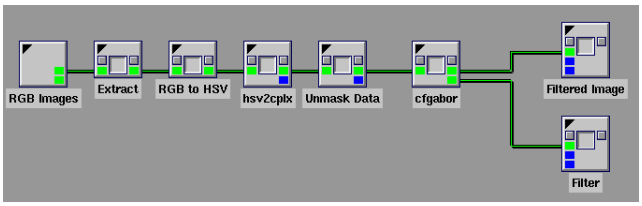

Abbildung 7: Teil des Workspace zur Gabortransformation

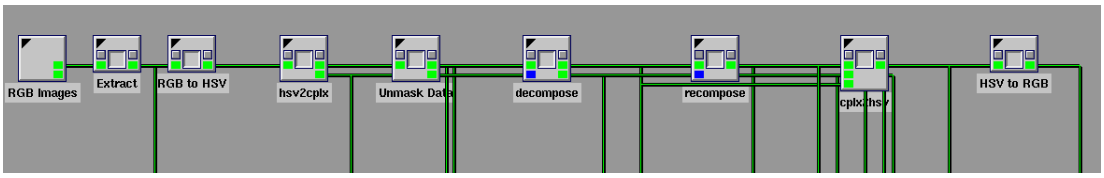

Abbildung 8: Teil des Workspace zur De- und Rekomposition

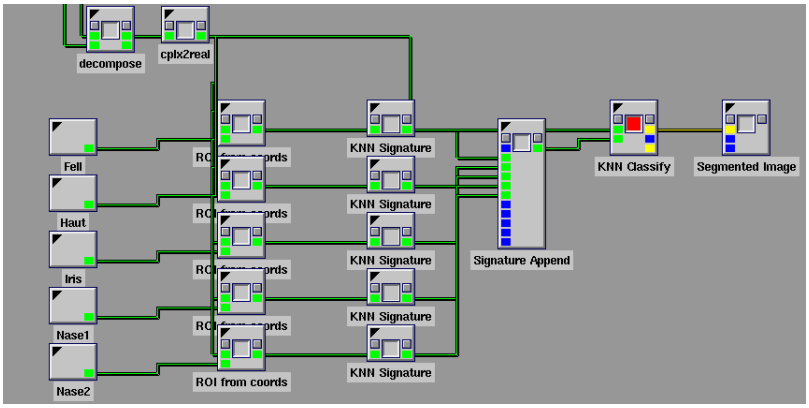

Abbildung 9: Teil des Workspace zur Bildsegmentierung

Ein weiterer Teil beschäftigt sich mit der Zerlegung des Beispielbildes in verschiedene Gaborkomponenten und der Rückgewinnung des Originalbildes daraus (Abbildung 8). Hier wird das Bild wieder zunächst ausgeschnitten, in den HSV-Raum transformiert, in die komplexe Darstellung umgewandelt und demaskiert. Dann wird es jedoch in verschiedene gefilterte Bilder dekomponiert und anschließend rekomponiert. Die Rekomposition wird aus der komplexen Darstellung zunächst in den HSV-Farbraum und dann in den RGB-Raum zurückübertragen, wobei die Helligkeitsinformation des Originalbildes verwendet wird. In der Abbildung nicht zu sehen ist das weitere Verfahren, bei dem zum Ausgleich der Verluste, die im Verlauf der Zerlegung in wenige Gaborfilter entstehen, das erhaltene Bild im Frequenzraum durch die Summe der Filter geteilt wird. Diese Korrektur ist in

der Routine `recompose` nicht enthalten. Das so erhaltene Bild hat nach der Rücktransformation in die RGB-Darstellung eine maximale Abweichung vom Original von 1, das heißt es entspricht diesem fast vollkommen. In allen Zwischenstufen kann der Betrachter sich die Ergebnisse anzeigen lassen. Diese sind in Abbildung 10 zu sehen.

