UNIVERSITÄT DORTMUND
FACHBEREICH INFORMATIK

Diplomarbeit

Klassifikation von Blütenpollen aus Mikroskopieaufnahmen mit Hilfe eines Fuzzy-Logik-Systems

Benjamin Schowe
14. Juli 2008

INTERNE BERICHTE
INTERNAL REPORTS

Diplomarbeit am
Fachbereich Informatik
der Universität Dortmund
Betreuer:
Prof. Dr. Bernd Reusch
Dr. Lars Hildebrand

Inhaltsverzeichnis

1. Einleitung

1.1. Aufgabenstellung

Viele Pflanzen vermehren sich durch Bestäubung mit Pollen. Dieser wird häufig durch die Luft transportiert. Es kann nun aus verschiedenen Gründen interessant sein, die Pollenkonzentration in der Luft zu kennen: Allergikern kann man vorhersagen, wann sie mit einer hohen Belastung zu rechnen haben. In der Landwirtschaft lassen sich zukünftige Ernten abschätzen. Oder es ließe sich feststellen, ob ein Feld mit genmanipulierten Pollen kontaminiert wurde.

Ziel dieser Diplomarbeit ist es, ein Erkennungssystem zu entwickeln, welches automatisch die Pollen in einer Probe aus 2D-Aufnahmen klassifiziert. Die Aufnahmen der Proben stammen aus herkömmlichen Lichtmikroskopen und die Klassifikation soll mit Hilfe eines Fuzzy-Logik-Systems geschehen.

Im Einzelnen sind hierzu sind folgende Schritte nötig:

Vorverarbeitung Die Bilder müssen für eine Segmentierung und Merkmalsgewinnung vorverarbeitet werden.

Segmentierung In der Segmentierung werden möglichst alle relevanten Objekte im Bild erkannt. Hierbei können auch andere Teilchen als Pollen erkannt werden.

Merkmalsgewinnung Für alle Objekte werden nun translations- und rotationsinvariante Merkmale berechnet.

Fuzzy-Zuordnung Als nächstes sollen allen Objekten anhand ihrer Merkmale Fuzzy-Variablen wie *groß*, *stark texturiert* oder *eher länglich* zugeordnet werden.

Klassifikation Im letzten Schritt sollen die Objekte mittels eines Fuzzy-Logik-Systems ihrer entsprechenden Klasse zugeordnet werden.

1.2. Stand der Technik

Bisher werden zur Bestimmung der Pollenkonzentration aus der Luft Proben entnommen. Momentan stehen zwei Verfahren zur Auswertung dieser Proben zur Verfügung. Zum einen können die Proben von Hand mikroskopiert und ausgezählt werden. Diese Methode ist jedoch personalintensiv und von schwankender Qualität. Zum anderen wurde im Zuge des Projekts *IMBUSS* [SHS+05] ein automatisches System zur Pollenzählung entwickelt. Dieses hat jedoch den Nachteil, dass sehr teure Gerätschaften erforderlich sind, da auch Volumendaten erfasst werden. Gemein haben beide Herangehensweisen, dass die Pollen

in einer Aufnahme erkannt, klassifiziert und gezählt werden. Auch muss zwischen Pollen und andere Schwebeteilchen beziehungsweise Staub oder Verunreinigungen unterschieden werden.

2. Grundlagen

2.1. Biologische Grundlage

Da die Motivation der Arbeit in der Erkennung von Pollenkörnern liegt, soll hier ein kurzer Überblick über die biologischen Grundlagen gegeben werden. Diese Kapitel basiert auf dem *Duden Basiswissen Schule Biologie* [SP04].

2.1.1. Fortpflanzung

Bei Samenpflanzen findet eine geschlechtliche Fortpflanzung statt. Dafür bilden diese Pflanzen *weibliche* und *männliche* Sporenzellen aus. Die männlichen Sporenzellen werden auch `Pollen` beziehungsweise in der Einzahl `Pollenkörner` genannt. Die weiblichen Sporenzellen bleiben in den Samenanlagen, den `Sporangien` eingeschlossen. Die männlichen Sporenzellen werden vom `Staubblatt`, der `Anthere`, abgesondert. Von dort aus werden sie zu den Samenanlagen transportiert, um dort auszukeimen und die weiblichen Sporenzellen zu befruchten. Dieser Transport kann auf drei Arten stattfinden:

Zoogamie Ein Tier oder Insekt nimmt den Pollen an den Antheren auf und lässt ihn an der Samenanlage einer anderen Pflanze zurück. Das bekannteste Beispiel hierfür ist die Biene.

Hydrogamie Die Pollenkörner werden vom Wasser von einer Pflanze zur Samenanlage einer anderen geschwemmt.

Anemogamie Windbestäubung. Die Pollenkörner werden vom Wind verteilt. Die Planzen sondern große Mengen an Pollenkörnern ab, um die Wahrscheinlichkeit zu erhöhen, dass eine Samenanlage auch von einem Pollenkorn erreicht wird.

Für diese Arbeit ist nur die Anemogamie relevant, da über die Luft verbreitete Pollenkörner erkannt werden sollen. Anemogamie tritt zum Beispiel bei Gräsern, Birke, Erle, Pappel, Hasel oder Nadelbäumen auf. Die Windbestäubung ist exemplarisch und schematisch in Abbildung 2.1 und 2.2 zu sehen.

2.1.2. Aufbau der Pollenkörner

Pollenkörner können in Aufbau, Form und Größe deutlich variieren, haben aber alle einige grundlegende Bestandteile. Die äußere widerstandsfähige Haut, das `Sporoderm`, besteht aus der außenliegenden `Exine` und der innenliegenden `Intine`. Die Exine ist recht hart ausgebildet und besteht aus `Sporopollenin`, kann aber auch Lücken aufweisen. Die Intine hingegen umgibt das Zellinnere vollständig, ist dafür aber weicher und besteht meist

Abbildung 2.1.: Anemogamie, Copyright ECARF - Europäische Stiftung für Allergieforschung

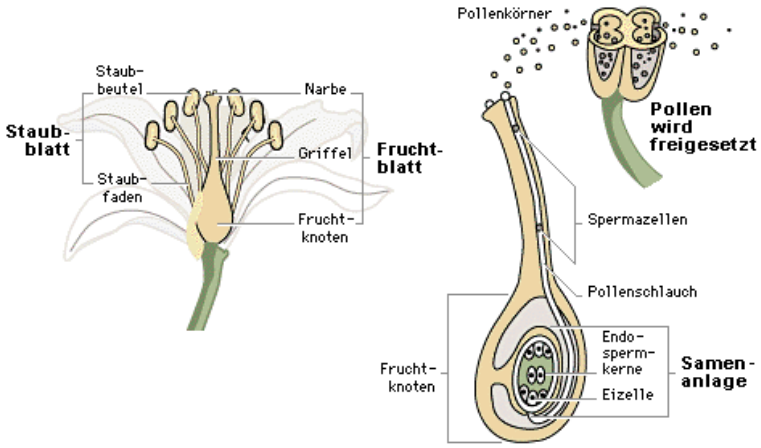


Abbildung 2.2.: Der Pollen wird vom Wind zu den Sporangien transportiert. Dort bildet ein Pollenkorn einen Pollenschlauch und befruchtet eine weibliche Eizelle. Quelle: MSN Encarta

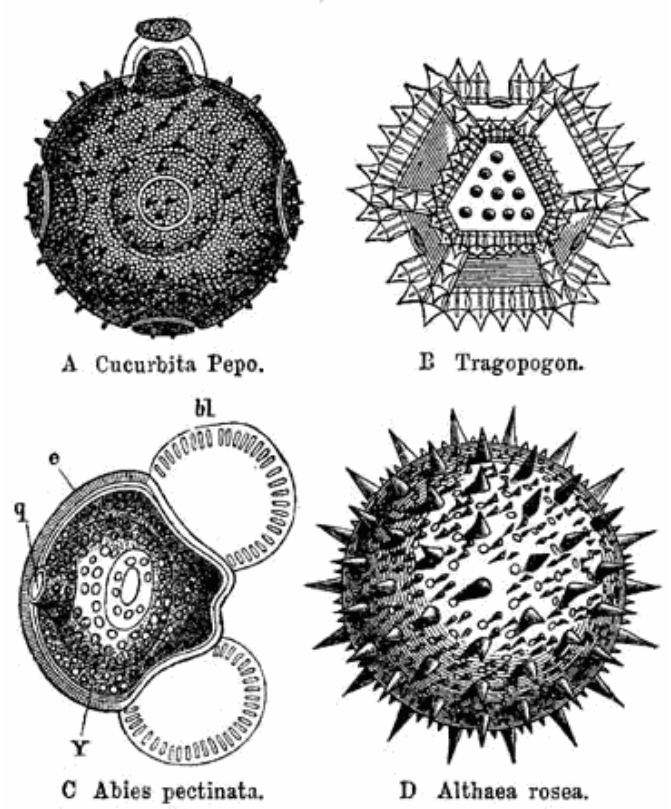


Abbildung 2.3.: Aufbau verschiedener Pollenkörner. Quelle: Meyers Konversation Lexikon 1888

aus Zellulose-Fibrillen. Beim Auskeimen bildet sich aus der Intine der Pollenschlauch, welcher dann durch den Griffel der bestäubten Blüte bis zur weiblichen Samenanlage wächst. Die Oberfläche der Exine kann aus Stäbchen, Keulen, Kegeln, Warzen oder ähnlichen Strukturen bestehen und wird dann als `intectat` bezeichnet. Bei `tectaten` Pollenkörnern sind die Säulchen an der Außenseite zu einer Schicht, dem `Tectum` verbunden. Ein Beispiel für ein intectates Pollenkorn ist in Abbildung 2.3D zu sehen. Damit der Pollenschlauch die Exine durchdringen kann, besitzen einige Sorten `Aperturen`, Keimöffnungen in der Exine. Nach der Anzahl der Keimöffnungen unterscheidet man `mono-` (ein), `tri-` (drei), `stephano-` (mehr als drei in Äquatorebene) und `panto-aperturate` (mehr als drei über die gesamte Oberfläche verbreitet) Pollen. Solche ohne Keimöffnungen werden als `inarperturat` bezeichnet.

2.1.3. Allergien

Laut ECARF [Fou06] werden bis 2010 ca. 50% der europäischen Bevölkerung von Allergien betroffen sein. Um so wichtiger ist es, Betroffenen die Möglichkeit zu geben, sich den auslösenden Stoffen zu entziehen. Hier spielt die Pollenflugvorhersage für Pollenallergi-

ker eine sehr wichtige Rolle. Patienten mit einer Pollenallergie reagieren überempfindlich auf die aus den Pollen freigesetzten Substanzen und bilden gegen diese für Nichtallergiker harmlosen Stoffe (Allergene) Abwehrstoffe. Beim Zusammentreffen der Allergene mit diesen Antikörpern an den Augen, in der Nase und in den Bronchien kann es zu einer Reizung der Schleimhäute mit folgenden Beschwerden kommen:

Augen Tränen, Juckreiz, Schwellung

Nase Niesreiz, Fließschnupfen, Verstopfte Nase

Bronchien Husten, Atemnot

Allgemein Kopfschmerzen, Schlafstörungen, Fieber, Abgeschlagenheit

Für Pollenallergiker ist es wichtig zu wissen, auf welche Pollen sie überempfindlich reagieren. Baum-, Kräuter- und Gräserpollen, wozu auch die Getreide wie der Roggen gehören, haben die größte Bedeutung.

2.1.4. Probeentnahme

Zur Bestimmung der Pollenmenge in der Luft müssen aus der Luft Proben etnommen werden. Dies geschieht mit sogenannten Pollenfallen, wie in Abbildung 2.4 zu sehen. Ein Luftstrom saugt alle Schwebeteilchen aus der Luft ein und „schießt“ sie auf eine rotierende Trommel. Diese ist mit einem mit Vaseline beschichteten Plastikstreifen umhüllt. In regelmäßigen Abständen werden die Trommel ausgetauscht und die eingefangenen Teilchen ausgewertet. Der Plastikstreifen kann direkt als Objektträger zum Mikroskopieren verwendet werden. Das Wirkprinzip wird in Abbildung 2.5 verdeutlicht.

Abbildung 2.4.: Pollenfalle auf dem Dach der Berliner Charite, Copyright ECARF-Europäische Stiftung für Allergieforschung

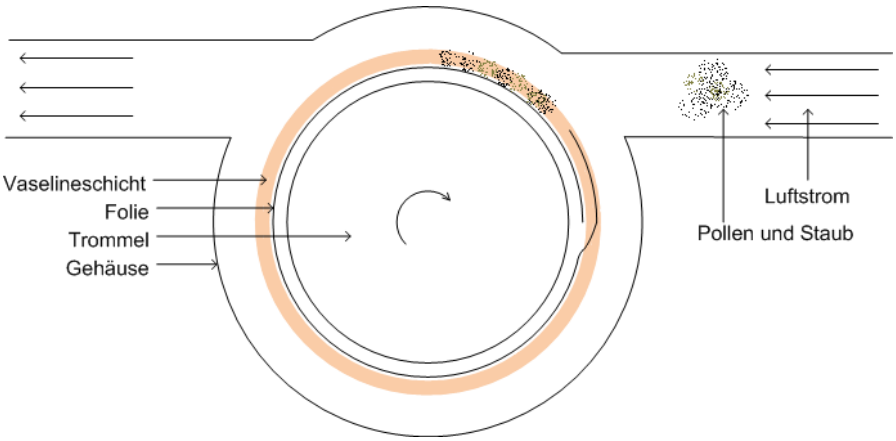


Abbildung 2.5.: Funktionsskizze einer Pollenfalle

2.2. Grundlagen der Bildverarbeitung

In diesem Kapitel sollen zunächst alle wesentlichen Verfahren zur Bildverarbeitung erläutert werden, die in dieser Arbeit verwendet wurden. Die Informationen zu diesem Kapitel stammen größtenteils aus dem Buch *Digital Image Processing* von Rafael C. Gonzalez und Richard E. Woods [GW01], bei Verfahren anderer Autoren werden diese explizit genannt.

2.2.1. Bild

Ein Graustufenbild ist ein endliches zweidimensionales Rechteckgitter mit Helligkeitsinformationen an jedem Kreuzungspunkt des Gitters. Die Helligkeitsinformation liegt in dem kontinuierlichen Intervall $[0, 1]$, wobei 0 die kleinst- und 1 die größtmögliche Helligkeit darstellt. Zur Darstellung im Rechner werden diskrete Werte benötigt. Bei einem gebräuchlichen 8-Bit-Graustufenbild stehen hierfür die Werte 0..255 zur Verfügung. Ein RGB-Farbbild wiederum besteht aus drei Graustufenbildern, welches jeweils die Helligkeiten für den roten, grünen und blauen Sehkanal angeben.

2.2.2. Konvertierung Farbbild zu Graustufenbild

Sind die Farbinformationen nicht relevant oder haben einen geringen Informationsgehalt, kann ein RGB-Farbbild in ein Graustufenbild umgewandelt werden. Die Intensität I an einem Punkt des Graustufenbildes ergibt sich dann aus der Formel $I = 0,3r+0,59g+0,11b$, wobei r, g und b die Helligkeitsinformationen der drei Farbkanäle am entsprechenden Punkt sind.

Die Faktoren $0,3$, $0,59$ und $0,11$ [Int96] sind hierbei nicht mathematisch begründet, sondern ergeben sich aus der Empfindlichkeit des menschlichen Auges. Im Auge finden sich drei Arten von Farbrezeptoren, die *L-*, *M-* und *S-Zapfen*. Jede Art nimmt einen bestimmten Wellenlängenbereich wahr. L-Zapfen (*long*) sind für langwelliges rötliches Licht zuständig, M-Zapfen (*middle*) sind bei mittleren Wellenlängen sensibel und nehmen eher grünes Licht wahr und S-Zapfen (*short*) reagieren auf kurzwelliges blaues Licht. Allerdings sind nur ca. 12% der für die Farbwahrnehmung im Auge verantwortlichen Zapfen kurzwellige S-Zapfen und die Wahrnehmungsbereiche der drei Arten überlappen sich im mittleren grünen Bereich. Daher wird die grüne Komponente am stärksten gewichtet.

Abbildung 2.6.: Konvertierung eines Farbbildes in ein Graustufenbild.

2.2.3. Faltung

Möchte man ein Bild mit einem Filter bearbeiten, der nicht allein vom Intensitätswert am jeweiligen Bildpunkt sondern auch von der Umgebung[1] des Bildpunktes abhängt, so benötigt man eine zweidimensionale Filterfunktion. Diese kann durch eine mathematische Funktionsvorschrift in der Form $h(s,t) = ...$ oder als ein Bild beliebiger Größe gegeben sein. Es wird dann für jeden Bildpunkt über die Produkte des Filters mit den Bildpunkten in der Umgebung aufsummiert.

Für ein Bild $f(x,y)$ der Größe $M \times N$ und einen Filter $h(s,t)$ der Größe $m \times n$ lautet die Gleichung

$$f(x,y) * h(s,t) = \sum_{s=-a}^{a} \sum_{s=-b}^{b} h(s,t) f(x+s, y+t) \tag{2.1}$$

mit $m = 2a + 1$ und $n = 2b + 1$, wobei $g(x,y)$ für alle $x = 0, 1, 2, ..., M-1$ und $y = 0, 1, 2, ..., N-1$ einzeln ausgewertet werden muss. Eventuell ist hier noch ein normalisierender Faktor voranzustellen, weil sonst das Ergebnis der Filterung den Wertebereich des Bildes überschreitet, wenn die Summe aller Koeffizienten des Filters nicht 1 ist. Normalisiert sieht die obige Gleichung dann wie folgt aus:

$$g(x,y) = \frac{\sum_{s=-a}^{a} \sum_{s=-b}^{b} h(s,t) f(x+s, y+t)}{\sum_{s=-a}^{a} \sum_{s=-b}^{b} h(s,t)}. \tag{2.2}$$

Ein sehr einfaches Beispiel für einen Glättungsfilter ist

1	1	1
1	1	1
1	1	1

beziehungsweise normalisiert

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

dessen Ergenis man in Abbildung 2.7 sehen kann.

Möchte man hingegen vertikale Kanten finden, bietet sich ein Filter der Form

1	0	-1
2	0	-2
1	0	-1

an. Dieser wird auch als Sobel-Operator bezeichnet [Sob70]. Am wichtigsten sind hier die Intensitätsunterschiede auf Höhe des betrachteten Bildpunktes, aber auch die Intensitätsunterschiede eine Zeile höher und tiefer gehen in das Ergebnis mit ein. Das Ergebnis dieses Filter sieht man in Abbildung 2.8.

[1] Die betrachtete Umgebung kann hier auch das gesamte Bild beinhalten.

Abbildung 2.7.: Filterung mit dem einfachen Glättungsoperator. Link: Originalbild (Quelle: ImageJ [Ras07a]), rechts: Filterungsergebnis.

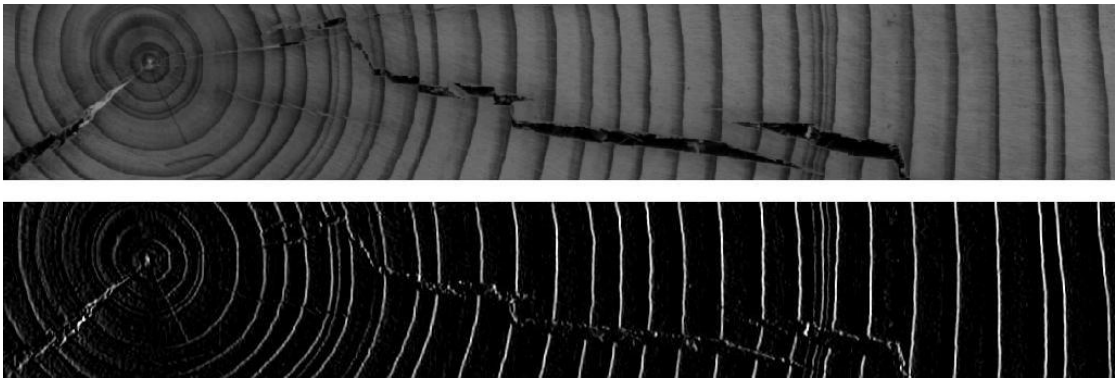


Abbildung 2.8.: Jahresringe eines Baumes, gefaltet mit einem vertikalen Kantenfilter. Oben: Originalbild (Quelle: ImageJ [Ras07a]), unten: Faltungsergebnis.

Es ist genauso möglich, einen Filter zu wählen, der so groß ist wie das Bild selber. Allerdings steigt dann die Rechenzeit enorm an, da für jeden einzelnen Bildpunkt $M \times N$ Additionen und Multiplikationen ausgeführt werden müssen. Für das gesamte Bild also je M^2N^2 Additionen und Multiplikationen. Ein Beispiel für solch einen Filter ist Abbildung 2.9.

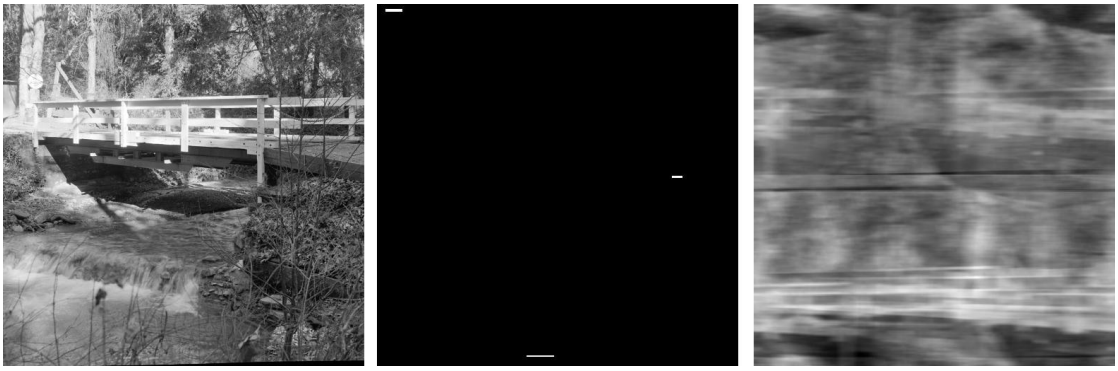


Abbildung 2.9.: Faltung mit einem Filter, der so groß ist, wie das Bild. Links: das Originalbild, Mitte: der Filter, rechts: das Ergebnis der Faltung

2.2.4. Histogramm

Ein Histogramm zählt für alle möglichen diskreten Helligkeitswerte eines Bildes wie oft diese vorkommen.

$$h(i) = \sum_{M-1}^{x=0} \sum_{N-1}^{y=0} \begin{cases} 1 \; wenn f(x,y) = i \\ 0 \; sonst \end{cases} \tag{2.3}$$

für $i = 0, 1, ..., L-1$ mit L als Anzahl der Graustufen und M und N als Höhe und Breite des Bildes.

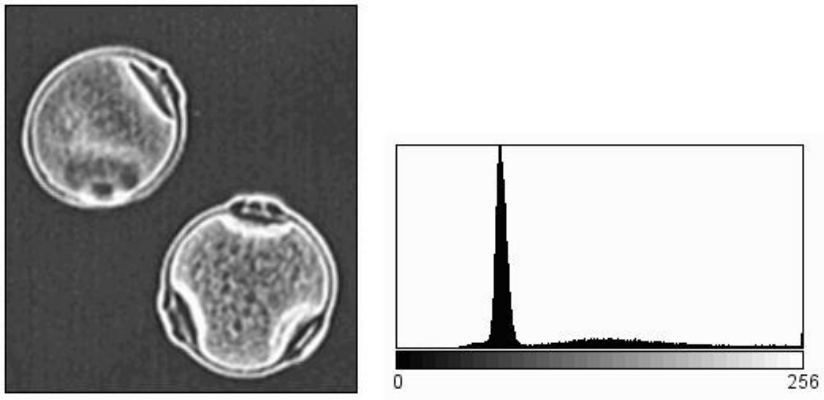


Abbildung 2.10.: Graustufenbild von zwei Pollenkörnern und dazugehöriges Histogramm

2.2.5. Fourier-Transformation

Nach *Fourier* lässt sich jede noch so komplizierte Funktion, wenn sie sich periodisch wiederholt, als Summe von Sinus- oder Cosinus-Funktionen unterschiedlicher Frequenz multipliziert mit unterschiedlichen Koeffizienten darstellen (*Fourier-Reihe*). Selbst nicht periodische Funktionen, deren Fläche unter ihrer Kurve nicht unendlich ist, lassen sich als Integral von Sinus- oder Cosinus-Funktionen multipliziert mit einer Gewichtsfunktion darstellen. Die Gewichtsfunktion entspricht hierbei den Koeffizienten aus der Fourier-Reihe.

Das Ergebnis der Fourier-Transformation liegt im sogenannten Frequenzbereich. Dieser besteht aus den Amplituden und Phasenverschiebungen der auftauchenden Frequenzen und im zweidimensionalen Fall auch deren Richtungen.

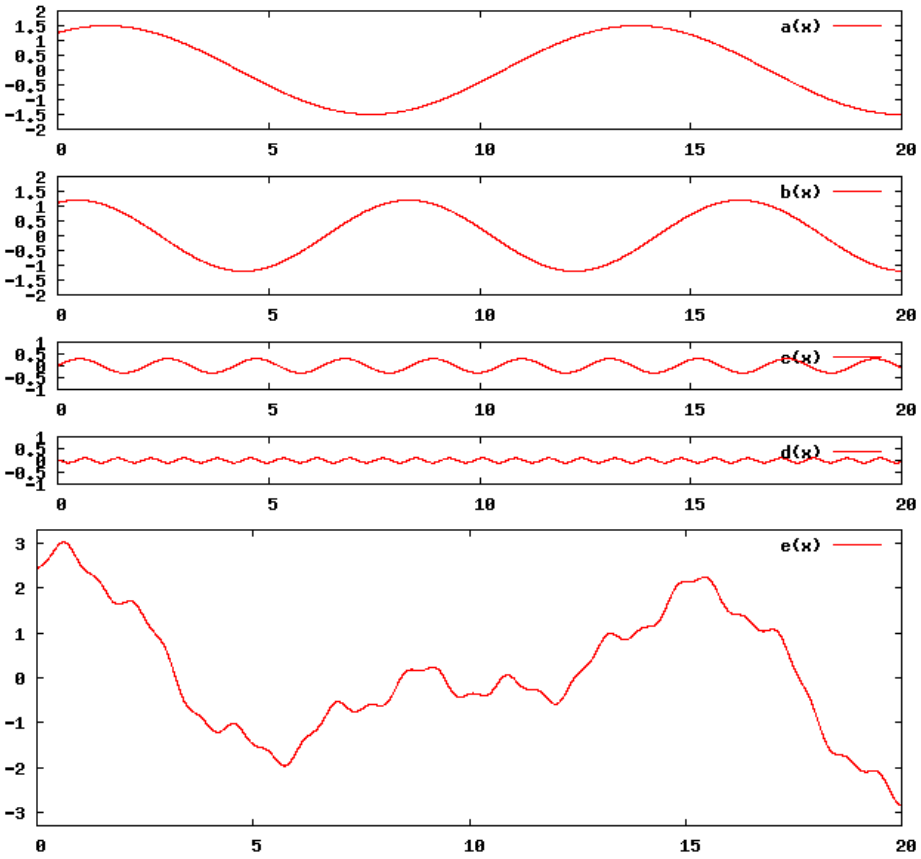


Abbildung 2.11.: Die unterste Funktion $e(x) = a(x) + b(x) + c(x) + d(x)$ ergibt sich als Summe der andere Funktionen mit $a(x) = 1.5 \cdot sin(0.5 \cdot x + 1)$, $b(x) = 1.2 \cdot sin(0.8 \cdot x + 1.2)$, $c(x) = 0.3 \cdot sin(3 \cdot x)$, $d(x) = 0.1 \cdot sin(8 \cdot x + 9)$

Die eindimensionale Fourier-Transformation

Die Fourier-Transformation $F(u)$ eines eindimensionalen Funktion $f(x)$ lautet

$$F(u) = \int_{-\infty}^{\infty} f(x) e^{-i2\pi ux} dx \tag{2.4}$$

mit dem komplexen Element $i = \sqrt{-1}$. Umgekehrt kann man aus dem Frequenzbereich bei gegebenem $F(u)$ mit Hilfe der inversen Fourier-Transformation $f(x)$ ermitteln:

$$f(x) = \int_{-\infty}^{\infty} F(u) e^{i2\pi ux} du \tag{2.5}$$

Wichtig zu bemerken ist, dass man verlustfrei von einem Bereich in den anderen und zurück transformieren kann.

Die zweidimensionale Form der Fourier-Transformation lautet

$$F(u,v) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x) e^{-i2\pi(ux+vy)} dx dy \tag{2.6}$$

mit ihrer Inversen

$$f(x,y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u,v) e^{i2\pi(ux+vy)} du dv \tag{2.7}$$

Diskrete Fourier-Transformation

In der Bildverabeitung nutzt man jedoch keine kontinuierlichen Signale, sondern arbeitet mit Rasterbildern, welche in gleichmäßigen Abständen abgetastet wurden und einen endlichen und diskreten Wertebereich haben. Dies erfordert eine diskrete Version der Fourier-Transformation (DFT):

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) e^{-i2\pi ux/M} \tag{2.8}$$

für $u = 0, 1, 2, ..., M-1$ und M als die Anzahl der abgetasteten (diskretisierten) Signalpunkte. Genau wie bei der kontinuierlichen Variante kann man auch bei der DFT von einem gegebenen $F(u)$ mit der inversen DFT $f(x)$ ermitteln:

$$f(x) = \sum_{u=0}^{M-1} F(u) e^{i2\pi ux/M} \tag{2.9}$$

für $x = 0, 1, 2, ..., M-1$

Wenn man die eulersche Identität

$$e^{i\phi} = cos(\phi) + i\ sin(\phi) \tag{2.10}$$

in Gleichung 2.9 einsetzt, kann man erkennen, dass jeder Funktionswert in jede Komponente $F(u)$ eingeht.

$$f(x) = \sum_{u=0}^{M-1} F(u)\ cos(2\pi ux/M) + i\ sin(2\pi ux/M) \tag{2.11}$$

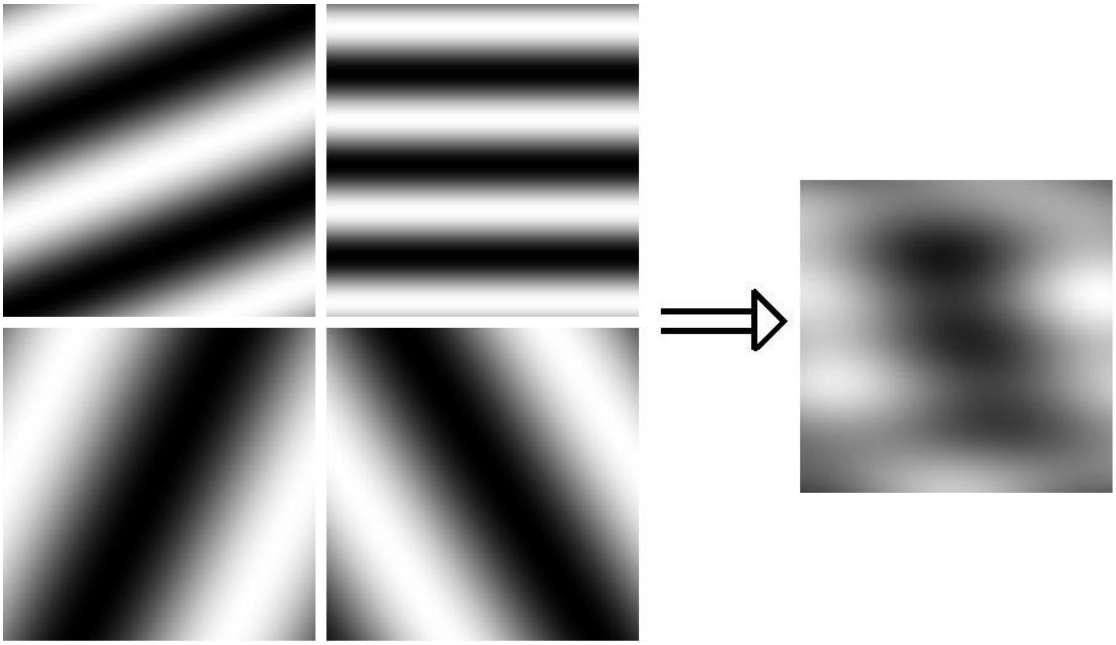


Abbildung 2.12.: Motivation der zweidimensionalen Fourier-Transformation: Das Bild rechts ist aus vier Sinuswellen unterschiedlicher Amplitude, Richtung und Phasenverschiebung zusammengesetzt.

Begriffe der Fourier-Transformation

An dem Faktor i in der Fourier-Transformation kann man sehen, dass es sich bei dem Ergebnis der Komponenten $F(u)$ (und auch $f(x)$) um komplexe Zahlen handelt. Überführt man $F(u)$ in Polarkoordinaten

$$F(u) = |F(u)|\, e^{-i\phi(u)} \tag{2.12}$$

nennt man auch den Betrag von $F(u)$

$$|F(u)| = \sqrt{R^2(u) + I^2(u)} \tag{2.13}$$

das `Amplitudenspektrum` und den Winkel

$$\phi(u) = tan^{-1}(\frac{I(u)}{R(u)}) \tag{2.14}$$

das `Phasenspektrum` von $F(u)$

Zweidimensionale diskrete Fourier-Transformation

Da ein Rasterbild eine zweidimensionale Ausdehnung besitzt, wird die zweidimensionale Variante der DFT benötigt, welche sich analog der eindimensionalen DFT aus der

zweidimensionalen kontinuierlichen FT ableiten lässt:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-i2\pi(ux/M+vy/N)} \tag{2.15}$$

für $u = 0, 1, 2, ..., M-1$ und $v = 0, 1, 2, ..., N-1$ für ein Bild mit den Dimensionen $M \times N$. Die Inverse ergibt sich zu

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{i2\pi(ux/M+vy/N)} \tag{2.16}$$

für $x = 0, 1, 2, ..., M-1$ und $y = 0, 1, 2, ..., N-1$. Genau wie bei der eindimensionalen DFT lassen sich die Begriffe `Amplitudenspektrum` 2.17 und `Phasenspektrum` 2.18 definieren als

$$|F(u,v)| = \sqrt{R^2(u,v) + I^2(u,v)} \tag{2.17}$$

$$\phi(u,v) = tan^{-1}(\frac{I(u,v)}{R(u,v)}) \tag{2.18}$$

Schnelle diskrete Fourier-Transformation

Die Berechnungsdauer der diskreten Fouriertransformation beträgt im eindimensionalen Fall $O(M^2)$ und im zweidimensionalen $O(M^2N^2)$ Operationen. Für ein 32×32 Pixel großes Bild macht das ungefähr 10^6 Operationen und bei einem 1024×1024 Pixel großen Bild ganze 10^{12} Operationen. Die schnelle Fourier-Transformation (im Folgenden FFT von Fast-Fourier-Transform) hingegen hat eine Laufzeit von $O(MlogM)$ beziehungsweise im zweidimensionalen $O(MNlogMN)$ Operationen. Für das 32×32-Bild wären das 25.600 statt 1.000.000 Operationen und für ein 1024×1024-Bild sogar 10^8 statt 10^{12} Operationen. Man kann sehen, dass der Geschwindigkeitsvorteil der FFT exponentiell anwächst. Erreicht wird dies durch die rekursiven Eigenschaften der Fourier-Transformation. Zunächst wird dies am Beispiel der eindimensionalen DFT erläutert. Hierzu schreibt man Gleichung 2.8 etwas abgekürzt als

$$F(u) = \frac{1}{M} \sum_{x=0}^{M-1} f(x) W_M^{ux} \tag{2.19}$$

mit

$$W_M = e^{-i2\pi/M} \tag{2.20}$$

Setzt man voraus, dass M eine Zweierpotenz ist

$$M = 2^n \tag{2.21}$$

kann man M auch schreiben als

$$M = 2K \tag{2.22}$$

Setzt man Gleichung 2.22 in Gleichung 2.19 ein, kann man den Term aufteilen

$$F(u) = \frac{1}{2K} \sum_{x=0}^{2K-1} f(x) W_{2K}^{ux} = \frac{1}{2}(\frac{1}{K} \sum_{x=0}^{K-1} f(2x) W_{2K}^{2ux} + \frac{1}{K} \sum_{x=0}^{K-1} f(2x+1) W_{2K}^{u(2x+1)}) \quad (2.23)$$

Aus Gleichung 2.20 ergibt sich, dass $W_{2K}^{2ux} = W_K^{ux}$. Dies setzt man in Gleichung 2.23 ein und erhält

$$F(u) = \frac{1}{2}(\frac{1}{K} \sum_{x=0}^{K-1} f(2x) W_K^{ux} + \frac{1}{K} \sum_{x=0}^{K-1} f(2x+1) W_K^{ux} W_{2K}^{u}) \quad (2.24)$$

Definiert man nun

$$F_{gerade}(u) = \frac{1}{K} \sum_{x=0}^{K-1} f(2x) W_K^{ux} \quad (2.25)$$

und

$$F_{ungerade}(u) = \frac{1}{K} \sum_{x=0}^{K-1} f(2x+1) W_K^{ux} \quad (2.26)$$

mit je $u = 0, 1, 2, ..., K-1$, verkürzt sich Gleichung 2.24 zu

$$F(u) = \frac{1}{2}(F_{gerade}(u) + F_{ungerade}(u) W_{2K}^{u}) \quad (2.27)$$

Mit $e^{i\phi} = cos\phi + i \cdot sin\phi$ ist $W_M^{u+M} = W_M^u$ und $W_{2M}^{u+M} = -W_{2M}^u$ wodurch sich mit den obigen Gleichungen ergibt, dass

$$F(u+K) = \frac{1}{2}(F_{gerade}(u) - F_{ungerade}(u) W_{2K}^{u}) \quad (2.28)$$

Zur Berechnung der $F(u)$ für $u = 0, 1, ..., \frac{M}{2} - 1$ wird Gleichung 2.27 ganz normal ausgewertet. Für $u = \frac{M}{2}, ..., M-1 = 0+K, 1+K, ..., \frac{M}{2} - 1 + K$ wird die Gleichung 2.28 ausgewertet, wobei die Werte für F_{gerade} und $F_{ungerade}$ jetzt schon bekannt sind und nicht mehr errechnet werden müssen. Bei genauerer Betrachtung ergibt sich für die Zahl der Multiplikationen $m(M)$ abhängig von der Anzahl der Werte

$$m(n) = \frac{1}{2} M \; log_2 M \quad (2.29)$$

und für die Anzahl der Additionen $a(M)$

$$a(n) = M \; log_2 M \quad (2.30)$$

. Somit hat der FFT-Algorithmus eine Laufzeit von $O(M \; log_2 M)$ bei M Punkten.

Um allerdings die FFT für Bilder nutzen zu können, bedarf es einer zweidimensionalen Variante. An dieser Stelle wird die `Separierbarkeit` der Fourier-Transformation benötigt:

$$F(u,v) = \frac{1}{M} \sum_{x=0}^{M-1} e^{-i2\pi ux/M} \frac{1}{N} \sum_{y=0}^{N-1} f(x,y) e^{-i2\pi vy/N} = \frac{1}{M} \sum_{x=0}^{M-1} F(x,v) e^{-i2\pi ux/M} \quad (2.31)$$

mit

$$F(x,v) = \frac{1}{N} \sum_{y=0}^{N-1} f(x,y) e^{-i2\pi vy/N} \tag{2.32}$$

Dies impliziert, dass man die 2D-FFT eines Bildes berechnen kann, indem man zuerst die FFT aller Zeilen und danach die FFT über alle resultierenden Spalten berechnet.

$$f(x,y) \overset{Zeilen}{\rightarrow} F(x,v) \overset{Spalten}{\rightarrow} F(u,v) \tag{2.33}$$

Genauso denkbar ist aber auch die Reihenfolge

$$f(x,y) \overset{Spalten}{\rightarrow} F(u,y) \overset{Zeilen}{\rightarrow} F(u,v) \tag{2.34}$$

2.2.6. Filtern im Frequenzbereich

Eine mögliche Anwendung der Fourier-Transformation besteht darin, die einzelnen Frequenzen eines Bildes zu bearbeiten. Zuerst wird das Bild in den Frequenzbereich transfomiert. Dort kann dann jede Frequenz einzeln manipuliert werden. Zum Schluss transformiert man das Bild wieder zurück in den Ortsbereich.

Zum Beispiel lässt sich auf diese Weise ein einfacher Tiefpassfilter erzeugen, in dem man im Frequenzbereich die Amplitude aller Frequenzen auf 0 setzt, die höher sind als ein gegebener Schwellwert. Im Beispiel in Abbildung 2.13 wurde dies bei allen Frequenzen durchgeführt, die auf eine Achse projeziert länger als 16 sind.

2.2.7. Faltungssatz

Die in Abschnitt 2.2.3 besprochene Faltung von Bildern im Ortsbereich benötigt quadratische Laufzeit. Da dies ab einer gewissen Größe von Bild und Filter nicht mehr praktikabel ist, kommt eine Eigenschaft der DFT hier sehr gelegen.

$$f(x,y) * h(x,y) \Leftrightarrow F(u,v)H(u,v) \tag{2.35}$$

$$f(x,y)h(x,y) \Leftrightarrow F(u,v) * H(u,v) \tag{2.36}$$

Dies bedeutet, dass man eine Faltung im Ortsbereich durch eine Multiplikation im Frequenzbereich ausdrücken kann. Betrachtet man die Laufzeiten, so benötigt man für $n = M \times N$ Bildpunkte mit der FFT $O(n\ log(n))$ Operationen für die Transformation des Bildes und des Filters in den Frequenzbereich. Der Multiplikationschritt benötigt lediglich $O(n)$ Operationen und die Rücktransformation des Bildes wieder $O(n\ log(n))$ Operationen. Insgesamt hat diese Art der Faltung eine Laufzeit von $O(n\ log(n))$ im Gegensatz $O(n^2)$ bei der normalen Faltung.

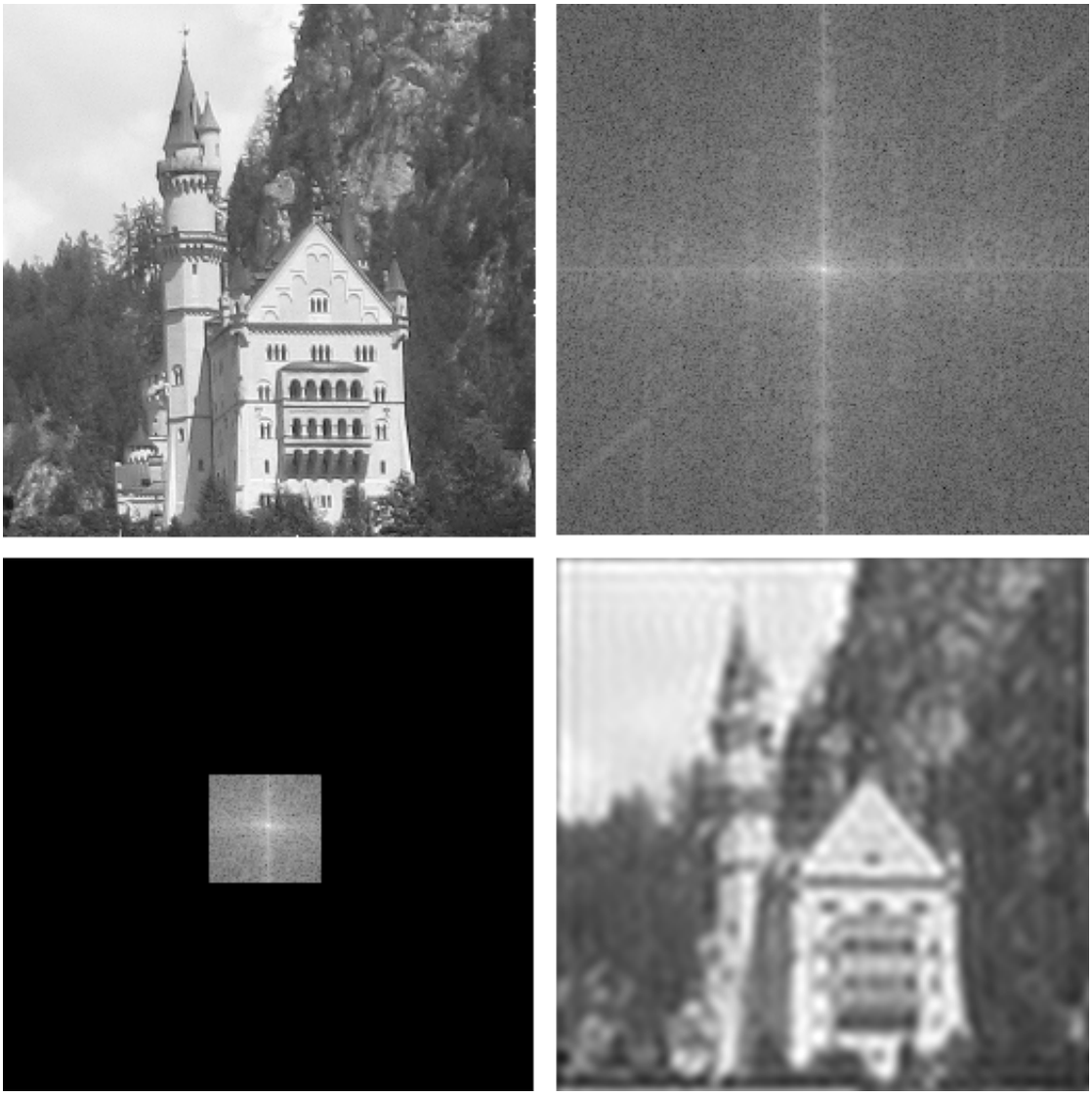


Abbildung 2.13.: Filtern im Frequenzbereich. Links oben: Originalbild, rechts oben: zentriertes Amplitudenspektrum, links unten: hohe Frequenzen auf 0 gesetzt, rechts unten: Rücktransformation

Padding

Um Artefakte und Überschattung bei der Faltung durch die Periodizität der DFT zu vermeiden, müssen die zu faltenden Bilder zuvor erweitert werden. Dieser Vorgang wird auch als **Padding** bezeichnet. Dabei werden beide Bilder auf die Größe mindestens $P \times Q$ erweitert, mit $P = a + b - 1$ und $Q = c + d - 1$. Hierbei stehen a und c für die Breite und Höhe des ersten b und d für die Breite und Höhe des zweiten Bildes. Zur ausführlichen Begründung siehe [GW01], Seite 204ff.