Der verbleibende Teil des Workspace zeigt schließlich die Verwendung der Gaborfilter zur Klassifikation (oder hier auch zur Segmentierung) von Texturen (Abbildung 9). Das zerlegte Bild wird in reellwertige Vektoren übertragen und vorher festgelegte Teile, die bestimmte Texturen repräsentieren, werden daraus ausgeschnitten. Aus diesen (verschieden großen) Teilen werden Signaturdaten für den K-Nearest-Neighbor-Klassifikator erzeugt und zu einer Datei zusammengefasst. Aus den Signaturen und dem reellwertigen Bild erzeugt der Klassifikator ein Bild, in dem jedem Pixel seiner Klasse zugeordnet ist. Die entstehenden Resultate sind in Abbildung 6 zu sehen.

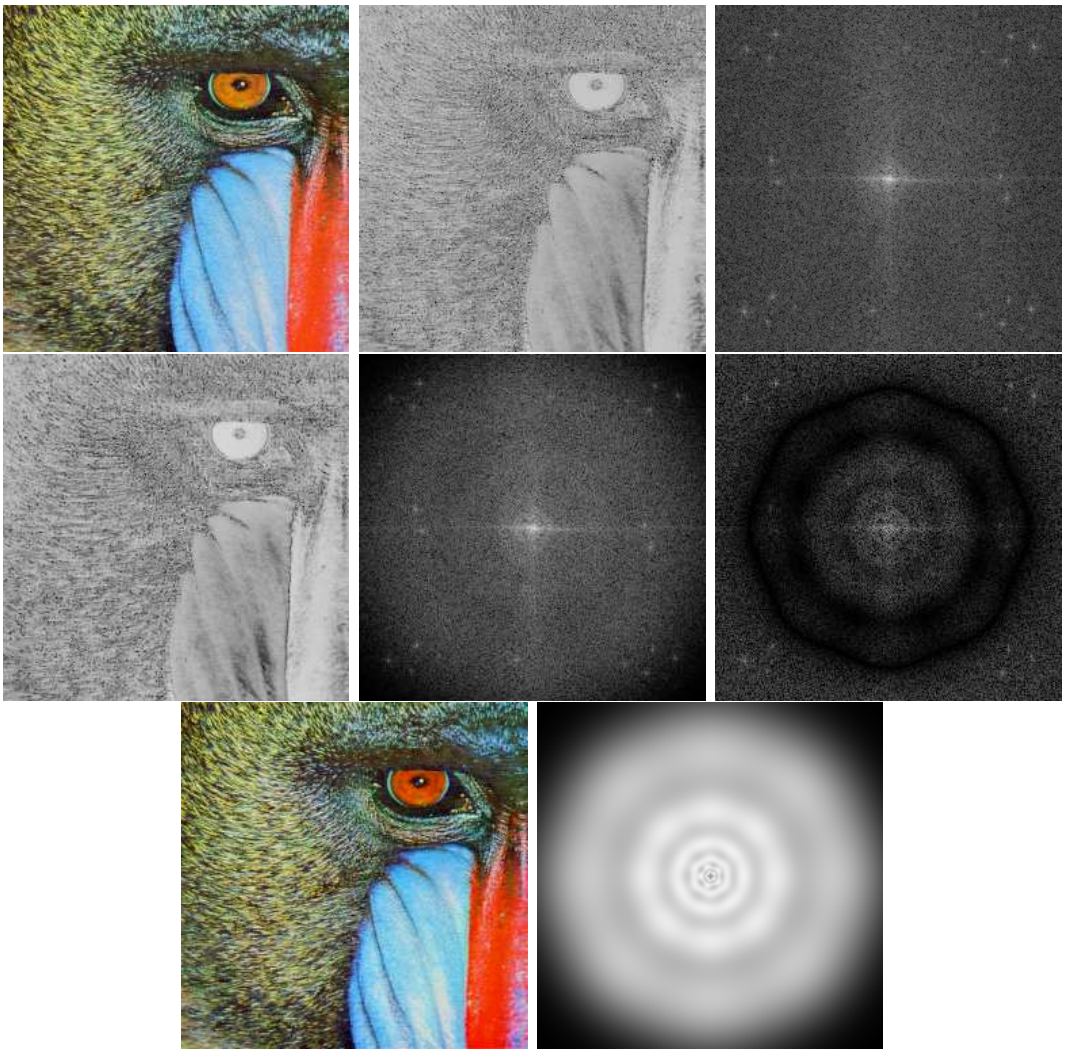

Abbildung 10: Bilder zum Verlauf der Dekomposition: (1) Original (s.o.) (2) HS-Anteil des Originals und (3) dessen Fouriertransformierte (4) HS-Komponente nach Rekomposition und (5) dessen Fouriertransformate (6) Differenz der Frequenzanteile (7) Rekomposition nach der Transformation in den RGB-Raum (8) Zum Ausgleich benutzte Summe der Filter im Frequenzraum - Nach Abschluss des Ausgleiches (nicht gezeigt) bestehen praktisch keine Unterschiede zum Originalbild mehr.

4 Klassifikationsversuch

In diesem Abschnitt wird der Versuch einer Klassifikation von Farbbildern von biologischen Texturen mit den implementierten Methoden beschrieben.

4.1 Aufgabenstellung

Gegeben waren Farbbilder von Baumrinden fünf verschiedener Arten von Bäumen, wie sie in Abbildung 11 zu sehen sind. Diese Bilder beinhalten einige Schwierigkeiten für die Klassifikation. So sind nicht nur die Rinden selber, sondern teilweise auch Hintergrund zu sehen. Ferner sind die Bilder nicht vollständig gleich skaliert und beleuchtet.

Abbildung 11: Beispielbilder aus dem Trainingssatz

Unter Verwendung von Trainingsdaten sollte ein k-Nearest-Neighbor-Klassifikator so trainiert werden, dass er die entsprechenden Testdaten den richtigen 5 Klassen zuordnen konnte.