2.2.8. Maxima finden

Diese Prozedur sucht in einem Bild Punkte mit maximaler Intensität und orientiert sich an dem Verfahren von Michael Meier in [Ras07a].

Zuerst werden alle Bildpunkte herausgesucht, welche lokale Maxima sind, das heißt, für die gilt, dass die Helligkeitswerte aller ihrer direkten Nachbarn kleiner sind. Diese werden als mögliche Maxima markiert und absteigend nach ihrer Intensität sortiert. Im nächsten Schritt werden sie der Reihe nach abgearbeitet.

Von jedem möglichen Maximum ausgehend werden alle Bildpunkte als besucht markiert, deren Intensität innerhalb einer vorgegebenen Toleranz bezüglich des Ausgangspunktes liegen. Dies geschieht, indem man alle Nachbarn eines möglichen Maximums in eine Warteschlange (Queue) einfügt. Nacheinander wird jeder Punkt in der Warteschlange betrachtet. Wurde der Punkt noch nicht als besucht markiert und liegt sein Intensitätswert innerhalb der Toleranz, so werden alle seine direkten Nachbarn in die Warteschlange eingefügt. Der Punkt wird als besucht markiert. Stößt man jedoch auf einen Punkt, welcher bereits vorher markiert wurde, so wird das aktuell betrachtete mögliche Maximum verworfen, da dieser Punkt im Toleranzbereich eine größeren möglichen Maximums liegt.

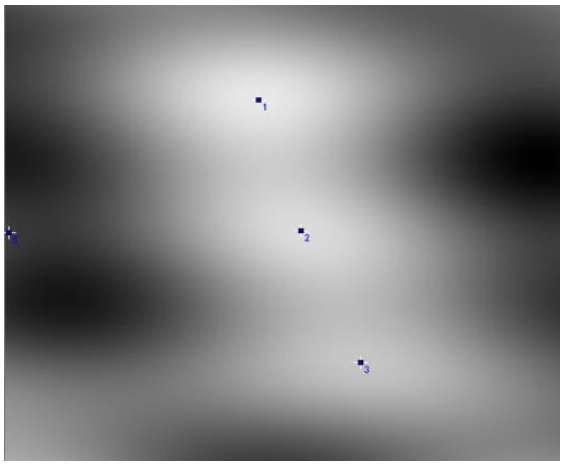


Abbildung 2.14.: Gefundene Maxima mit einer Toleranz von fünf Graustufen

2.2.9. Mixture-Modeling

Angenommen, ein Bild besteht aus einem Hintergrund ungefähr gleicher geringer Intensität und Objekten auf diesem Hintergrund, welche auch alle ungefähr die gleiche hohe Intensität haben. Dann besitzt das Histogramm des Bildes zwei markante Ausschläge (zwei Berge). Dieses Verfahren liefert einen Schwellwert, so dass möglichst alle Bildpunkte mit Intensitäten kleiner als der Schwellwert zum Hintergrund und alle anderen Bildpunkte zum Vordergrund, zu den Objekten, gehören.

Es wird versucht, die Parameter μ für den Erwartungswert und σ für die Standardabweichung zweier Gauß-Glocken so zu wählen, dass diese Gauß-Glocken die beiden Ausschläge im Histogramm möglichst gut approximieren. Die Formel der Gaußverteilung lautet

$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} exp(-\frac{1}{2}(\frac{x-\mu}{\sigma})^2) \tag{2.37}$$

Der ausgegebene Schwellwert liegt dann am (mittleren) Schnittpunkt der beiden Gaußglocken. Dies kann man an dem Histogramm in Abbildung 2.15 erkennen. Eine Beispielanwendung ist in Abbildung 2.16 zu sehen.

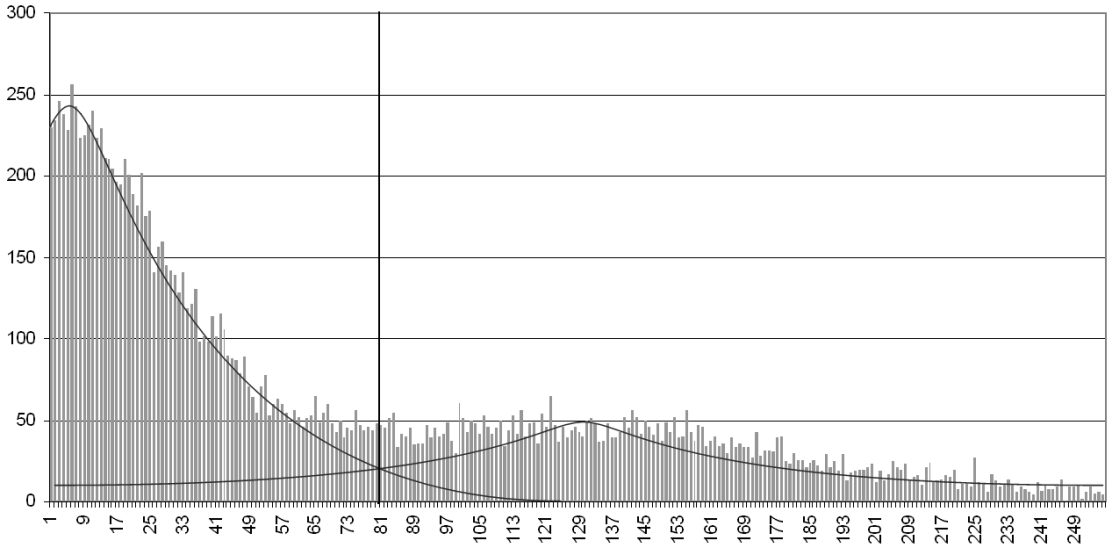


Abbildung 2.15.: Der Schnittpunkt der gefundenen Gaußkurven liegt bei 80.

Abbildung 2.16.: Graustufenbild von Gingkoblättern (Quelle: www.hauenstein-rafz.ch), links das Original, rechts nachdem ein durch Mixture-Modeling ermittelter Schwellwert gesetzt wurde. Alle Pixel, die heller sind als der Schwellwert, wurden auf 1,0 gesetzt, die anderen auf 0,0.

2.2.10. Kantenfilter nach Canny

Der Kantenfilter (auch Gradientenfilter) nach Canny [Can86] arbeitet aus einem Bild klare Kanten heraus, ist jedoch deutlich effektiver als die bloße Faltung mit einem einfachen Kantenoperator wie zum Beispiel dem Sobel-Operator [Sob70]. Es wird nämlich versucht, den Weg der Kanten „entlangzulaufen“, um eventuelle Brüche zu überbrücken.

Cannys Algorithmus gliedert sich in sechs Schritte:

1. **Glättung** Das Bild wird mit einem Gauß-Tiefpassfilter geglättet.

2. **Gradientenbilder** Das Bild wird einmal mit einen vertikalen $\begin{matrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{matrix}$ und einem horizontalen $\begin{matrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{matrix}$ Sobel-Operator gefaltet. Es entstehen die beiden Gradientenbilder G_x und G_y, welche die erste Ableitung des Bildes in x- und y-Richtung widerspiegeln.

3. **Betrag** Ein weiteres Gradientenbild G entsteht aus den Längen der einzelnen Gradientenvektoren, welche sich aus $\binom{G_x}{G_y}$ ergeben. $G = \sqrt{G_x^2 + G_y^2}$

4. **Richtungen der Kanten** Es wird für jeden Bildpunkt der Richtungswinkel θ der Kante abgespeichert. Dieser ergibt sich aus dem Arcustangens der Gradienten in x- und y-Richtung.

$$\theta(x, y) = arctan(G_x(x, y), G_y(x, y)) \tag{2.38}$$

5. **Richtungen diskretisieren** Der Richtungswinkel wird in vier Intervalle aufgeteilt und θ der entsprechenden Hauptrichtung zugewiesen:

$0°$	$0° \leq \theta < 22,5° \vee 157,5° \leq \theta < 180°$
$45°$	$22,5° \leq \theta < 67,5°$
$90°$	$67,5° \leq \theta < 112,5°$
$135°$	$112,5° \leq \theta < 157,5°$

(2.39)

6. **Unterdrückung** Für alle Punkte in $G(x, y)$ wird überprüft, ob sie in Richtung ihres Gradientenwinkels lokale Maxima sind. Falls dies nicht der Fall ist, wird $G(x, y)$ an dieser Stelle auf 0 gesetzt.

7. **Hysterese** Es werden zwei Schwellwerte $T_1 > T_2$ festgelegt. Alle Bildpunkte, deren Intensität größer als T_1 ist, werden automatisch als Kanten in das Ergebnisbild übernommen. Ausgehend von diesen Punkten läuft man senkrecht zur Gradientenrichtung die Kante entlang und fügt zu den Kanten noch die Punkte hinzu, deren Intensitätswert größer als T_2 ist.

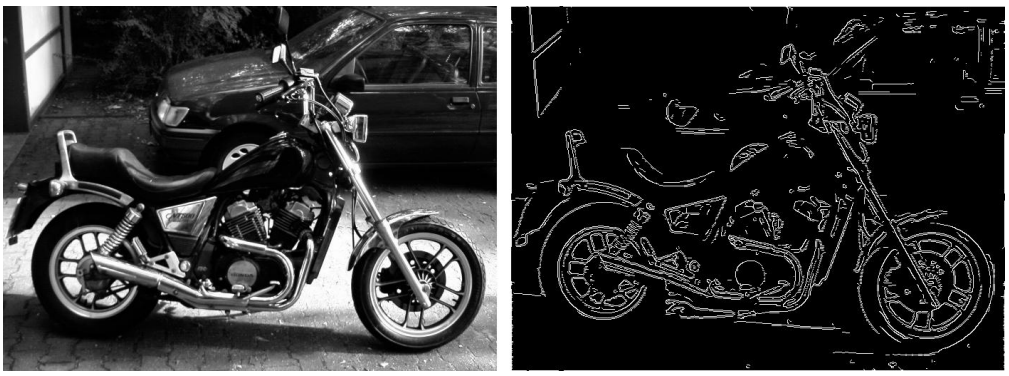


Abbildung 2.17.: Kantenfilter nach Canny mit den Grenzwerten $T_1 = 0,6$ und $T_2 = 0,2$

2.2.11. Medianfilter

Ein Medianfilter ist ein Verfahren zur Glättung von Bildern. Jeder Bildpunkt wird einzeln betrachtet. Der neue Intensitätswert für jeden Bildpunkt erhält man folgendermaßen: Alle Intensitätswerte der Bildpunkte in einer vorgegebenen Umgebung um den Punkt werden in eine Liste geschrieben sortiert. Der Wert in der Mitte der sortierten Liste ist die neue Intensität.

Als Umgebung wird üblicherweise ein Kreis mit einem wählbaren Radius benutzt. Die Größe des Kreises beeinflusst hierbei die Stärke des Filters. Ein wichtiger Vorteil des Medianfilter ist, dass Kanten im Bild erhalten bleiben und nicht verwischt werden.

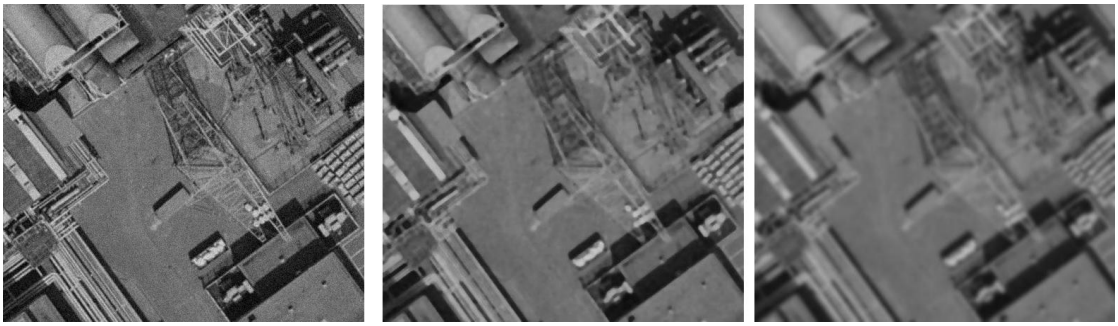


Abbildung 2.18.: Links das verrauschte Originalbild, in der Mitte nach einem Medianfilter mit Radius 3, rechts nach einem Gauß-Tiefpass-Filter mit Radius 3. Quelle des Originalbildes: GoogleEarth.

2.2.12. Snakes

Snakes sind ein Verfahren nach Kass, Witkin und Terzopolus [KWT88] um die Konturen von Segmenten in vorverarbeiteten Bildern klarer herauszuarbeiten. Die Kontur soll dabei zwei Ansprüchen genügen. Zum einen soll sie möglichst glatt und differenzierbar sein und zum anderen sich möglichst gut an die echten Konturen im Bild anpassen. Beides ist über Parameter steuerbar.

Bei der hier betrachteten energieminimierenden Variante notiert man die Kontur als Kurve in Parameterdarstellung $v(s) = (x(s), y(s))$ mit $s = [0, 1]$. Es wird versucht, das Energiefunktional $E(v) = S(v) + P(v)$ zu minimieren, welches von der Form $S(v)$ der Kruve und ihrer Lage $P(v)$ im Bild abhängt.

$$S(v) = \int_0^1 \alpha(s)v'(s)^2 + \beta(s)v''(s)^2 ds \tag{2.40}$$

$S(v)$ lässt sich über die Parameter $\alpha(s)$ für die Spannung der Kurve und $\beta(s)$ für die Steifigkeit der Kurve steuern, siehe Gleichung 2.40. Meist wählt man der Einfachheit halber $\alpha(s)$ und $\beta(s)$ über die Kurve konstant, so dass $\alpha(s) = \alpha$ und $\beta(s) = \beta$. Die Funktion für P wird anwendungsabhängig gewählt, sodass P zum Beispiel die Intensitätwerte des Gradientenbildes

$$P(v) = -c \int_0^1 \nabla(v(s))ds \tag{2.41}$$

$\nabla(x, y)$ steht für den Intensitätswert des Gradientenbildes am Punkt (x, y) und c steuert, wie stark die Lage im Bild das Energiefunktional beeinflusst. Das negative Vorzeichen resultiert daraus, dass ein hoher Intensitätswert im Gradientenbild eine gute Lage darstellt und somit geringe Energie benötigt. Eine minimale Kontur $v(s)$ erfüllt die Euler-Lagrange-Gleichung

$$\alpha v''(s) + \beta v''''(s) + \frac{\delta P(v(s))}{\delta s} = 0 \tag{2.42}$$

Allerdings müssen noch einige Randbedingungen vorgegeben werden, wie zum Beispiel Ort und Ableitung des ersten Punktes und des letzten Punktes. Nun wird versucht,

$E(v)$ iterativ zu minimieren, indem an diskreten Stützstellen von s die Kurve immer ein wenig verschoben wird. Die diskreten Stützstellen werden festgelegt als $v_0, ..., v_{n+2}$, mit $v_0 = v_{n+1}$ und $v_1 = v_{n+2}$ und $v_i = v(s_i)$ mit $s = i/n$, so dass die Kurve sich in der Nähe der Zielkurve befindet. Als Nächstes stellt man für diese Punkte die Euler-Lagrange-Gleichung 2.42 und erhält $2(n+1)$ Gleichungen mit $2(n+1)$ Unbekannten.

$$\begin{aligned}
&\alpha \cdot (v_i - v_{i-1}) \\
&+\alpha \cdot (v_{i+1} - v_i) \\
&+\beta \cdot (v_{i-2} - 2v_{i-1} + v_i) \\
&-2\beta \cdot (v_{i-1} - 2v_i + v_{i+1}) \\
&+\beta \cdot (v_i - 2v_{i+1} + v_{i+2}) \\
&+(P(x_{i+1}, y_i) - P(x_i, y_i), P(x_i, y_{i+1}) - P(x_i, y_i))^* \\
&= 0
\end{aligned} \tag{2.43}$$

$n+1$ für die x- und $n+1$ für die y-Komponente. Da es sich um eine Diskretisierung handelt, werden die Ableitungen durch partielle Differenzen ersetzt. Schreibt man Gleichung 2.43 als Gleichungssystem in Matrixschreibweise, so erhält man

$$\begin{aligned}
Ax + P_x(x, y) = 0 \\
Ay + P_y(x, y) = 0
\end{aligned} \tag{2.44}$$

mit $x = (x_0, ..., x_n)^*, y = (y_0, ..., y_n)^*$ und A eine diagonale Bandmatrix mit fünf Einträgen pro Zeile.

Dieses Gleichungssystem wird nun iterativ gelöst und damit die Kurve immer ein Stück verfeinert.

$$\begin{aligned}
x_t = (A + \gamma I)^{-1}(\gamma x_{t-1} - P_x(x_{t-1}, y_{t-1})) \\
y_t = (A + \gamma I)^{-1}(\gamma y_{t-1} - P_y(x_{t-1}, y_{t-1}))
\end{aligned} \tag{2.45}$$

γ gibt die Schrittweite an und t zählt die Iterationen.

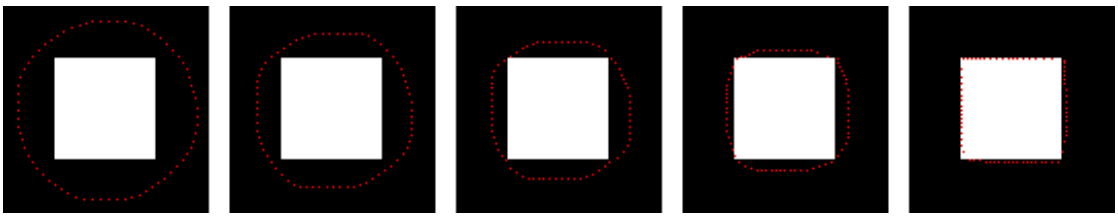


Abbildung 2.19.: Die Snake zieht sich von einer vorgegebenen Form immer weiter um das Objekt zusammen. Zustand nach 0, 6, 10, 14 und 18 Iterationen

2.2.13. Mustererkennung

Um ein Objekt zu klassifizieren, müssen sowohl dem Objekt als auch den verschiedenen Klassen beschreibende Merkmale zugewiesen werden. Möchte man einem Auto seinen Marken- und Modellnamen zuordnen, sind Merkmal wie Form, Leistung oder Höchstgeschwindigkeit. Bei Objekten in Bildern sind andere Merkmale interessant, welche in

ihrer Komplexität von trivial bis aufwändig variieren. In dieser Arbeit wurden zum einen einfache Formmerkmale betrachtet aber auch Eigenschaften der Textur, welche deutlich komplexer sind. Die Aussagekraft der benutzten Merkmale soll später in dieser Arbeit analysiert werden.

Objektmerkmale

Ein zweidimensionales Objekt besitzt eine Kontur, aus der sich einige Maßzahlen ableiten lassen. Siehe hierzu [Mül06].

Fläche Die Anzahl der Pixel, welche vom Objekt überdeckt werden. Diese kann durch einfaches Abzählen ermittelt werden.

Umfang Die Anzahl der Pixel, die auf dem Rand des Objekts liegen. Auch diese kann durch einfaches Abzählen der Randpunkte ermittelt werden. Ist der Rand eines Objekts als Polygon $(P_0, P_1, ..., P_{n-1}, P_n), P_n = P_0$ gegeben, ist es auch möglich, die Abstände aller benachbarten Punkte zu addieren:

$$U = \sum_{i=1}^{n-1} |P_i - P_{i+1}| \tag{2.46}$$

Verhältnis Fläche zu Umfang Das Verhältnis A/U gibt an, wie kompakt das Objekt ist.

Kreisabweichung Das Verhältnis $\frac{U}{\sqrt{\frac{A}{P_i}}}$ gibt an, wie stark der Umfang vom dem Unfang eines Kreises mit der Grundfläche A abweicht. Ist das Objekt ein Kreis geht der Wert gegen 1, ist der Rand sehr ausgefranst oder das Objekt eher länglich wird der Wert größer.

Mittlere 1-Krümmung Man mittelt über alle Punkte die Winkeldifferenzen zwischen den Geraden-Paaren, die durch die Nachbarpunkte induziert werden: (P_{i-1}, P_i), (P_i, P_{i+1}). Die mittlere Krümmung ergibt sich dann zu

$$k = \frac{1}{n} \sum_{i=1}^{n} cos^{-1}\left(\frac{(P_{i-1} - P_i) \cdot (P_i - P_{i+1})}{|P_{i-1} - P_i| \cdot |P_i - P_{i+1}|}\right) \tag{2.47}$$

Texturmerkmale

Neben seiner Form besitzt ein Objekt in einem zweidimensionalen Bild auch eine Textur. Textur ergibt sich aus der räumlichen und statistischen Verteilung der Grauwerte in einem Bild. Mit sprachlichen Mitteln beschrieben kann eine Textur grob oder fein, glatt oder rau, gleichmäßig oder zufällig, wellig oder eben sein. Die Schwierigkeit besteht darin, diese Beschreibung numerisch greifbar zu machen.

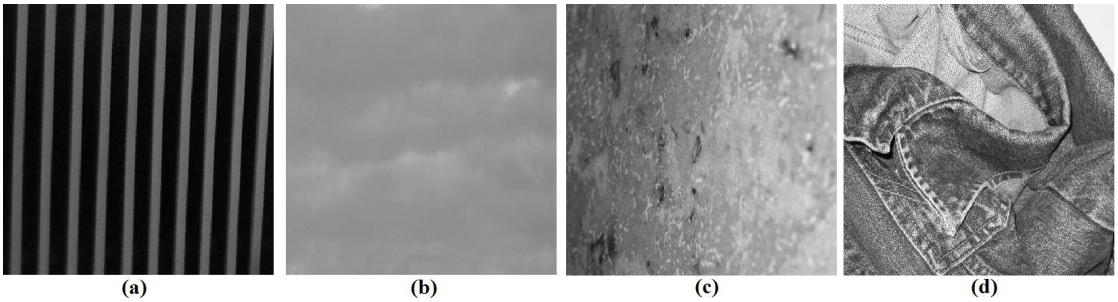


Abbildung 2.20.: a) gleichmäßig, grob und wellig b) ungleichmäßig, weich und eben c) sehr zufällig, eher fein und rau d) sowohl grob als auch fein, unregelmäßig

Haralick et al. betrachten hierzu in [HDS73] die **Grauwertematrix** (*engl. co-occurence matrix*) eines Bildes beziehungsweise Objekts und berechnen auf Grund dieser 14 statische Merkmale.

Die Grauwertematrix P eines Bildes mit N_g Graustufen ist für einen beliebigen gegebenen Abstand d eine $N_g \times N_g$-Matrix, welche die relative Lage der Grauwerte in eine Richtung widerspiegelt. Für die folgenden Texturmerkmale werden vier Grauwertmatrizen berechnet, P^0 für die horizontale Nachbarschaft, P^{90} für die vertikale und P^{45} und P^{135} für die beiden diagonalen Nachbarschaften. Die Terme $P(i,j,d,dir)$ aus Gleichung 2.48 geben an, wie viele Bildpunkte mit der Intensität i und j in Richtung dir mit Abstand d zueinander liegen.

$$\begin{aligned}
P(i,j,d,0°) = \; & \#\{((k,l),(m,n)) \in (L_x \times L_y) \times (L_x \times L_y) \\
& |k-m=0, \|l-n\| = d, \\
& I(k,l) = i, I(m,n) = j\} \\
P(i,j,d,45°) = \; & \#\{((k,l),(m,n)) \in (L_x \times L_y) \times (L_x \times L_y) \\
& |(k-m=d \wedge l-n=-d) \vee (k-m=-d \wedge l-n=d), \\
& I(k,l) = i, I(m,n) = j\} \\
P(i,j,d,90°) = \; & \#\{((k,l),(m,n)) \in (L_x \times L_y) \times (L_x \times L_y) \\
& |\|k-m\| = d, l-n=0, \\
& I(k,l) = i, I(m,n) = j\} \\
P(i,j,d,135°) = \; & \#\{((k,l),(m,n)) \in (L_x \times L_y) \times (L_x \times L_y) \\
& |(k-m=d \wedge l-n=d) \vee (k-m=-d \wedge l-n=-d), \\
& I(k,l) = i, I(m,n) = j\}
\end{aligned} \tag{2.48}$$

Das Erzeugen der Grauwertmatrizen lässt sich am einfachsten grafisch erklären, siehe Abbildung 2.2.13.

Um die Datenmenge einzuschränken und damit die Berechnung zu beschleunigen, werden zuvor die Graustufen des Bildes auf $N_g << 256$ Graustufen quantisiert.

Auf diesen Grauwertmatrizen zu einem beliebig festgelegten Abstand d werden nun für alle vier Richtungen vierzehn statische Merkmale berechnet. Zuvor noch eine Liste verwendeter Abkürzungen und Formeln:

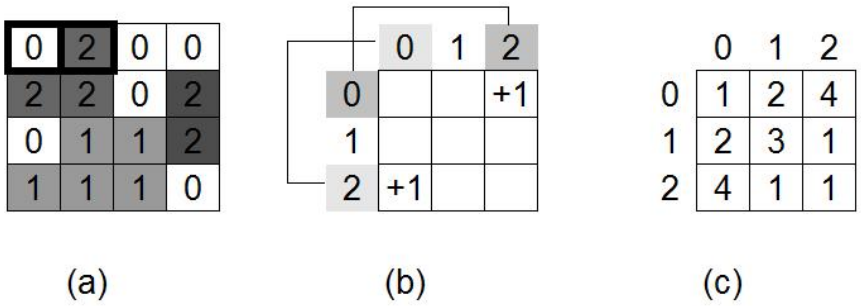


Abbildung 2.21.: Erzeugen einer horizontalen Grauwertematrix für $d = 1$: Das Bild (a) hat drei verschiedene Graustufen; in (b) werden die Zellen (0,2) und (2,0) je um eins inkrementiert; (c) fertige Grauwertematrix

- N_g Anzahl der Quantisierungsstufen für die Intensitätwerte des Bildes
- $\sum_i$ kurz für $\sum_{i=1}^{N_g}$
- $\sum_j$ kurz für $\sum_{j=1}^{N_g}$
- R Anzahl benachbarter Bildpunktepaare $= \sum_i \sum_j P(i,j)$
- $p(i,j)$ Eintrag in der mit R normalisierten Grauwertmatrix, $= P(i,j)/R$.
- $p_x(i)$ Vertikale Randverteilung, $= \sum_j p(i,j)$, Summe über alle Zeilen von $p(i,j)$.
- $p_y(j)$ Horizontale Randverteilung, $= \sum_i p(i,j)$, Summe über alle Spalten von $p(i,j)$.
- $p_{x+y}(k) = \underbrace{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g}}_{i+j=k} p(i,j)$ mit $k = 2, 3, ..., 2N_g$
- $p_{x-y}(k) = \underbrace{\sum_{i=1}^{N_g} \sum_{j=1}^{N_g}}_{|i-j|=k} p(i,j)$ mit $k = 0, 1, ..., N_g - 1$
- μ_x Mittelwert von p_x
- μ_y Mittelwert von p_y
- σ_x Standardabweichung von p_x
- σ_y Standardabweichung von p_y
- HXY Entropie von $p(i,j)$, $= -\sum_i \sum_j p(i,j) log(p(i,j))$

- HX Entropie von $p_x(i)$, $= -\sum_i p_x(i) log(p_x(i)))$
- HY Entropie von $p_y(j)$, $= -\sum_j p_y(j) log(p_y(j))$
- $HXY1 = -\sum_i \sum_j p(i,j) log(p_x(i) p_y(j))$
- $HXY2 = -\sum_i \sum_j (p_x(i) p_y(j)) log(p_x(i) p_y(j))$
- $Q(i,j) = sum_{k=1}^{N_g} \frac{p(i,k)p(k,j)}{p_x(i)p_y(k)}$

Die 14 Texturmerkmale, die auf Basis der Grauwertematrix berechnet werden, lauten:

1. Moment zweiter Ordnung
$$f_1 = \sum_i \sum_j p(i,j)^2 \tag{2.49}$$

2. Kontrast
$$f_1 = \sum_{k=0}^{N_g-1} k^2 p_{x-y}(k) \tag{2.50}$$

3. Korrelation
$$f_3 = \frac{\sum_i \sum_j (ij) p(i,j) - \mu_x \mu_y}{\sigma_x \sigma_y} \tag{2.51}$$

4. Varianz
$$f_4 = \sum_i \sum_j (i-\mu)^2 p(i,j) \tag{2.52}$$

5. Inverses Differenzen-Moment
$$f_5 = \sum_i \sum_j \frac{1}{1+(i-j)^2} p(i,j) \tag{2.53}$$

6. Summen-Mittelwert
$$f_6 = \sum_{i=2}^{2N_g} i p_{x+y}(i) \tag{2.54}$$

7. Summen-Varianz
$$f_7 = \sum_{i=2}^{2N_g} (i-f_6)^2 p_{x+y}(i) \tag{2.55}$$

8. Summen-Entropie
$$f_8 = -\sum_{i=2}^{2N_g} p_{x+y}(i) log(p_{x+y}(i)) \tag{2.56}$$

   $log(p)$ wird hier durch $log(p+\epsilon)$ ersetzt, um $log(0)$ zu vermeiden.

9. Entropie
$$f_9 = -\sum_i \sum_j p(i,j) log(p(i,j)) \tag{2.57}$$

10. Differenzen-Varianz
$$f_{10} = \text{Varianz von } p_{x-y} \tag{2.58}$$

11. Differenzen-Entropie
$$f_{11} = -\sum_{i=0}^{N_g-1} p_{x-y}(i) log(p_{x-y}(i)) \tag{2.59}$$

12. Informationsgehalte der Korrelation 1
$$f_{12} = \frac{HXY - HXY1}{max\{HX, HY\}} \tag{2.60}$$

13. Informationsgehalte der Korrelation 2
$$f_{13} = \sqrt{1 - exp(-2(HXY2 - HXY))} \tag{2.61}$$

14. Maximaler Korrelationskoeffizient
$$f_{14} = \sqrt{\text{Zweitgrößter Eigenwert von Q}} \tag{2.62}$$

Aus den 14 Funktionen f_i erhält man 56 Werte für ein festes d. Über diese vier Werte ermittelt man je den Mittelwert und die Intervallgröße, welche die Schwankung zwischen den Richtungen ausdrückt, und erhält 28 aussagekräftige Merkmale zur Beschreibung einer Textur. Diese wurden beispielhaft für $d = 1$ für die vier verschiedenen Texturen in Abbildung 2.20 in Tabelle 2.2.13 augezählt.

.

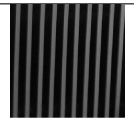
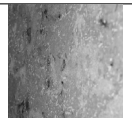
Name des Merkmals	**a)** 	**b)** 	**c)** 	**d)** 
MEAN_ANGULAR_SECOND_MOMENT	0,12	0,33	0,04	0,03
RANGE_ANGULAR_SECOND_MOMENT	0,07	0,11	0,04	0,04
MEAN_CONTRAST	0,70	0,09	0,64	13,22
RANGE_CONTRAST	2,49	0,19	1,82	26,84
MEAN_CORRELATION	161.137,59	15.113,56	162.252,95	519.359,70
RANGE_CORRELATION	312,96	8,62	863,20	30.096,30
MEAN_SUM_OF_SQUARES	643,61	170,51	172,29	292,40
RANGE_SUM_OF_SQUARES	0,04	0,02	0,00	1,33
MEAN_INVERSE_DIFFERENCE_MOMENT	0,90	0,95	0,88	0,63
RANGE_INVERSE_DIFFERENCE_MOMENT	0,26	0,10	0,28	0,74
MEAN_SUM_AVERAGE	49,57	26,07	24,46	31,63
RANGE_SUM_AVERAGE	0,00	0,00	0,00	0,11
MEAN_SUM_VARIANCE	−499,08	−109,29	−161,18	−259,88
RANGE_SUM_VARIANCE	15,63	8,73	1,74	7,27
MEAN_SUM_ENTROPY	−2,53	−1,38	−3,17	−3,46
RANGE_SUM_ENTROPY	0,58	0,44	0,65	0,62
MEAN_ENTROPY	2,73	1,45	3,46	4,52
RANGE_ENTROPY	1,18	0,58	1,35	2,75
MEAN_DIFFERENCE_VARIANCE	0,76	0,85	0,72	0,56
RANGE_DIFFERENCE_VARIANCE	0,57	0,31	0,59	0,89
MEAN_DIFFERENCE_ENTROPY	−0,45	−0,24	−0,50	−1,21
RANGE_DIFFERENCE_ENTROPY	1,18	0,49	1,09	2,42
MEAN_INFMEAS_CORR_COEFF_1	0,79	0,77	0,78	0,57
RANGE_INFMEAS_CORR_COEFF_1	0,53	0,49	0,47	0,87
MEAN_INFMEAS_CORR_COEFF_2	0,98	0,90	0,99	0,87
RANGE_INFMEAS_CORR_COEFF_2	0,06	0,12	0,02	0,25

Tabelle 2.1.: Diese Werte erhält man, wenn man die Texturmerkmale für die vier Bilder aus Abbildung 2.20 für einen Abstand von $d = 1$ berechnet. Es wurde jeweils über die vier Richtungen gemittelt beziehungsweise die Intervallgröße angegeben. Unterschiedliche Merkmale sind bei unterschiedlichen Texturarten signifikant.

2.3. Fuzzy-Logik

In diesem Kapitel sollen die Grundlagen der Fuzzy-Logik erläutert werden. Es basiert auf dem Buch *Fuzzy-Systeme* von Kruse, Gebhardt und Klawonn [RK93] und dem Artikel über Fuzzy-Logik von Roland Stelzer [Ste04].

2.3.1. Aussagenlogik

Die Aussagenlogik dient dazu Faktenwissen in Formeln auszudrücken und syntaktisch neues Wissen abzuleiten. Eine aussagenlogische Formel oder Variable kann einen der zwei Wahrheitswerte *wahr* oder $falsch$ Annahmen. Äquivalent hierzu sind die Schreibweisen *true* und $false$ beziehungsweise 1 und 0. Ein aussagenlogischer Term besteht dabei aus einer oder mehr Variablen, welche mit den folgenden Operatoren verknüpft sind.

Negation $\neg A$ ist wahr, wenn A nicht wahr ist.

Konjunktion $A \wedge B$ ist wahr, wenn sowohl A als auch B wahr sind.

Disjunktion $A \vee B$ ist wahr, wenn mindestens eine der beiden Variablen A oder B wahr ist.

Implikation $A \Rightarrow B$ bedeutet, wenn A wahr ist, so muss auch B wahr sein. Ist A falsch, so kann über B keine Aussage getroffen werden.

Äquivalenz $A \Leftrightarrow B$ bedeutet, dass der Wahrheitswert von A immer dem von B entspricht.

A	B	$\neg A$	$A \wedge B$	$A \vee B$	$A \Rightarrow B$	$A \Leftrightarrow B$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Tabelle 2.2.: Wahrheitstafel: A und B seien zwei logische Aussagen. 1 steht für den Wahrheitswert `wahr` und 0 für `falsch`.

Die Implikation kann auch durch die Konjunktion und Negation ausgedrückt werden: $A \Rightarrow B = \neg A \vee B$. Siehe auch 2.3. Auch die Äquivalenz lässt sich durch die drei Grundoperatoren ausdrücken:$A \Leftrightarrow B = (A \wedge B) \vee (\neg A \wedge \neg B)$.

Das syntaktische Umformen aussagenlogischer Formeln ist über die fundamentalen Äquivalenzen der Aussagenlogik möglich:

Idempotenz $A \wedge A \equiv A$ und $A \vee A \equiv A$

Kommutativität $A \wedge B \equiv B \wedge A$ und $A \vee B \equiv B \vee A$

Assoziativität $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ und $(A \vee B) \vee C \equiv A \vee (B \vee C)$

A	B	$\neg A$	$\neg A \vee B$	$A \Rightarrow B$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	0
1	1	0	1	1

Tabelle 2.3.: Beweis für die Äquivalenz von $A \Rightarrow B$ und $\neg A \vee B$.

Absorption $A \wedge (A \vee B) \equiv A$ und $A \vee (A \wedge B) \equiv A$

Distributivität $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ und $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$

Doppelnegation $\neg\neg A \equiv A$

deMorgan'sche Regel $\neg(A \wedge B) \equiv \neg A \vee \neg B$ und $\neg(A \vee B) \equiv \neg A \wedge \neg B$

Tautologieregel Ist A immer war (Tautologie), so ist $A \vee B \equiv A$ und $A \wedge B \equiv B$

Unerfüllbarkeitsregel Ist A unerfüllbar, so ist $A \vee B \equiv B$ und $A \wedge B \equiv A$

Inferenz

Mittels der Implikation ist es möglich, aus dem vorhandenen Wissen neues Wissen abzuleiten. Wenn bekannt ist, dass A wahr und B falsch ist und die Formel $(A \wedge \neg B) \Rightarrow C$ gilt, kann man postulieren, dass auch C wahr sein muss. Genauso lassen sich einfache Schluss-/ Expertensysteme bauen, welche mit Wenn-Dann-Regeln arbeiten.

2.3.2. Motivation

Mit Hilfe der Fuzzy Logik lassen sich natürlichsprachliche Aussagen mathematisch handhabbar machen, die nicht so eindeutig abgegrenzt sind wie mathematische Aussagen. In der klassischen Boole'schen Logik kann eine Aussage genau die zwei Wahrheitswerte wahr oder falsch Annahmen. So hat die Aussage "*Anna ist volljährig*"genau dann den Wahrheitswert wahr, wenn Anna 18 Jahre oder älter ist. Ist sie jünger als 18 Jahre, so hat die Aussage den Wahrheitswert falsch. Schwerer greifbar ist hier die Aussage "*Anna ist erwachsen.*". Man kann Menschen generell nicht ab einem bestimmten Alter als erwachsen ansehen. So legt eine 16jährige Person schon eigenverantwortliches Handeln an den Tag. Aber auch eine 24jährige Person, die schon deutlich erwachsener ist, verhält sich eventuell manchmal noch kindisch. Hier ist also ein fließender Übergang von „*Anna ist nicht erwachsen.*“ zu „*Anna ist erwachsen.*“ gefordert, der widerspiegelt, dass man mit 22 zwar generell erwachsener ist als mit 17, aber trotzdem noch nicht vollkommen erwachsen. Dies leisten Fuzzy-Mengen.

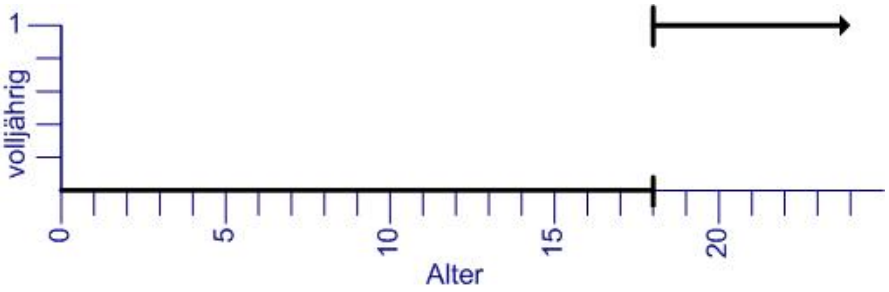


Abbildung 2.22.: Die Boole'sche Funktion „volljährig“

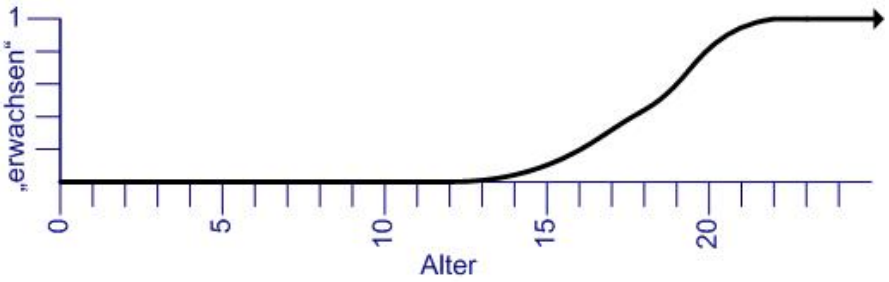


Abbildung 2.23.: Die Fuzzy-Funktion „erwachsen“

2.3.3. Fuzzy-Mengen

Bei Boole'schen Aussagen lassen sich die Wahrheitswerte auch über Mengen darstellen. Sei das Universum U = 0,1,2,...129,130 das Alter eines Menschen in Jahren, dann repräsentiert die Menge $M_1 = 18, 19, 20, ..., 129, 130$ alle die Altersstufen, für welche die Aussage „*Anna ist erwachsen.*“ wahr ist. Die Elemente $1, 2, 3, ..., 17$ gehören nicht zu dieser Menge, für sie ist die Aussage „*Anna ist erwachsen.*“ falsch. Eine andere Schreibweise ist die charakteristische Funktion $\mu_M : U \rightarrow \{0, 1\}$ beziehungsweise $\mu_M : U \rightarrow \{wahr, falsch\}$, welche jedes Objekt aus dem Universum auf einen der beiden Wahrheitswerte abbildet.

Bei Fuzzy-Mengen existiert ein Pendant zu dieser charakteristischen Funktion, die Zugehörigkeitsfunktion $\mu_M : U \rightarrow [0, 1]$. Diese gibt für jedes Objekt des Universums an, wie sehr eine Aussage auf eben dieses zutrifft. 0 entspricht hierbei dem Boole'schen falsch und 1 dem Boole'schen wahr.

Die mathematische Modellierung der Fuzzy-Mengen ist vollkommen frei wählbar. Gängig sind einfache Dreicks- oder Trapezfunktionen mit geraden Flanken wie zum Beispiel

$$\nu_{Dreieck} = \begin{cases} x - 4 & bei\ 4 < x \leq 5 \\ 6 - x & bei\ 5 < x \leq 6 \\ 0 & sonst \end{cases}$$

oder

$$\nu_{Trapez} = \begin{cases} x-2 & bei\ 2 < x \leq 3 \\ 1 & bei\ 3 < x \leq \\ 10-2x & bei\ 4,5 < x \leq 5 \\ 0 & sonst \end{cases}$$

Einen etwas fließenderen Übergang bieten die von Zadeh[Zad73] vorgeschlagenen s-Funktion und z-Funktion für Schwellwerte und s/z-Funktion für Intervalle:

$$s(x,\alpha,\beta) = \begin{cases} 0 & bei\ x \leq \alpha \\ 2 \cdot (\frac{x-\alpha}{\beta-\alpha})^2 & bei\ \alpha < x \leq \frac{\beta-\alpha}{2} \\ 1-2 \cdot (\frac{x-\alpha}{\beta-\alpha})^2 & bei\ \frac{\beta-\alpha}{2} < x \leq \beta \\ 1 & bei\ x > \beta \end{cases} \tag{2.63}$$

.

$$z(x,\alpha,\beta) = \begin{cases} 1 & bei\ x \leq \alpha \\ 1-2 \cdot (\frac{x-\alpha}{\beta-\alpha})^2 & bei\ \alpha < x \leq \frac{\beta-\alpha}{2} \\ 2 \cdot (\frac{x-\alpha}{\beta-\alpha})^2 & bei\ \frac{\beta-\alpha}{2} < x \leq \beta \\ 0 & bei\ x > \beta \end{cases} \tag{2.64}$$

.

$$s/z(x,\alpha,\beta,\gamma,\delta) = min \begin{pmatrix} s(x,\alpha,\beta) \\ z(x,\gamma,\delta) \end{pmatrix}. \tag{2.65}$$

. die z-Funktion ist das Gegenstück zur s-Funktion, stellt also nicht den Verlauf von 0 nach 1 sondern von 1 nach 0 dar. Als Zugehörigkeitsfunktion ist aber grundsätzlich jede denkbare und beliebig komplizierte Funktion möglich, welche auf das Intervall $[0,1]$ abbildet.