4.2 Durchführung und Ergebnisse

Zur Durchführung der Klassifikation wurde jedes Trainingsbild mit den folgenden Schritten bearbeitet:

- Zunächst wurde aus dem Bild ein quadratischer Teil mit einer ganzzahligen Zweierpotenz als Kantenlänge ausgeschnitten, um die Anwendung der FFT zu erlauben.
- Im zweiten Schritt wurden die Bilder vom RGB-Farbraum in den HSV-Farbraum konvertiert.
- Dann wurde der Buntheitsanteil in die komplexwertige Darstellung überführt.
- Im nächsten Schritt wurde die Gaborzerlegung durchgeführt.

- Schließlich wurden die komplexen Werte der Gabortransfomation in reelle umgewandelt, um sie dem Klassifikator zugänglich zu machen.

Beim Training des Klassifikators wurden die erhaltenen Werte in wenige Cluster unterteilt und in einer Signaturdatei gespeichert. Die Beschränkung auf wenige Daten pro Trainingsbild wirkt sich wahrscheinlich ungünstig auf die Klassifikation aus, war aber zur Reduzierung der Datenmenge nötig.

In der Testphase wurden dann alle 68 Bilder pro Klasse durch den Klassifikator zugeordnet. Dies ergibt eine Zuordnung jedes Pixels zu einer der 5 Klassen. Dabei ergab die Benutzung der kreisförmig angeordneten Filter fast immer eine einheitliche Zuordnung aller Pixel eines Bildes zu einer Klasse. Die gitterförmig angeordneten Filter ergaben dagegen sehr unterschiedliche Zuordnungen. Dies könnte an einer stärkeren Betonung der niedrigen Frequenzen durch die erste Variante liegen. Trotzdem erreichte die zweite Variante deutlich schlechtere Ergebnisse in der Klassifikation (hier wurde die häufigst vorkommende Klasse eines Bildes als Ergebnis betrachtet). So wurden bei einer Verwendung von 8 Filtern pro Gitterseite bis auf ein Bild alle Bilder in die Klassen 3 und 4 eingeteilt.

Die Laufzeit des gesamten Verfahrens ist wesentlich bestimmt durch die Zeit, die der Klassifikator benötigt.