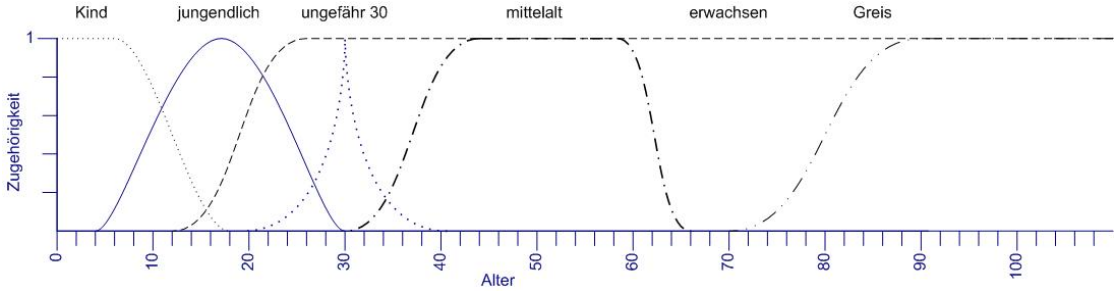


Abbildung 2.24.: Verschiedene Altersstufen als Fuzzy-Mengen modelliert. Zum Beispiel lässt sich erkennen, dass ein 18jähriger eine Zugehörigkeit von 0,6 zu der Fuzzy-Menge „erwachsen“ und eine Zugehörigkeit von 1,0 zu der Fuzzy Menge „Jugendlicher“ hat

Eine Fuzzy-Menge repräsentiert einen Wert einer *Linguistische Variable*. Eine linguistische Variable ist eine Eigenschaft wie zum Beispiel Farbe oder eine Altersstufe. Zu der

linguistischen Variablen Altersstufe gäbe es zum Beispiel die Fuzzy-Mengen Kleinkind, Kind, Jugendlicher, Erwachsener, Greis. Diese können sich auch problemlos überschneiden, da durch die Zugehörigkeitsfunktion angegeben ist, wie sehr ein Objekt zu einer Fuzzy-Menge gehört. Klar ist zum Beispiel, dass jeder, der vollkommen vergreist ist ($\mu_{Greis}(x) = 1$), definitiv auch vollkommen zur Gruppe der erwachsenen Personen gehört ($\mu_{erwachsen}(x) = 1$).

2.3.4. Operatoren

Um die Verknüpfung von Fuzzy-Mengen zu realisieren, benötigt man Operatoren auf Mengen analog zu denen der Aussagenlogik. Da alle komplexeren Operatoren der Aussagenlogik auf den drei Grundoperatoren Negation, Konjunktion und Disjunktion aufbauen, genügt es, hierfür Äquivalenzen zu finden.

Angenommen, Anna ist 18 Jahre alt, dann gilt für Anna die Aussage „*Anna ist erwachsen.*“ mit einem Zugehörigkeitswert von 0,4 und „*Anna ist jugendlich.*“ mit einem Zugehörigkeitswert von 0,95. Wie stark ist dann die Zugehörigkeit zu der Aussage „*Anna ist erwachsen und jugendlich.*“? Diese Verknüpfung erreicht man mit *t-Normen* und *s-Normen* (auch *t-Conormen*), wobei eine t-Norm der Konjunktion und eine s-Norm der Disjunktion entspricht.

Die einfachste Möglichkeit ist durch die t-Norm $max()$ und die s-Norm $min()$ gegeben. Da die Aussage „*Anna ist erwachsen und jugendlich.*“ einer Konjunktion entspricht, wählt man als Zugehörigkeit den kleinsten Zugehörigkeitswert der beiden Teilaussagen, also $0,5$. Umgekehrt hat die Zugehörigkeit zur Aussage „*Anna ist erwachsen oder jugendlich.*“ den Wert $0,9$, weil bei einer Disjunktion das Maximum der Teilaussagen gewählt wird.

$$
\begin{aligned}
Anna\ ist\ 18 \rightarrow x &= 18 \\
\mu_{erwachsen}(x) &= \begin{cases} 0 & wenn\ x < 12 \\ 1 & wenn\ x > 26 \\ \frac{1}{1+e^{x-18}} & sonst \end{cases} \\
\mu_{jugendlich}(x) &= \begin{cases} \frac{5}{\sigma\sqrt{2\pi}} exp(-\frac{1}{2}(\frac{x-17}{2})^2) & wenn\ 4 \leq x \leq 30 \\ 0 & sonst \end{cases} \\
\mu_{erwachsen}(18) &= 0,5 \\
\mu_{jugendlich}(18) &= 0,88 \\
\mu_{jugendlich \wedge erwachsen}(18) &= min\{\mu_{jugendlich}(18), \mu_{erwachsen}(18)\} \\
\mu_{jugendlich \wedge erwachsen}(18) &= 0,5
\end{aligned}
$$

Eine t-Norm $T : [0,1] \times [0,1] \rightarrow [0,1]$ muss einige Bedingungen erfüllen:

1. Einselement $T(a,1) = a$
2. Monotonie $a \leq b \Rightarrow T(a,c) \leq T(b,c)$
3. Kommutativität $T(a,b) = T(b,a)$

4. Assoziativität $T(a, T(b, c)) = T(T(a, b), c)$

t-Norm und s-Norm sind dual zueinander. Eine Übersicht über andere gängige t-Normen und s-Normen bietet Tabelle 2.4 und ein Vergleich dieser findet sich in Abbildung 2.25.

Name	t-Norm	s-Norm
Zadeh	$min\{a, b\}$	$max\{a, b\}$
Lukasiewicz	$max\{0, a + b - 1\}$	$min\{1, a + b\}$
prod	$a \cdot b$	$a + b - ab$

Tabelle 2.4.: Gängige t-Normen und s-Normen

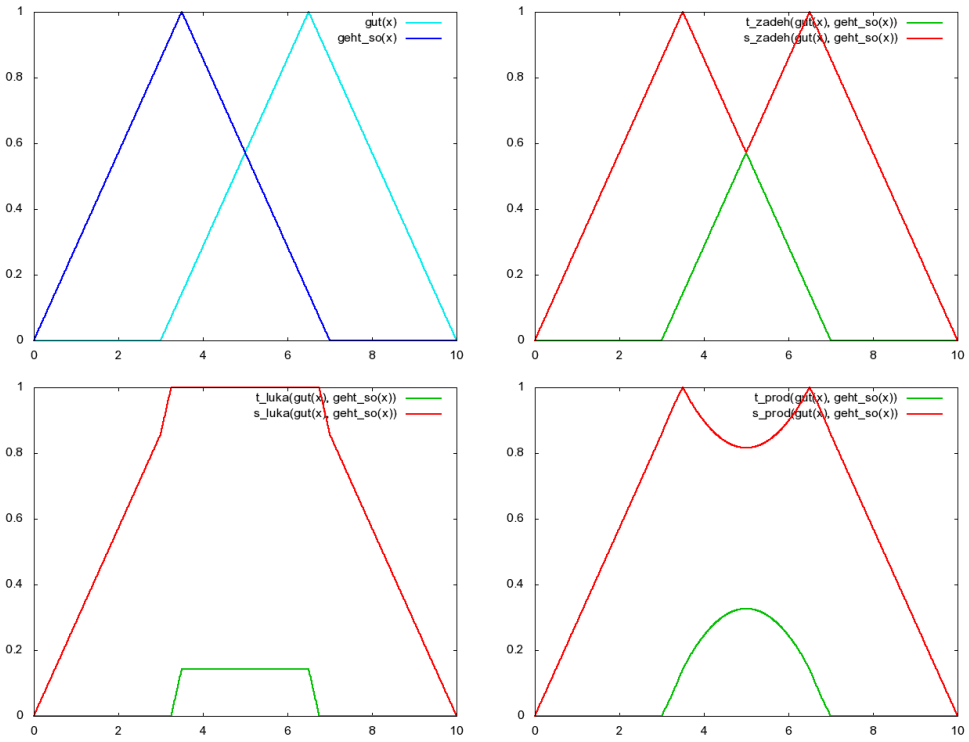


Abbildung 2.25.: Die zwei Fuzzy-Mengen *gut* und *geht_so* und ihre Verknüpfung durch verschiedene Normen.

Die Negation einer Aussage erhält man, indem man den Zugehörigkeitswert dieser Aussage von 1 abzieht.

$$\neg\mu_A(x) = 1 - \mu_A(x) \tag{2.66}$$

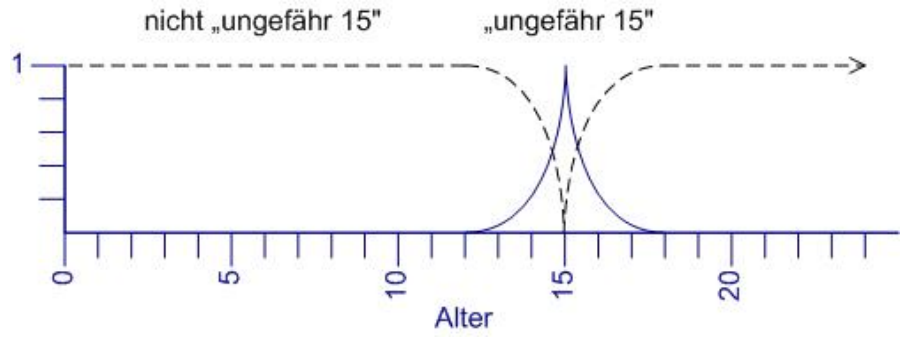


Abbildung 2.26.: Die Fuzzy-Menge zu der Aussage $ungefaehr15$ und ihre Negation $\neg ungefaehr15$

2.3.5. Fuzzy-Regeln

Bisher wurden nur Aussagen in Fuzzy-Form dargestellt und verknüpft. Doch Wissen beinhaltet nicht nur Fakten, sondern auch Handlungsvorschriften. In der Fuzzy-Logik sind diese Handlungsvorschriften einfache Regeln der Form

```
WENN Vorbedingung DANN Konsequenz.
```

wobei die Konsequenz (engl. *consequence*) aus einer Fuzzy-Menge besteht und die Vorbedingung (auch häufig Prämisse oder engl. *premise* oder *condition*) sich aus der Verknüpfung beliebig vieler Fuzzy-Mengen zusammensetzt.

Als Beispiel betrachte man die Regel

```
WENN Die Fernsehsendung ist sehr brutal UND der Zuschauer ist jungendlich
   ODER Die Fernsehsendung ist brutal UND der Zuschauer ist Kind DANN
        Eignung des Programms ist ungeeignet.
```

Hier wurden drei verschiedene linguistische Variablen benutzt: *Brutalität*, *Altersstufe* und *Eignung* mit den Fuzzy-Mengen *sehr brutal*(Brutalität), *brutal*(Brutalität), *Kind*(Altersstufe), *jugendlich*(Altersstufe) und *ungeeignet*(Eignung). Man kann entweder festlegen, dass der UND-Operator stärker bindet als der ODER-Operator oder eine konsequente Klammerung fordern. Im Weiteren wird aber von einer stärkeren Bindung in der Reihenfolge NICHT > UND > ODER ausgegangen.

2.3.6. Inferenz

Um das oben in Fuzzy-Form dargestellte Wissen zu nutzen, wird ein (Fuzzy-)Inferenz-System (FIS) benötigt, mit dem aus dem bestehenden Wissen basierend auf scharfen Eingabewerten Schlüsse gezogen werden können. Ein solches FIS wird auch als Controller bezeichnet. Der Begriff kommt aus der Regelungstechnik, da hier Fuzzy-System zuerst im großen Stil Einsatz fanden.

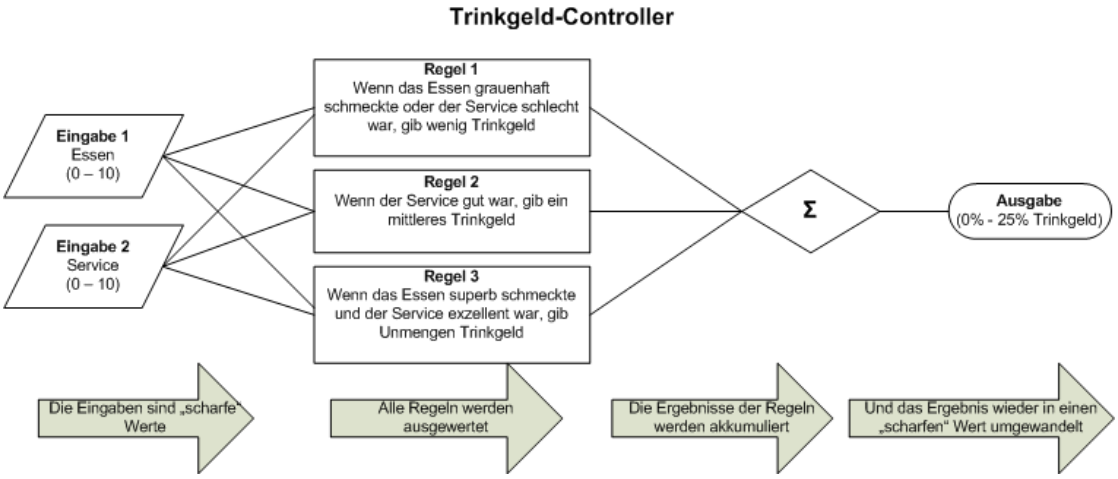


Abbildung 2.27.: Fuzzy-Controller mit Regelbasis, der beim Berechnen des Trinkgelds hilft

Fuzzyfizierung

Da die Regeln der Regelbasis nur aus Fuzzy-Mengen bestehen, müssen die „scharfen" Eingabewerte zunächst in Zugehörigkeitswerte zu den Fuzzy-Mengen überführt werden. Dazu werden für die zum Eingabewert passende linguistische Variable die Zugehörigkeitswerte aller ihrer Fuzzy-Mengen errechnet. Die Abbildung 2.28 zeigt, dass ein Essen, das mit 9 bewertet wurde, eine Zugehörigkeit von 1,0 zur Menge *superb* und 0,4 zur Menge *lecker* hat. Für alle anderen Mengen ist die Zugehörigkeit 0.

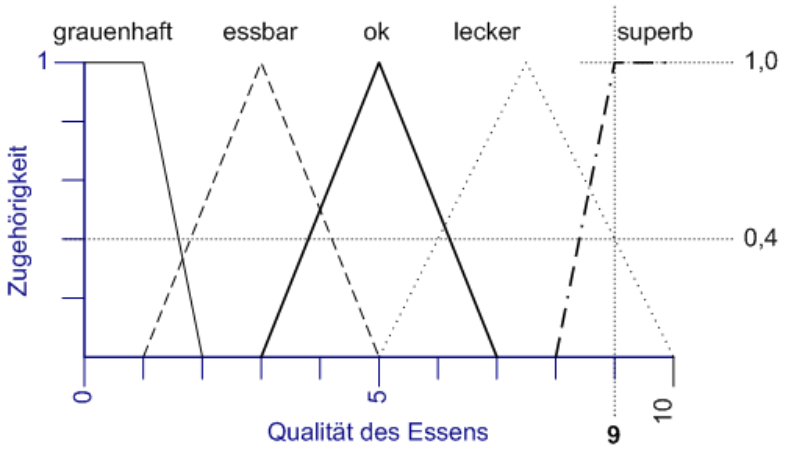


Abbildung 2.28.: Fuzzyfizierung des scharfen Werts 9 für die Qualität des Essens

Einer Bewertung von 10 für das Essen wird die Fuzzy Menge *superb* mit 1,0 zugeornet und die Menge *lecker* mit 0,4. Der mit 8,2 bewertete Service gehört mit 0,5 zur Fuzzy-

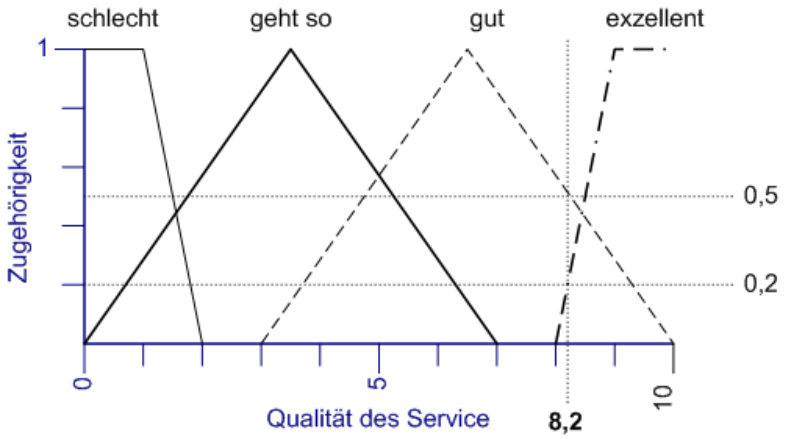


Abbildung 2.29.: Fuzzyfizierung des scharfen Werts 4 für den Service

Menge *gut* und mit 0,2 zur Fuzzy-Menge *exzellent*.

Aggregation

In der Aggregationsphase werden für alle Regeln die Zugehörigkeitswerte ihrer Vorbedingungen berechnet. Dafür werden die Zugehörigkeitsgrade zu den Fuzzy-Mengen einer Regeln mit t- und s-Normen verknüpft, je nach gewählter logischer Verknüpfung (UND oder ODER). Hierbei ist zu beachten, dass ein Controller durchgängig nur eine t- und eine s-Norm verwendet. Der Einfachheit halber werden in diesem Beispiel die t-Norm $max()$ und die s-Norm $min()$ benutzt.

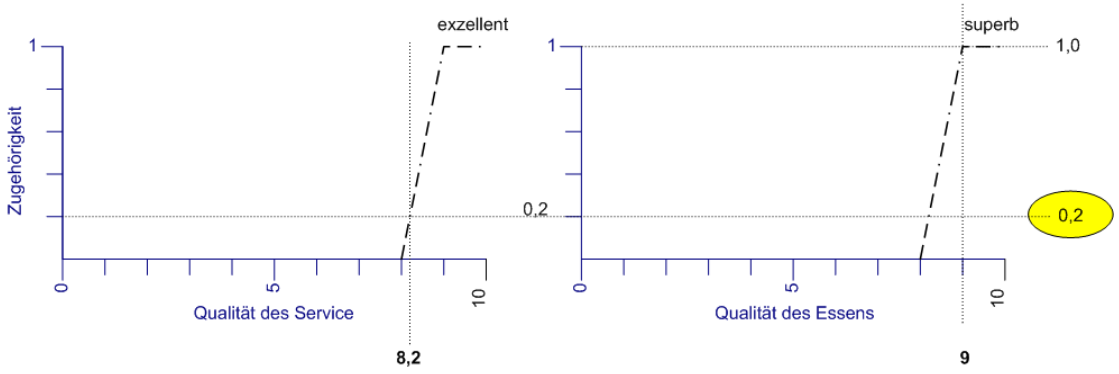


Abbildung 2.30.: Prämisse der Regel 3

In Regel 3 werden zwei Fuzzy-Mengen mit UND verknüpft. Deshalb ist der Zugehörigkeitswert zur Prämisse der Regeln das Minimum der beiden einzelnen Zugehörigkeitswerte, also 0,2. Siehe 2.30. Eine Liste mit den Zugehörigkeitswerten aller Fuzzy-Mengen und Prämissen findet sich in Tabelle 2.5.

Fuzzy-Menge	Zugehörigkeit	
Essen↦grauenhaft	0,0	
Essen↦essbar	0,0	
Essen↦ok	0,0	
Essen↦lecker	0,4	
Essen↦superb	1,0	
Service↦schlecht	0,0	
Service↦geht so	0,0	
Service↦gut	0,5	
Service↦exzellent	0,2	
Essen↦grauenhaft ODER Service↦schlecht	0,0	in Regel 1
Service↦gut	0,5	in Regel 2
Essen↦superb ODER Service↦exzellent	0,0	in Regel 3

Tabelle 2.5.: Zugehörigkeiten zu Fuzzy-Mengen und -Regeln bei Essen = 9 und Service = 8,2

Implikation

Jetzt wird die Ausgabe jeder Fuzzy-Regeln bestimmt. Die Ausgabe einer Regel ist die Fuzzy-Menge der Konsequenz, „abgeschnitten“ an dem Zugehörigkeitswert der Prämisse.

$$\nu_{Regel} = min(\nu_{Praemisse}, \nu_{Konsequenz}) \tag{2.67}$$

Auch dies lässt sich am besten wieder an einem Bild erklären

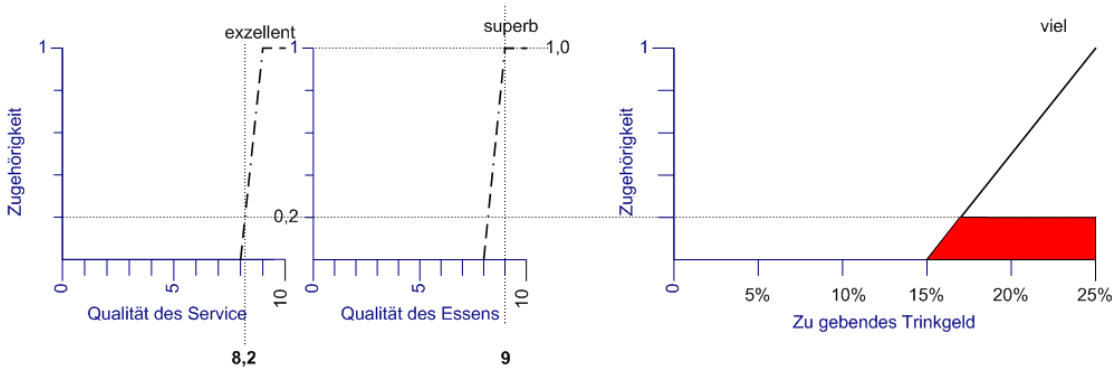


Abbildung 2.31.: Die Fuzzy-Menge der Kosequenz wird auf die Höhe gekürzt, welche sich durch die Verknüpfung der Fuzzy-Mengen der Prämisse ergeben

Akkumulation

Nachdem alle Regeln ausgewertet wurden, müssen die resultierenden Fuzzy-Mengen verknüpft werden, welche Aussagen über dieselbe linguistische Variable treffen. Dies ge-

schieht über die verwendete s-Norm und hat als Resultat wieder eine Fuzzy-Menge zur betrachteten linguistischen Variable.