Das bisher beste erreichte Ergebnis ist eine Erkennungsrate von etwa 45%. Dies scheint eine sehr geringe Zahl. Unter Berücksichtigung der Schwere der Aufgabe und der großen Menge an Verbesserungsmöglichkeiten, die noch geprüft werden können (insbesondere im Hinblick auf andere berichtete gute Ergebnisse des Verfahrens) ist das Ergebnis jedoch akzeptabel. Dieses Ergebnis wurde erzielt mit nur 6 Trainingsbildern pro Klasse, kreisförmiger Filterverteilung mit 11 Phasen, Beginn bei der dritten inneren Komponente, einer Wichtung des Gleichstromanteils von 1.0 und ohne Berücksichtigung der komplexen Phase. Diese Parameter können dabei keinesfalls als optimal angesehen werden. Eine Optimierung erwies sich als sehr schwierig, da ein Durchlauf durch die Klassifikation auf einer SUN Ultra-Sparc Workstation etwa 24 Stunden dauert. Dabei entfällt ein großer Anteil der Rechenzeit auf den Klassifikator und die Signaturberechnung, weshalb nur 10 Vektoren pro Klasse als Basis der Klassifikation ausgewählt wurden, was eine weitere starke Einschränkung bedeutet. Auch aus dem folgenden Grund ist es erwägenswert, einen anderen Klassifikator im Zusammenspiel zu untersuchen. Von den 30 im Training benutzten Bildern wurden lediglich 14 der richtigen Klasse zugeordnet, was für einen Klassifikator, der auf dem Nearest-Neighbor-Prinzip beruht eigentlich atypisch ist. Die folgende Tabelle zeigt die Klassifikationen im einzelnen.

	1	2	3	4	5	$\sum$	%
1	9	3	23	29	4	68	13.2
2	1	49	0	18	0	68	72.1
3	9	0	36	23	0	68	52.9
4	2	8	7	51	0	68	75.0
5	7	16	0	38	7	68	10.3
$\sum$	28	76	66	159	11	340	
%	32.1	64.5	54.5	32.1	63.6		44.7

Hier ist eine deutliche Asymmetrie festzustellen, die mit einem anderen Klassifikator eventuell auch beseitigt werden könnte. Hierzu sind weitere Tests nötig. Es fallen ferner die guten Ergebnisse für die Klasse 2 auf, die durch die besondere Textur der Rinde der zugehörigen Baumart auch zu erwarten war.

Auf den gleichen Daten erreicht die Methode der Co-Occurence Matrizen [11] eine Erkennungsrate von etwa 61%. Nach den bisherigen Erfahrungen scheint es durchaus möglich, auch mit Gaborfilterung vergleichbare Werte zu erzielen.

5 Abschließende Bemerkungen

Das Verfahren ermöglicht die Integration von Farbinformation in die Texturanalyse und liefert in einigen Bereichen bessere Ergebnisse als eine Analyse ohne Berücksichtigung der Buntheit.

Dabei scheint die Hoffnung berechtigt, das Verfahren noch deutlich zu verbessern, insbesondere auch, weil für die Methode bei der Anwendung auf Grauwertbildern sehr gute Ergebnisse berichtet wurden [3, 8, 5, 1, 10, 7]. Es folgen einige Vorschläge zur weiteren Verbesserung des Verfahrens.

Eine starke Bedeutung für den Erfolg der Klassifikation hat der verwendete Klassifikator. Hier wäre zunächst eine Verbesserung des benutzten k-NN-Verfahrens zum Beispiel durch Skalierung der Achsen und Normierung der Merkmalsvektoren denkbar. Vielleicht ist es aber auch sinnvoll einen anderen Klassifikator zu verwenden, wie beispielsweise einen statistischen Ansatz mit Gauß'schen Mischverteilungen oder einen holografischen Klassifikator. Eine Frage in diesem Zusammenhang wäre, welche Art von Klassifikator am besten mit den durch die Gaborzerlegung erzeugten Daten harmoniert.

Eine weitere wichtige offene Frage ist die (annähernd) optimale Kombination der freien Parameter bei der Methode. Diese ist aufgrund der langen Laufzeit einer Klassifikation schwer zu bestimmen und abhängig von der Anwendung. Hier könnte eventuell eine (automatische) Einstellung zum Beispiel durch eine Art von Gradientenverfahren oder andere Optimierungsverfahren hilfreich sein.

Ferner ist es möglicherweise empfehlenswert, die einfache Berechnung der Phasendifferenzen durch eine (genauer zu definierende) Verschiebungsdifferenz zu ersetzen, die die Frequenz der Filter berücksichtigt. Wegen der unterschiedlichen Frequenzen der Filter ergibt die direkte Differenz nicht die Phasenverschiebung der einzelnen Frequenzanteile. Weiterhin könnte es je nach Aufgabenstellung interessant sein, den Effekt des „wrap-around“, der in der FT eingebracht wird, durch geeignete Verfahren zu vermindern, zum Beispiel durch Verwendung eines Fensters (Kaiser-Bessel-Fenster [9]).

In der konkreten Anwendung der Klassifikation von Baumrinden könnte der existierende Hintergrund in den Bildern durch Maskierung ausgeblendet werden. Außerdem könnte es sinnvoll sein, durch Vorverarbeitung der Bilder oder Einbeziehung entsprechender Methoden in die Verarbeitung selber eine Skalierungsinvarianz zu erreichen und eventuell Beleuchtungseffekte zu vermindern.

Die folgenden Absätze schildern Verfahren, die unterschiedliche Arbeitsweisen benutzen, von denen vielleicht einige für eine Erweiterung der Implementierung interessant sein könnten. Sie arbeiten jedoch ausschließlich mit Grauwertbildern.

In [1] wird ein auf Gaborfiltern basierendes Texturklassifizierungsverfahren beschrieben, dass als Beschreibungsvektoren die Quadrate der Antworten der Ga-

borkomponenten benutzt. Diese entsprechen der Energie der einzelnen Kanäle. Unter Verwendung einer komplexen Abstandsfunktion und einem nicht-parametrisierten statistischen Ansatz erreicht das beschriebene System bei der Klassifikation von regelmäßigen Texturen im günstigsten Fall eine Erkennunngsrate von 98,7%.

Das in [5] vorgestellte Klassifikationsystem erreicht durch die Anwendung der FFT auf die bei verschiedenen Orientierungen erhaltenen Antworten von isotropischen Gaborfiltern einer Frequenz zwei Ziele. Einerseits sind die erhaltenen Vektoren rotationsinvariant und andererseits kann die Länge der Vektoren durch den Verzicht auf die hohen Frequenzen der transformierten Daten reduziert werden. Deshalb könne vergleichsweise mehr Orientierungen untersucht werden. Gute Ergebnisse wurden hier mit Orientierungsdifferenzen zwischen 5° und 15° und einem euklidischen Klassifikator erzielt. Die Autoren erzielten in einer großen Datenbank eine Erkennungsrate von 94.4% und untersuchten auch den Einfluss von Gauß-Rauschen auf die Erkennung.

Eine weitere Idee, die bisher nicht umgesetzt wurde, ist die Verwendung von „Postfiltern“ zur Milderung der gegenseitigen Beeinflussung benachbarter Texturen in einem Bild, wie sie in [3] beschrieben wird. Diese Methode dient vor allem der Schaffung von klaren Texturgrenzen bei der Segmentierung von Bildern.