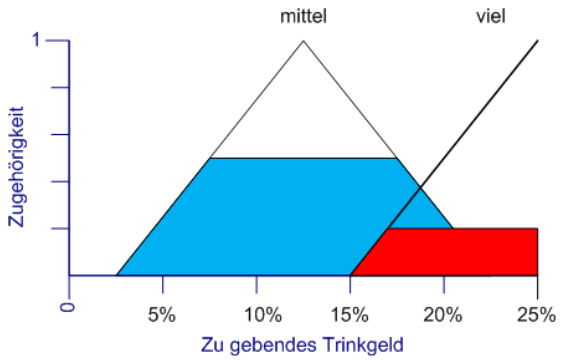


Abbildung 2.32.: Die Ergebnis-Fuzzy-Mengen der Regeln 2 und 3 mit der s-Norm $min()$ verknüpft.

Defuzzyfizierung

Die gewünschte Ausgabe eines Fuzzy-Controllers ist aber keine Fuzzy-Menge, sondern wieder ein „scharfer“ Ausgabewert. Deshalb muss die unscharfe Ergebnis-Fuzzy-Menge jeder linguistischen Variablen defuzzyfiziert, das heißt in einen „scharfen“ Wert umgerechnet werden. Hierfür stehen verschiedene Verfahren zur Verfügung: Die *Maximum*-Methode kann angewendet werden, wenn das Ergebnis nur ein einziges globales Maximum besitzt. Dann wird einfach der zugehörige Wert als Ausgabe gewählt. Die Mean-of-Maximum kann zum Einsatz kommen, wenn in der Ausgabe mehrere Maxima auftauchen. Hier wird der Mittelwert über alle Maxima ausgegeben. Diese Methode ist jedoch sehr anfällig für kleine Änderungen in der Eingabe, da es hier zu Sprüngen im Ausgabeverhalten kommen kann. Auch ist das Verhalten bei einer symmetrischen Ausgabe mit einem „Loch“ in der Mitte problematisch, da hier ein Ausgewert gewählt würde, der eine Zugehörigkeit von 0 zum Ergebnis hat. Die üblichste Methode ist die Schwerpunktmethode (engl. *Center-of-Gravity*, *CoG*). Hier wird die Ergebnis-Fuzzy-Menge des Akkumulationsschritts als geometrische Form aufgefasst und ihr Schwerpunkt berechnet. Die X-Koordinate des Schwerpunkts entspricht dann der Ausgabe. Rechnerisch ist das über Integration über die Fläche der Fuzzy-Menge möglich:

$$x_{Ausgabe} = \frac{x \cdot \int \mu(x)}{\int \mu(x)} \tag{2.68}$$

Am Beispiel des Restaurantbesuchs lässt sich festellen, dass der Service eher gut als exzellent war und trotz des superben Essens für den Kellner nur 13,57% Trinkgeld übrigbleiben.

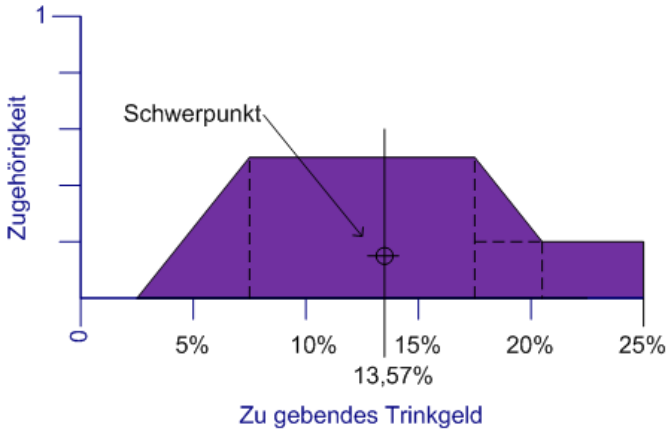


Abbildung 2.33.: Der Schwerpunkt der Ergebnis-Fuzzy-Menge liegt bei (13,57%; 0,18).

3. Konzeption

Klassifikation bedeutet, jedem Objekt aus einem gegebenen Universum einer Teilmenge dieses Universums zuzuordnen. Diese Teilmengen des Universums können sich je nach Zielsetzung überschneiden oder vollständig disjunkt sein.

Da die Zuweisung nicht willkürlich erfolgen soll, müssen dem Klassifikationssystem Entscheidungskriterien an die Hand gegeben werden. Dafür werden für jedes Objekt beschreibende Merkmale ermittelt. In der realen Welt können dies zum Beispiel Eigenschaften wie Farbe, Größe oder Geschmack sein. In einem Computersystem hängen solche Merkmale von der jeweiligen Aufgabe ab und werden durch Zahlenwerte repräsentiert.

Die Klassifikation kann zum einen durch ein angelerntes Klassifikationssystem erfolgen oder durch ein Expertensystem. Bei letzterem muss das Wissen, welches die Zuordnungen veranlasst, vom Benutzer bereitgestellt oder manuell erfasst werden.

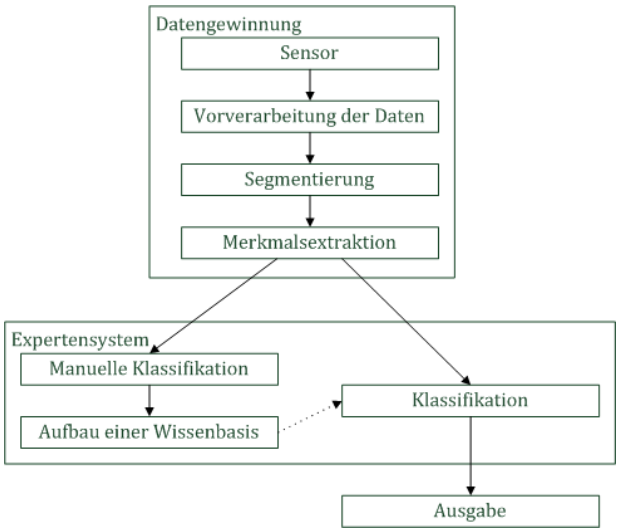


Abbildung 3.1.: Reihenfolge der Arbeitschritte eines Klassifikationssystems

Meistens besteht die Aufgabe eines Klassifikationssystems nicht allein in der Zuordnung von Objekten zu Teilmengen sondern auch im Auffinden solcher Objekte in gegebenen Daten. Hierbei kann es sich zum Beispiel um Audiosignale oder Bilder handeln, aus welchen die gesuchten Segmente ausgeschnitten werden. Um die Segmentierung zu erleichtern oder bestimmte Merkmale zu berechnen, werden die Eingabedaten vorverarbeitet. So kann ein eventuell verrauschtes Eingabesignal durch einen vorgeschalteten Glättungsfilter ausgeglichen werden. Bei Bildern kann zur Merkmalsextraktion die Umrechnung in einen anderen Farbraum nützlich sein.

4. Eigene Arbeiten

Um die titelgebende Klassifikation von Blütenpollen zu erreichen, sind drei Arbeitsschritte notwendig. Zuerst müssen aus den gegebenen Aufnahmen mögliche Pollen segmentiert werden. Für diese Kandidaten werden nun beschreibende Merkmale berechnet. Zuletzt werden diese Kandidaten anhand ihrer Merkmale bezüglich einer vorher erstellten Wissensbasis mit einem Fuzzy-Logik-System klassifiziert. Diese drei Schritte und der Aufbau der Wissenbasis werden im folgenden detailliert beschrieben. Zum besseren Verstehen werden alle Arbeitschritte auch durch Abbildungen sichtbar gemacht.

4.1. Vorverarbeitung der Daten

4.1.1. Format der Eingabedaten

Alle zur Verfügung stehenden Bilder von Pollen lagen als farbige JPEG-Dateien vor. Es fanden sich vier Aufnahmen in 100facher und fünf Bilder in 400facher Vergrößerung. Trotz des JPEG-Formates hatten alle Aufnahmen eine gute Qualität und wiesen keinerlei JPEG-Artefakte oder sichtbare Spuren einer verlustbehafteten Kompression auf. Die Abmessungen aller Bilder betrugen 1280 x 1024 Pixel. Beispiele sind in Abbildung 4.2 zu sehen. Auf einigen Aufnahmen finden sich neben den zu klassifizierenden Pollenkörner auch noch Verschmutzungen, Schwebeteilchen, Staub, Sporen und auch Asphalt. Auch die Verschmutzung auf der Optik des Mikroskops ist auf diversen Aufnahmen wiederzuerkennen. Leider standen keine Kalibrierungsaufnahmen zur Verfügung, mit denen es möglich gewesen wäre, diese herauszurechnen.

4.1.2. Graustufenbild

Die Farbinformationen der vorliegenden Bilder ist so insignifikant, dass eine Betrachtung der reinen Helligkeitswerte vollkommen ausreichend ist. Außerdem wäre ein Verfahren, welches die Farbinformationen berücksichtigt, nur aufwändig und speicherintensiv zu implementieren. Von daher werden alle Bilder direkt beim Laden nach der Formel aus Kapitel 2.2.2 in Graustufenbilder mit dem Wertebereich $[0, 1]$ konvertiert. An Abbildung 4.3 kann man erkennen, dass die Bilder fast ausschließlich aus Gelbtönen unterschiedlicher Helligkeit bestehen.

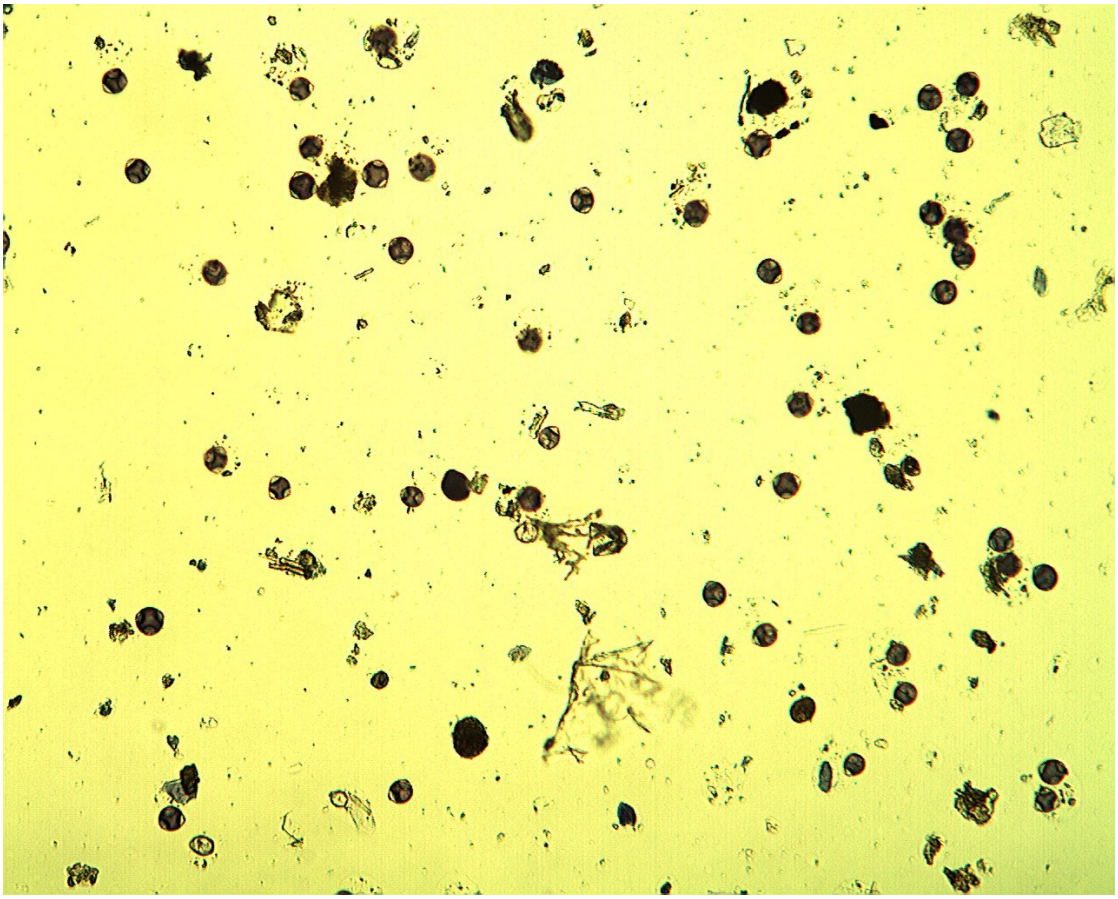


Abbildung 4.1.: Aufnahme in 100facher Vergrößerung. Zu sehen sind einige Pollenkörner sowie Staub und Mineralien.

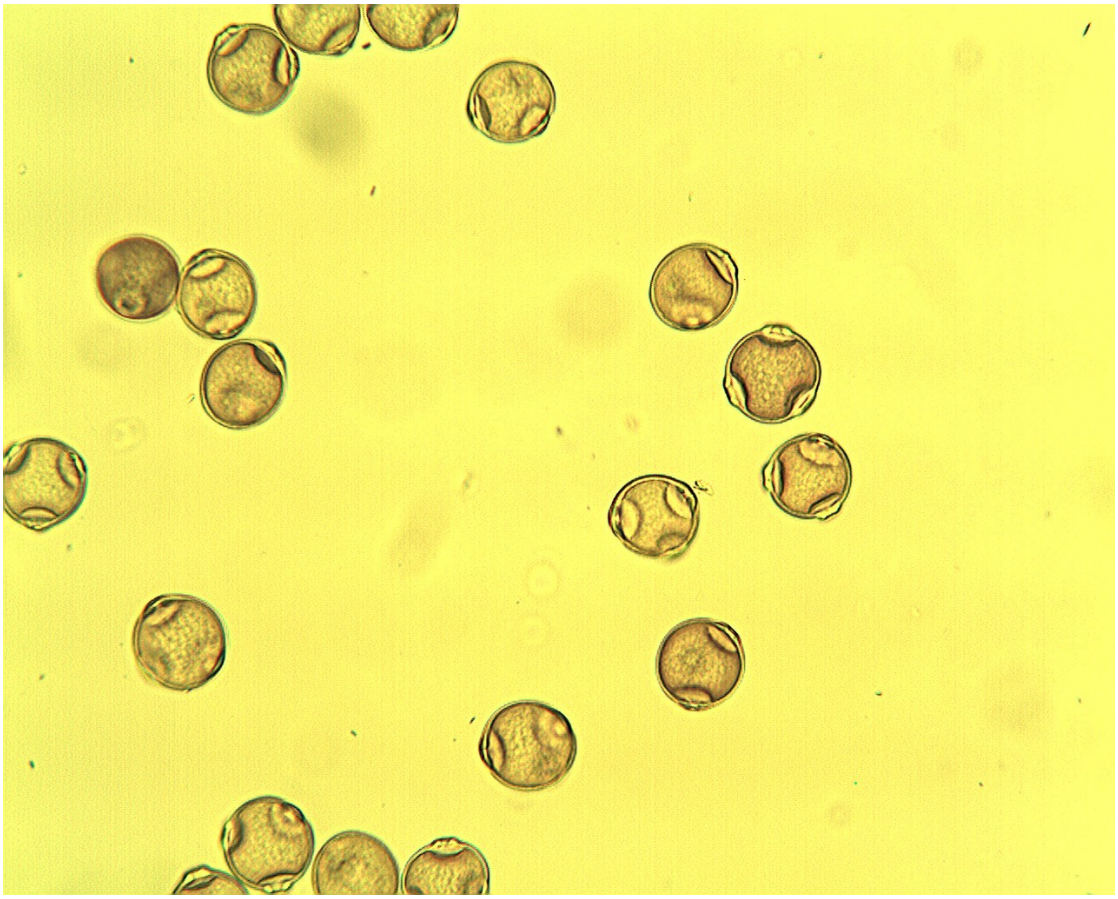


Abbildung 4.2.: Aufnahme in 400facher Vergrößerung. Es sind einige Birkenpollenkörner zu sehen.

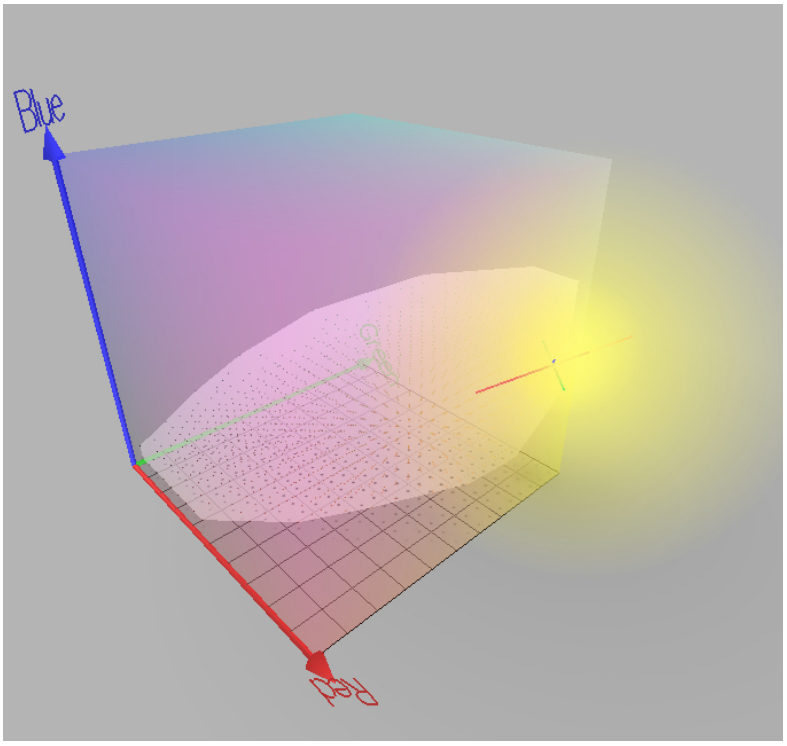


Abbildung 4.3.: 3D-Histogramm von Abbildung 4.2. Es überwiegen Gelbtöne unterschiedlicher Helligkeit. Erstellt mit *Coleur.org*.

4.1.3. Invertieren

Dieser Schritt ist nicht direkt notwendig, entspricht aber der Denkweise, dass ein hoher Intensitätswert im Bild für eine hohe Wahrscheinlichkeit spricht, dass an dieser Stelle ein gesuchtes Objekt liegt. So werden auch direkt beim Laden alle Intensitätswerte mit der Formel $I_{neu}(x, y) = 1 - I(x, y)$ invertiert.

4.1.4. Glätten

Die Aufnahmen weisen ein geringes Rauschen auf. Um dieses auszugleichen, bietet es sich an, einen leichten Glättungsfilter auf das Bild anzuwenden. Da für die Segmentierung der Erhalt der Kanten im Bild relevant ist, wurde statt eines einfachen Gauß-Tiefpassfilters ein Medianfilter mit Radius 3 angewandt, wie in Kapitel 2.2.11 beschrieben. Wichtig ist jedoch, dass nur für die Segmentierung geglättet wird, für die Merkmalsgewinnung aber wieder das Originalbild verwendet wird, da ein Glättungsfilter stark die Textureigenschaften verändert.

4.2. Segmentierung

4.2.1. Lage der Pollenkörner im Bild

Der Grundgedanke bei der Segmentierung der Pollenkörner war, dass es sich sehr vereinfacht um Kreise handelt. Wenn man also das Bild mit den Pollenkörnern mit dem Bild eines Kreises faltet, der ungefähr so groß ist wie die Pollenkörner, hat das Ergebnisbild überall dort hohe Intensitätswerte, wo Kreis und Pollenkorn sich gut überdecken. Siehe Kapitel 2.2.3 und [Mül06].

Wenn man die vorliegenden Aufnahmen betrachtet, kann man den durchschnittlichen Durchmesser der Pollenkörner messen. Bei den Aufnahmen in 100facher Vergrößerung schwankte dieser ziwschen 23 und 28 Pixeln, so dass für die Faltung ein Kreis mit Radius von 13 Pixeln gewählt wurde. Bei den Aufnahmen in 400facher Vergrößerung hingegen bewegten sich die Durchmesser zwischen 95 und 123 Pixeln. Es ergab sich ein Kreis mit einem Radius von 55 Pixeln. Die Ergebnisse der Faltung sind in Abbildung 4.4 und Abbildung 4.5 zu sehen.

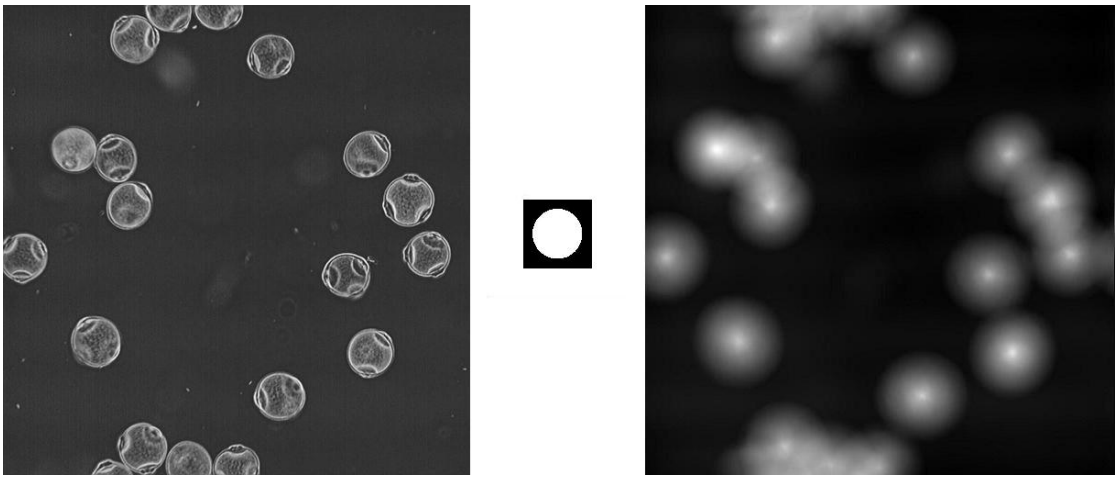


Abbildung 4.4.: Originalbild in 400facher Vergrößerung, Kreis mit Radius 55 und Ergebnis der Korrelation

Wie in Abschnitt 2.2.7 erklärt, ist das Ergebnisbild einer Faltung größer als die beiden Eingabebilder. Deshalb wird aus dem Ergebnisbild ein Bild ausgeschnitten, welches genauso groß ist, wie die Aufnahme der Pollenkörner und dessen oberste linke Ecke an den Koordinaten (Breite des Kreisbildes / 2; Höhe des Kreisbildes / 2) liegt. Das reine Faltungsergebnis ist in Abbildung 4.6 zu sehen.

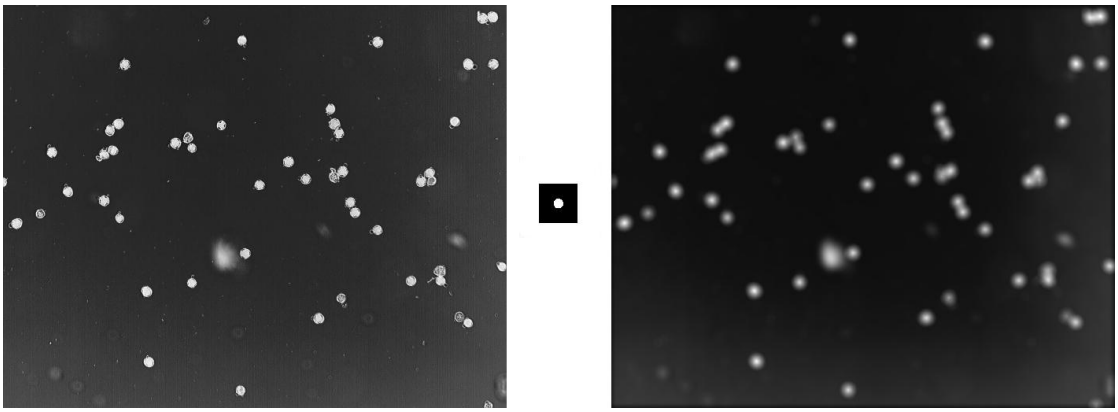


Abbildung 4.5.: Originalbild in 100facher Vergrößerung, Kreis mit Radius 23 und Ergebnis der Korrelation

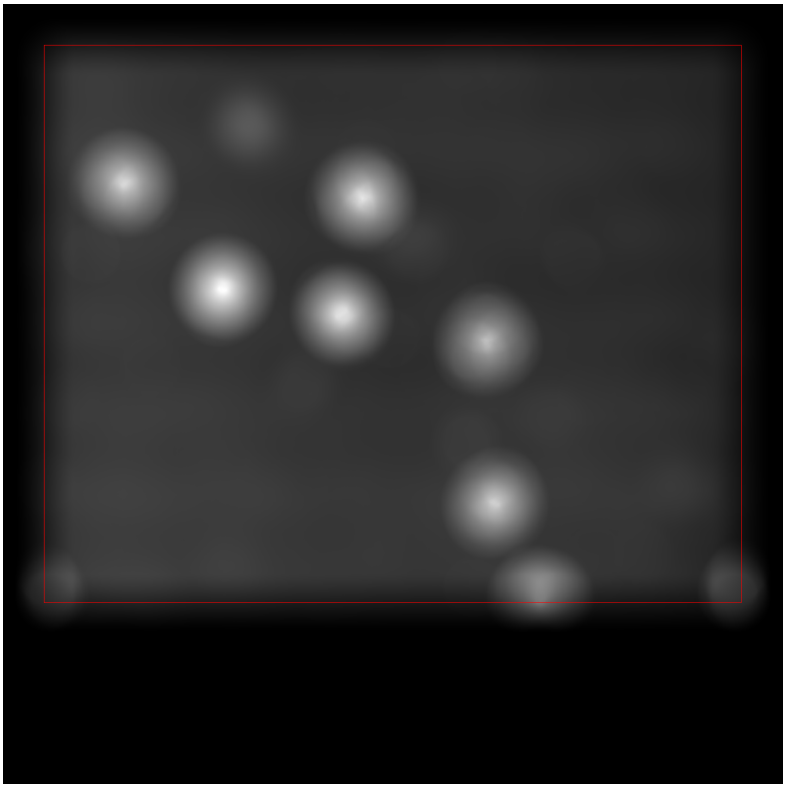


Abbildung 4.6.: Eigentliches Faltungsergebnis mit relevantem Ausschnitt

Das erzeugte Bild hat nun überall dort besonders hohe Intensitäteswerte, wo man den Mittelpunkt eines Pollenkorns erwarten kann. Als nächstes gilt es, diese Mittelpunkte zu finden, um sie als Ausgangspunkte für die Segmentierung zu benutzen. Hierzu wird das in Kapitel 2.2.8 beschriebene Verfahren zum Auffinden regionaler Maxima verwendet. Ein Schwellwert von 7,0 hat sich hierbei als sehr erfolgreich bei den Aufnahmen in 400facher Vergrößerung erwiesen. Bei den Aufnahmen in 100facher Vergrößerung war ein Festlegen des Wertes deutlich komplizierter, da selbst bei sehr geringem Schwellwert noch Pollenkörner ausgelassen wurden. Ein Schwellwert von 3,0 stellt einen guten Kompromiss zwischen Überselektion und falschem Zurückweisen dar, da hier bei über 50 Pollenkörnern nur 3 nicht erkannt wurden. Die Ergebnisse sind in Abbildung 4.7 und Abbildung 4.8 zu sehen.

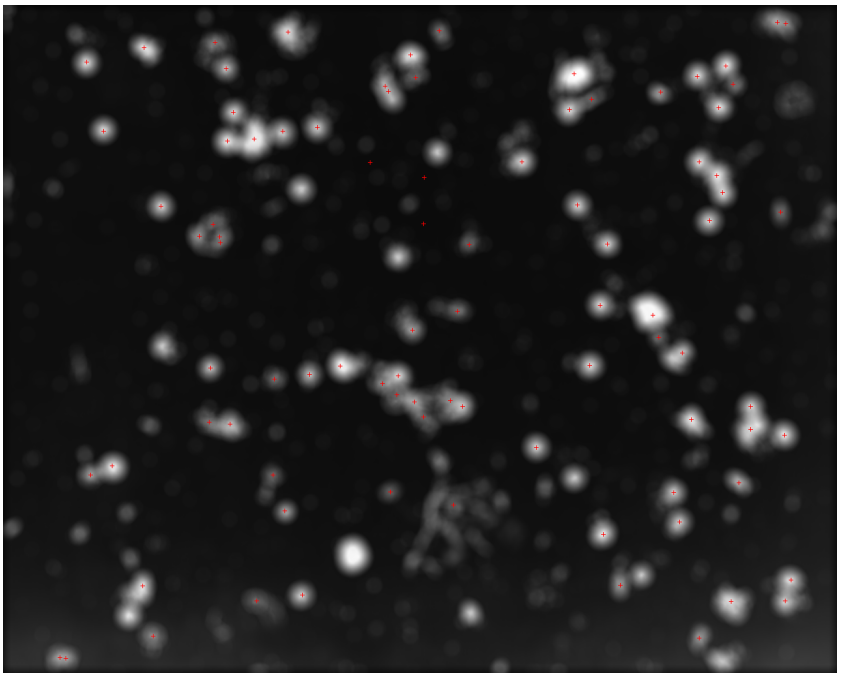


Abbildung 4.7.: Gefundene Maxima im Faltungsergebnis bei Schwellwert $3,0$.

4.2.2. Kandidaten ausschneiden

Nachdem nun die Mittelpunkte möglicher Pollenkörner (im folgenden *Kandidaten*) bekannt sind, müssen diese mit einer deutlichen Kontur ausgeschnitten werden, so dass möglichst keine Pixel des Hintergrunds oder anderer Objekte darin liegen. Nur so kann man sicher sein, dass die später berechneten Merkmale dieser Kandidaten sich auch wirklich auf die Eigenschaften der Kandidaten beziehen und nicht auch auf den Hintergrund oder ihre Umgebung, welche beide nicht signifikant sind.

Hier wurden verschiedene Ansätze ausprobiert.

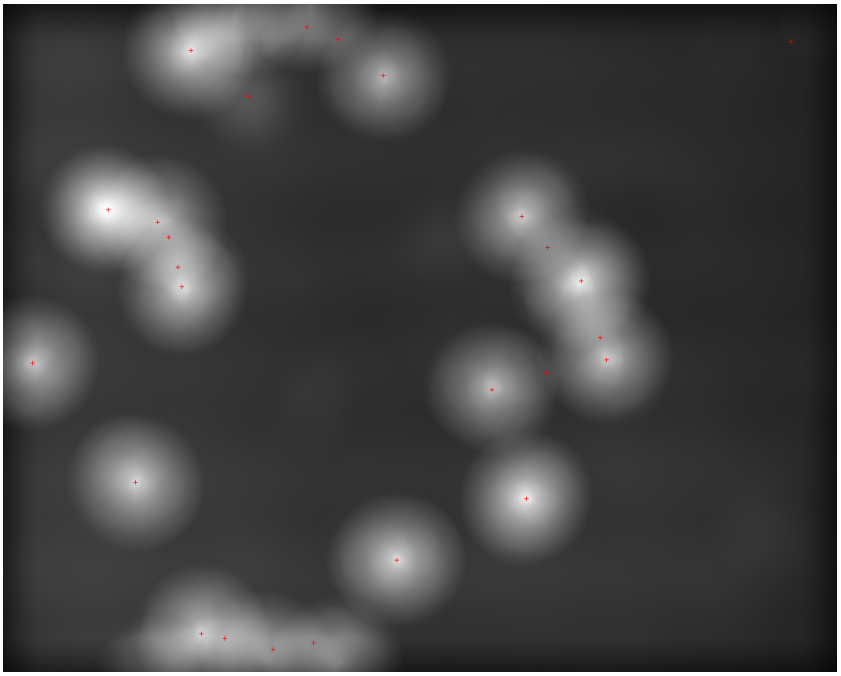


Abbildung 4.8.: Gefundene Maxima im Faltungsergebnis bei Schwellwert $7,0$.

Ansatz 1: Schwellwert

Der einfachste Ansatz war, je eine Umgebung um einen Kandidaten-Mittelpunkt auszuschneiden, welche einige Pixel größer ist als der durschnittliche Pollenradius. Danach wird der Hintergrund des Bildes mit einem Schwellwert bezüglich der Intensitätswerte ausgeblendet. Den Schwellwert erhält man über das in Kapitel 2.2.9 beschriebene *Mixture-Modeling*. Allerdings muss der Schwellwert für das Ausgangsbild berechnet werden und nicht für den Ausschnitt

Deutlicher Nachteil dieses Ansatzes ist jedoch, dass bei dichter Lage der Pollenkörner zwar der Hintergrund korrekt weggeschnitten wird, nicht aber anliegende andere Pollenkörner oder Objekte, zu sehen in Abbildung 4.9. Diese würden so die Berechnung der Merkmale des Kandidaten störend beeinflussen.

Ansatz 2: Snake + Schwellwert + Glätten

Eine kreisrunde Kontur, etwas größer als das durschnittliche Pollenkorn, wird um den Kandidatenmittelpunkt gelegt und dann nach dem Verfahren aus Kapitel 2.2.12 um das Objekt zusammengezogen. Als Bildenergie wird bei diesem Verfahren üblicherweise das Gradientbild nach Sobel verwendet. Um die Kanten an den Rändern der Objekte zu verstärken wird zunächst das Originalbild mit einem Gauß-Tiefpassfilter (siehe Abschnitt 2.2.3, $\sigma = 1,4, \nu = 0$) etwas geglättet und anschließend wieder mit Hilfe des Mixture-Modeling das Schwellwertbild erzeugt und von diesem dann das Gradientenbild berechnet

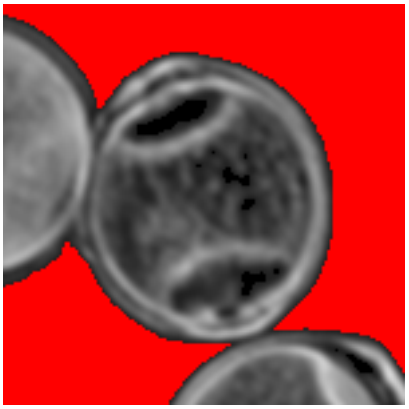


Abbildung 4.9.: Ausschnit aus dem Ergebnisbild des Schwellwertverfahren mit Mixture-Modeling. Der ausgeschnittene Bereich ist rot markiert.

und als Bildenergie verwendet. Ein Vergleich des normalen Gradientenbildes mit dem zuvor mit einem Schwellwert behandelten ist in Abbildung 4.10 zu sehen.

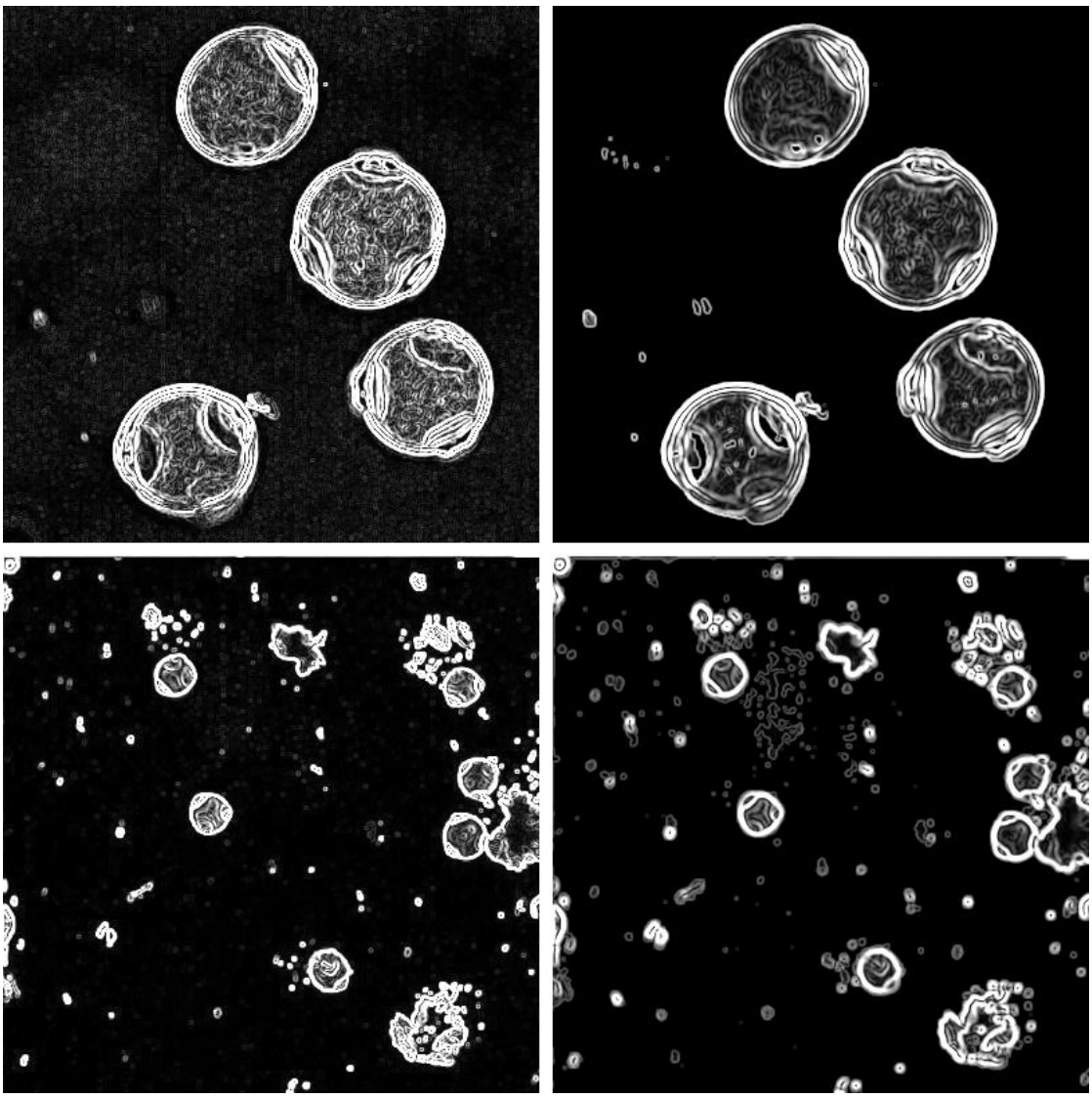


Abbildung 4.10.: Auf der linken Seite Gradientbilder nach Faltung mit dem Sobel-Operator und auf der rechten Seite nach vorheriger Glättung und Schwellwertverfahren. Oben bei 400facher und bei 100facher Vergrößerung. Bei den vorbehandelten Bildern sind die äußeren Konturen der Objekte deutlich hervorgehoben.

Ansatz 3: Snake + Canny

Am erfolgreichsten war es jedoch, nicht das einfache Gradientenbild nach Sobel als Bildenergie für die Snake zu benutzen, sondern das Gradientenbild nach Canny, wie in Kapitel 2.2.10 beschrieben. Vorteil ist, dass nur sehr dünne und exakte Kanten geliefert werden und die Snake so wirklich nur von den Rändern der Objekte angezogen wird und nicht von schwächeren Artefakten in deren Umgebung. Bei den Bildern in 400facher Vergrößerung ist das Kantenbild nach Canny jedoch noch klarer und die Kanten geschlossener als bei 100facher Vergrößerung. Siehe Abbildung 4.11.

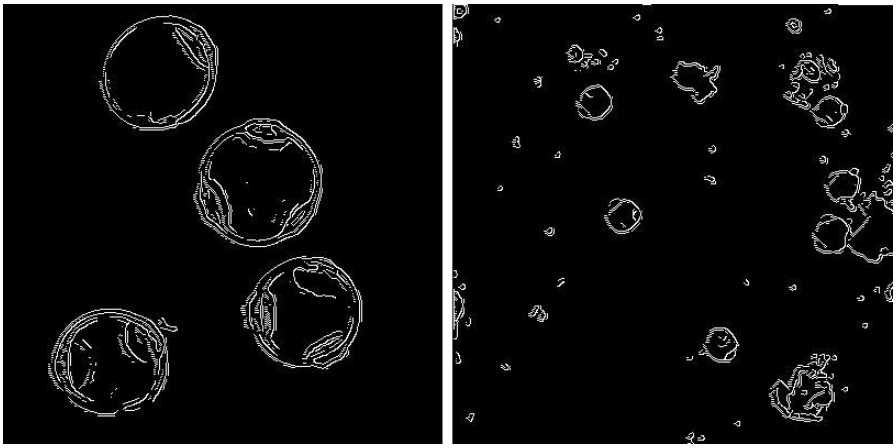


Abbildung 4.11.: Kantenbild nach Canny bei 400facher (links) und 100facher Vergrößerung (rechts).

Wie beim voherigen Ansatz wird nun um jeden Kandidatenmittelpunkt ein Bild der Größe 160 x 160 beziehungsweise 64 x 64 Pixel ausgeschnitten. Um den Mittelpunkt wird eine kreisrunder Kontur gelegt, welche durch Punkte im Abstand von initial vier Pixeln festgelegt wird. Diese wird nun nach dem Snake-Verfahren aus Kapitel 2.2.12 immer weiter um das Objekt zusammengezogen und an dessen Kontur angepasst. Die benutzen Parameter hierfür waren $1,0$ für die Gewichtung der Stetigkeitsenergie, $1,5$ für die Gewichtung der Krümmungsenergie und $1,1$ für die Gewichtung der Bildenergie. Gute Ergebnisse zeigten sich nach zirka 50 bis 100 Iterationen der Snake. Die verschiedenen Iterationsstufen für ein einzelnes Pollenkorn sind in Abbildung 4.2.2 sehen, für das ganze Bild sieht man das Ergebnis in Abbildung 4.13.

Zuletzt wird alles außerhalb der Kontur ausgeblendet und nur der innenliegende Bereich des Bildes sowie die Kontur selbst zur Berechnung der Merkmale herangezogen.