In [7] wird eine rotationsinvariante Texturklassifikation beschrieben, die auf Strukturmerkmalen in zwei Ebenen beruht. Grundlage bilden dabei Gabor-Wavelet-Analysen, die als weniger rechenintensiv und effektiver als Markov-Felder bezeichnet werden. Dabei werden Gaborfilter benutzt, die sich in zwei Eigenschaften wesentlich von den hier benutzten unterscheiden. Einerseits werden sie in einer Polarkoordinaten-Form verwendet, die bei Anordnung auf Kreisen eine bessere Abdeckung des Frequenzraums bietet und weniger Überlappungen verursacht. Andererseits wird eine „analytische“ Form der Gaußfunktionen benutzt, die bessere Eigenschaften in Bezug auf störende niederfrequente Anteile hat und rechnerische Vorteile bietet. Durch die Erhaltung von möglichst viel Information auf der ersten, lokalen Ebene (zur Segmentierung) ist der erste Schritt der Parametergewinnung umkehrbar. Dabei werden auch aus den Phasenwerten Informationen gewonnen, die einer Klassifikation zuträglich sind. Über stochastische Modellierungen und invariante, großflächigere Merkmale in der zweiten Ebene (zur Klassifizierung) werden dann entsprechende Beschreibungsvektoren für Texturen erzeugt. Die angegebenen Erkennungsraten sind 80.4% für alle Brodatz-Texturen, bzw. 96.8% für eine Auswahl. Dabei wird die Bedeutung des angemessenen Klassifikators betont.

In [10] werden Gabor-Wavelet-Merkmale zur inhaltsbasierten Bildsuche benutzt. Dazu wird ein adaptiver Algorithmus zur Filterauswahl vorgestellt. Weiterhin werden als Merkmale Mittelwerte und Standardabweichungen berechnet, unter der Annahme, dass die Texturen in dem für das Training benutzten Bereich

homogen sind. Eine Erweiterung (Lockerung) dieser Annahme würde für eine Klassifizierung mittels Mischverteilungen sprechen, nicht für Clustering. Es wird bemerkt, dass die Hinzunahme der Standardabweichung zur Klassifikation die Erkennungsrate erheblich verbessert. Ferner wird auf die Wichtigkeit des richtigen Abstandsmaßes hingewiesen (z.B. Euklidischer Abstand / Mahalanobis Abstand). Die Autoren behaupten außerdem, dass die Gabor-Merkmale in der betrachteten Aufgabenstellung im Vergleich mit drei anderen Merkmalsfamilien die besten Resultate erbrachten.

Literatur

[1] Robert Azencott, Jia-Ping Wang, and Laurent Younes. Texture classification using windowed fourier filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):148–153, February 1997.

[2] Martin J. Bastiaans. Gabor's signal expansion and degrees of freedom of a signal. *Optica Acta*, 29(9):1223–1229, 1982.

[3] Alan Conrad Bovik, Marianna Clark, and Wilson S. Geisler. Multichannel texture analysis using localized spatial filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(1):55–73, January 1990.

[4] Charles K. Chui. *An Introduction to Wavelets.* Academic Press, Boston, 1992.

[5] S. R. Fountain and T. N. Tan. Rotation invariant texture features from Gabor filters. *Lecture Notes in Computer Science*, 1352:57–64, 1997.

[6] H. Frey. *Digitale Bildverarbeitung in Farbräumen.* Dissertation, Technische Universität Ulm, Ulm, 1988.

[7] George M. Haley and B. S. Manjunath. Rotation-invariant texture classification using a complete space-frequency model. *IEEE Trans. Image Processing*, 8(2):255–269, February 1999.

[8] Anil K. Jain and Farshid Farrokhnia. Unsupervised texture segmentation using Gabor filters. *Pattern Recognition*, 24(12):1167–1186, 1991.

[9] T. Lehmann, W. Oberschelp, E. Pelikan, and R. Repges. *Bildverarbeitung für die Medizin: Grundlagen, Modelle, Methoden, Anwendungen.* Springer, Berlin, 1997.

[10] B. S. Manjunath and W. Y. Ma. Texture features for browsing and retrieval of image data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(8):837–842, August 1996.

[11] C. Palm, V. Metzler, B. Mohan, O. Dieker, T. Lehmann, and K. Spitzer. Co-Occurrence Matrizen zur Texturklassifikation in Vektorbildern. In *Bildverarbeitung für die Medizin, Proceedings des Heidelberger Workshops*, Seiten 367–371, Springer, Heidelberg, 1999.

[12] Alexander D. Poularikos, editor. *The Transforms and Applications Handbook.* CRC Press, 1996.