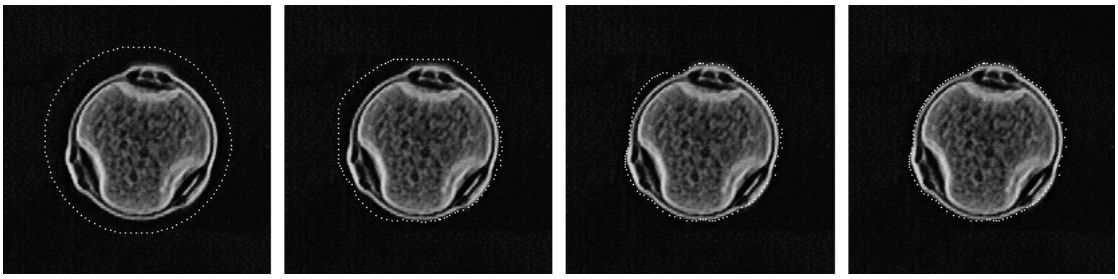


Abbildung 4.12.: Die Snake zieht sich vom vorgegebenen Kreis immer weiter um das zu segmentierende Pollenkorn zusammen. Zustand nach 1, 10, 20 und 40 Iterationen

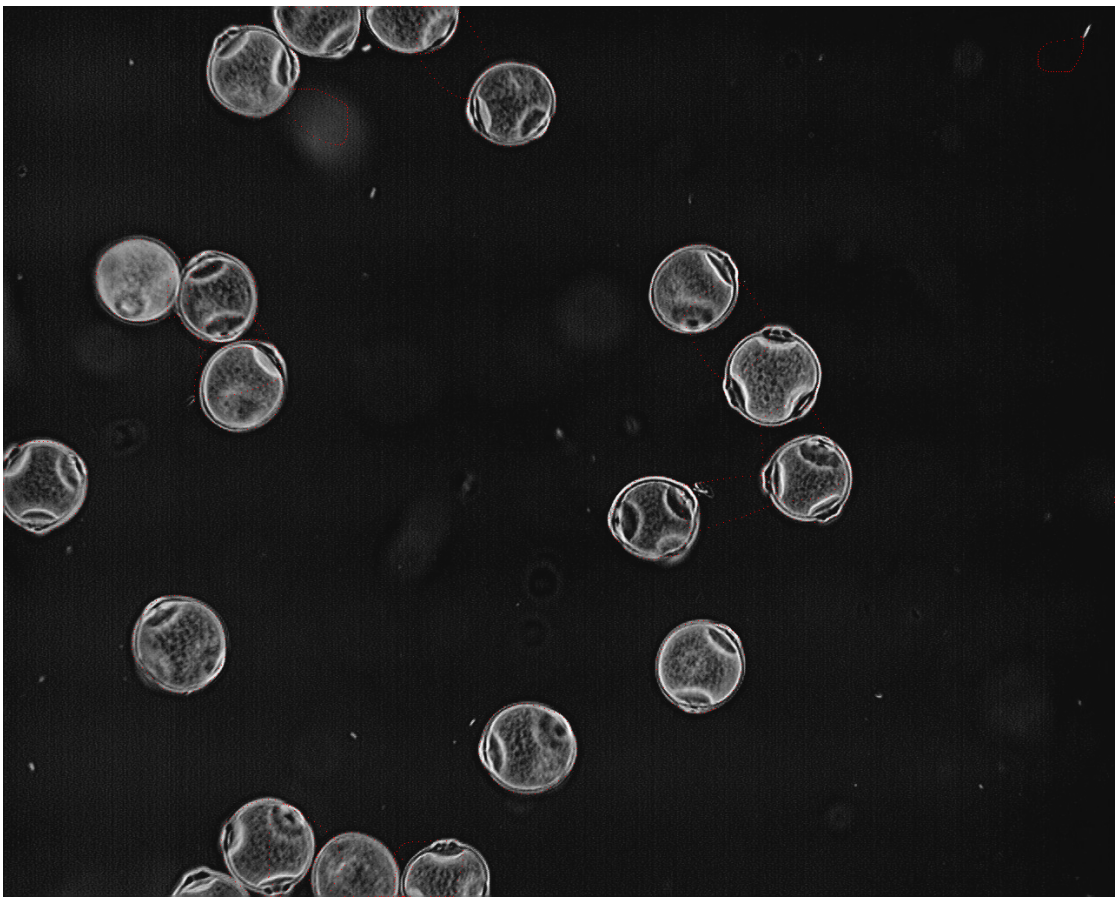


Abbildung 4.13.: Segmentierungsergebnis bei einer Aufnahme 400facher Vergrößerung. Die Snakes haben sich um die Kandidaten zusammengezogen.

4.3. Merkmalsgewinnung

Für alle im letzten Abschnitt segmentierten Kandidaten müssen nun beschreibende Merkmale ermittelt werden, um diese unterscheiden und klassifizieren zu können. Hierfür steht zum einen die Form der Objekte zur Verfügung, welche zuvor durch die Kontur ermittelt wurde. Darüber hinaus sind die Texturen der verschiedenen Objektklassen sehr unterschiedlich und es lohnt sich, diese beschreibenden Merkmale zu berechnen. Die berechneten Merkmale werden im folgenden erläutert.

4.3.1. Konturmerkmale

Merkmale, welche die Form beschreiben, sind schnell und einfach zu berechnen und orientieren sich an Kapitel 2.2.13. Es werden für die Kontur jedes Kandidaten dessen Umfang, die Fläche, das Verhältnis Flache zu Umfang, die Abweichung des Umfangs von einem Kreis gleicher Fläche und die durchschnittliche Krümmung der Kontur berechnet. Für die Unterscheidung zwischen Pollenkörnern und Artefakten sind die Merkmale Fläche und Verhältnis Fläche zu Umfang sehr gut geeignet, da die Pollenkörner alle sehr rund sind und ungefähr die gleich Größe haben, wo hingegen Artefakte auch kleiner und größer ausfallen können sowie länglicher oder ausgefranster. Eine genaue Unterscheidung zwischen den verschiedenen Klassen sowie signifikanten und nicht signifikanten Merkmalen findet sich im nächsten Kapitel 4.4.

4.3.2. Texturmerkmale

Bei den Merkmalen, welche die Textur der Objekte beschreiben, wurden hauptsächlich die Merkmale nach Haralick [HDS73] aus Kapitel 2.2.13 verwendet. Da dieses Verfahren bereits für einen gewählten Pixelabstand 28 Merkmale zur Verfügung stellt, wurden ausschließlich die Werte für einen Grauwertmatrix für einen Abstand von einem Pixel berechnet. Genau wie in Kapitel 2.2.13 beschrieben, werden von den 4×14 Merkmalen je der Mittelwert und die Intervallgröße über die Richtungen verwendet. Als weiteres Merkmal wurde nur noch die mittlere Intensität der Pixel des Kandidaten hinzugenommen.

4.4. Fuzzy-Zerlegung

Bis hierher liegen für jeden Kandidaten 32 beschreibende Merkmale vor. Um eine Wissenbasis für den Fuzzy-Controller aufzubauen, muss bekannt sein, welche Arten von Objekten welche Eigenschaften aufweisen. Zu diesem Zweck wurden alle gefundenen Kandidaten manuell klassifiziert. Bei den Aufnahmen in 400facher Vergrößerung wurden in sieben Originalaufnahmen und zwei Testbildern insgesamt 108 Kandidaten gefunden. Diese wurden von Hand einer der folgenden Klassen zugeordnet: Hintergrund, Artefakt, Birke(npolle), schlecht segmentierte Birkenpolle (Birke_kaputt), Olive(npolle) und schlecht segmentierte Olivenpolle (Olive_kaputt). Beispiele für jede Klasse sind in Abbildung 4.14 zu sehen.

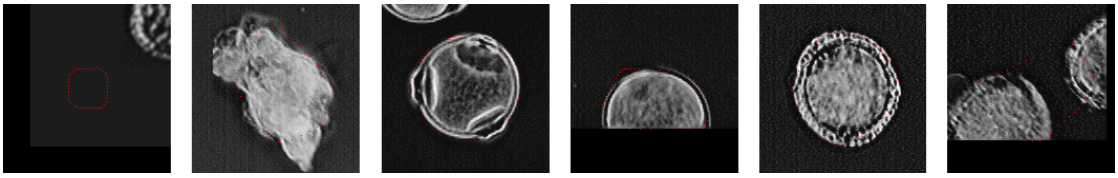


Abbildung 4.14.: Beispiele für die sechs verschiedenen Klassen. Von links nach rechts: Hintergrund, Artefakt, Birke, Birke_kaputt, Olive und Olive_kaputt

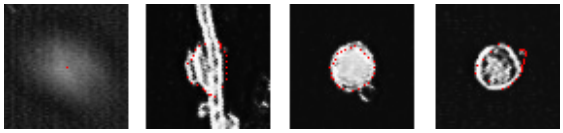


Abbildung 4.15.: Beispiele für die sechs verschiedenen Klassen. Von links nach rechts: Hintergrund, Artefakt, PolleA, PolleB

Bei den Aufnahmen in 100facher Vergrößerung wurden in vier Originalaufnahmen insgesamt 153 Kandidaten gefunden. Die zugeordneten Klassen lauten: Hintergrund, Artefakt, PolleA und PolleB. Da die Aufnahmen nicht beschriftet und die Vergrößerung zu schlecht für eine manuelle Bestimmung waren, konnte nur zwischen zwei Pollenarten unterschieden werden. Die Beispiele für die Klassen finden sich in Abbildung 4.15.

Die Verteilung der Werte der Merkmale der verschiedenen Klassen kann man am besten grafisch darstellen. Aufgrund der Fülle von Daten wurden diese Grafiken in die Kapitel A.1 und A.2 ausgelagert. Exemplarisch sind hier die Verteilung der Merkmale `HARALICK_5_MEAN_INVERSE_DIFFERENCE_MOMENT` bei 400facher Vergrößerung und `HARALICK_3_MEAN_CORRELATION` bei 100facher Vergrößerung zu sehen, siehe Abbildung 4.16.

Einige Merkmale wie zum Beispiel `HARALICK_7_RANGE_SUM_VARIANCE` bieten kaum eine Möglichkeit, zwischen den Klassen zu unterscheiden, wohingegen andere wie `HARALICK_3_RANGE_CORRELATION` oder `KONTUR_UMFANG` bezüglich mindestens einer Klasse signifikant sind. Die nicht signifikanten Merkmale wurden im Folgenden nicht mehr berücksichtigt und für die Wissensbasis wurden nur zu einigen Merkmalen, die eine deutliche Unterscheidung zulassen, Fuzzy-Mengen erzeugt. Im speziellen sind dies die Merkmale:

1. `KONTUR_FLAECHE` Fläche des segmentierten Objekts
2. `HARALICK_1_MEAN_ANGULAR_SECOND_MOMENT`
3. `HARALICK_1_RANGE_ANGULAR_SECOND_MOMENT`
4. `HARALICK_4_MEAN_SUM_OF_SQUARES`
5. `HARALICK_5_MEAN_INVERSE_DIFFERENCE_MOMENT`
6. `HARALICK_5_RANGE_INVERSE_DIFFERENCE_MOMENT`

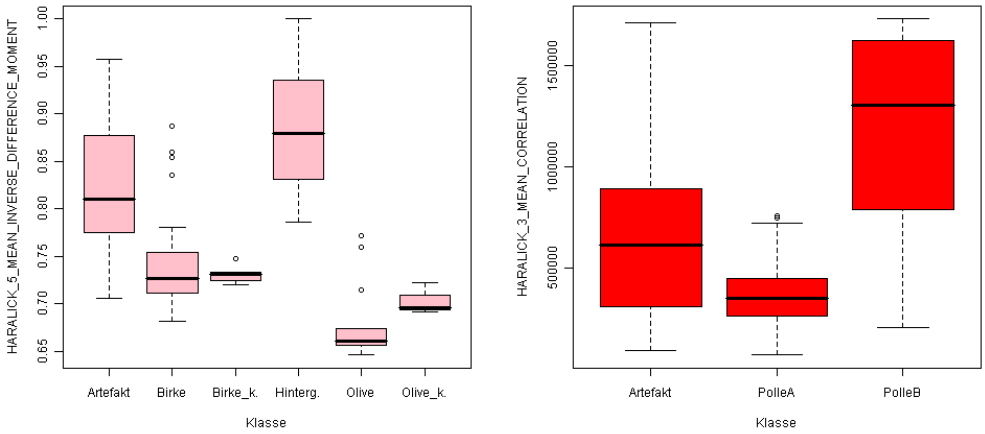


Abbildung 4.16.: Verteilung der Merkmale `HARALICK_5_MEAN_INVERSE_DIFFERENCE_MOMENT` bei 400facher Vergrößerung und `HARALICK_3_MEAN_CORRELATION` bei 100facher Vergrößerung.

Die dazugehörigen Fuzzy-Mengen wurden manuell erzeugt und sind in Abbildung 4.17 bis 4.22 zu sehen. Alle Fuzzy-Mengen werden durch Eckpunkte ihrer Zugehörigkeitsfunktion festgelegt, zwischen denen linear interpoliert wird.

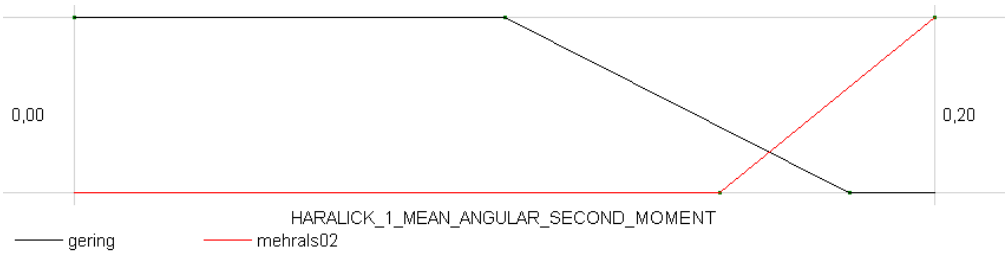


Abbildung 4.17.: Die Fuzzy-Mengen der linguistischen Variable HARALICK_1_MEAN_ANGULAR_SECOND_MOMENT

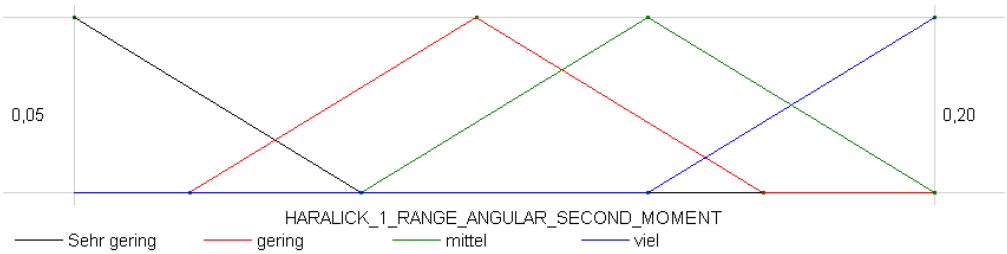


Abbildung 4.18.: Die Fuzzy-Mengen der linguistischen Variable HARALICK_1_-RANGE_ANGULAR_SECOND_MOMENT

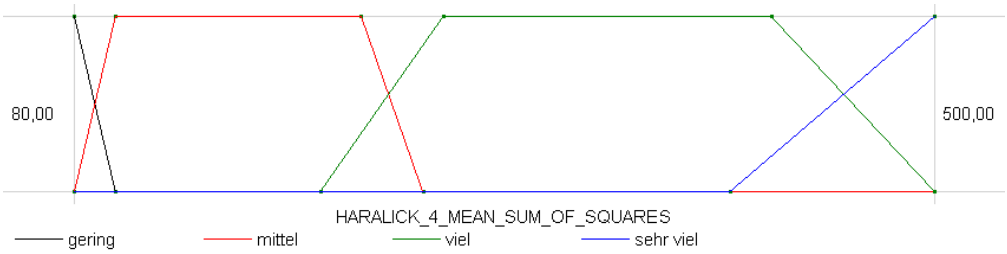


Abbildung 4.19.: Die Fuzzy-Mengen der linguistischen Variable HARALICK_4_-MEAN_SUM_OF_SQUARES

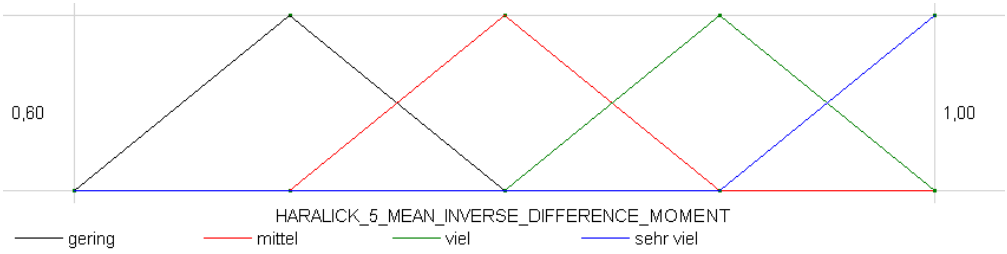


Abbildung 4.20.: Die Fuzzy-Mengen der linguistischen Variable HARALICK_5_-MEAN_INVERSE_DIFFERENCE_MOMENT

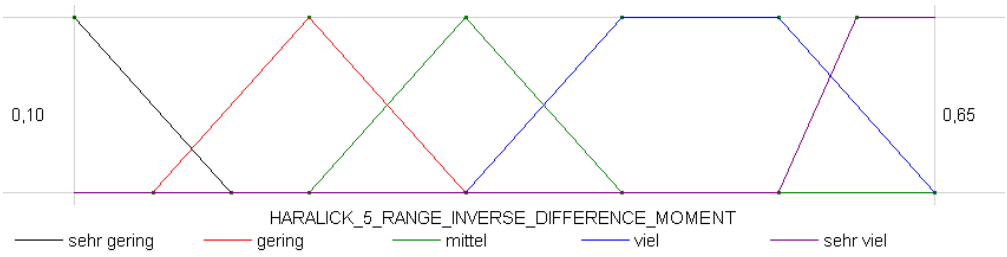


Abbildung 4.21.: Die Fuzzy-Mengen der linguistischen Variable HARALICK_5_-RANGE_INVERSE_DIFFERENCE_MOMENT

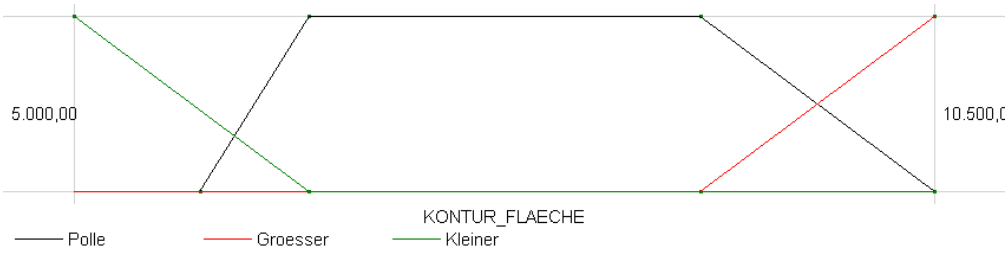


Abbildung 4.22.: Die Fuzzy-Mengen der linguistischen Variable KONTUR_FLAECHE

4.5. Fuzzy-Regeln

Alle Regeln bestehen aus einer Prämisse und einer Konsequenz. Die Konsequenz ist immer eine Fuzzy-Menge zu der linguistischen Variable *Klasse*. Die Prämisse setzt sich aus einer oder mehreren Fuzzy-Mengen zusammen, die allerdings alle zu unterschiedlichen linguistischen Variablen gehören. Die Fuzzy-Mengen der Prämisse werden alle über eine t-Norm verknüpft, was einer Konjunktion ähnelt. Eine Negation ist im Programm selbst nicht vorgesehen, kann aber realisiert werden, indem man die entsprechenden Fuzzy-Mengen manuell erzeugt. Ebenso ist keine Disjunktion vorgesehen. Um diese umzusetzen, kann man aber entsprechend viele Regeln erzeugen

Die Regeln ergaben sich teilweise durch bloßes „hinschauen“ auf die Merkmalsverteilung. Auch recht aufschlussreich war der mit RapidMiner erzeugte Entscheidungsbaum aus Abbildung 4.23.

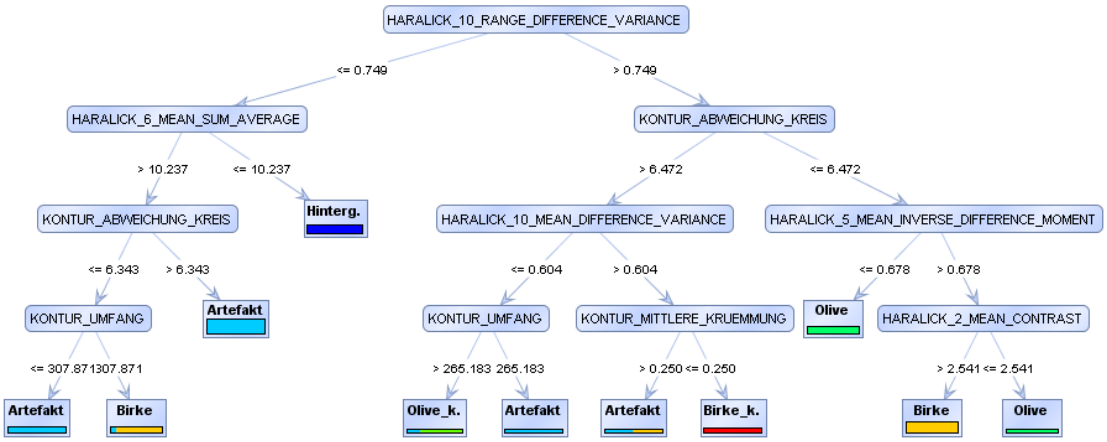


Abbildung 4.23.: Ein mit dem Programm RapidMiner erzeugter Entscheidungsbaum bezüglich der Klasse

Bei Betrachtung der Verteilung der Merkmalswerte ergaben sich die folgenden Regeln:

- HARALICK_4_MEAN_SUM_OF_SQUARES = mittel

HARALICK_5_MEAN_INVERSE_DIFFERENCE_MOMENT = gering
HARALICK_5_RANGE_INVERSE_DIFFERENCE_MOMENT = sehr viel
=> KLASSE = Olive

- HARALICK_1_MEAN_ANGULAR_SECOND_MOMENT = mehrals02
  => KLASSE = Hintergrund

- HARALICK_1_RANGE_ANGULAR_SECOND_MOMENT = viel
  => KLASSE = Hintergrund

- HARALICK_5_MEAN_INVERSE_DIFFERENCE_MOMENT = mittel
  => KLASSE = Birke

- HARALICK_4_MEAN_SUM_OF_SQUARES = mittel
  HARALICK_5_RANGE_INVERSE_DIFFERENCE_MOMENT = viel
  => KLASSE = Birke

4.6. Klassifikation

Zur Klassifikation wurde ein Mamdami-Controller wie in Kapitel 2.3.6 verwendet, welcher allerdings leicht modifiziert wurde. Der Mamdani-Controller gibt normalerweise scharfe (Dezimal-) Zahlwerte aus. Hier ist jedoch eine Klasse gesucht. Dies wurde umgangen, indem eine weitere linguistische Variable „Klasse“ eingeführt wurde und zu jeder Klasse eine Fuzzy-Menge angelegt wurde, die nur einen kleinen Ausschlag hat und sich nicht mit anderen überschneidet, zu sehen in Abbildung 4.24.

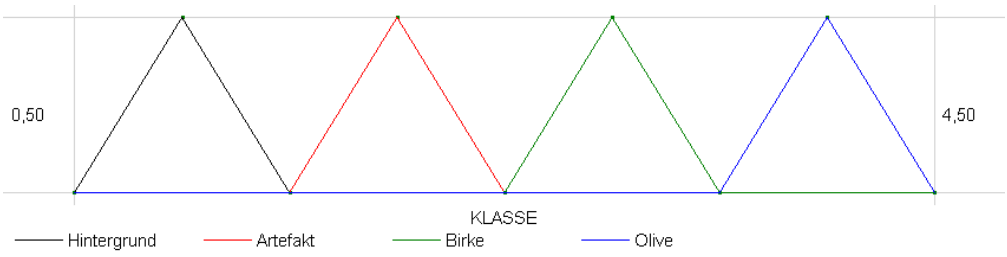


Abbildung 4.24.: Die Fuzzy-Mengen der linguistischen Variable KLASSE

Die Akkumulation funktioniert nun nicht mehr über die Schwerpunktsmethode, sondern über die Maximum-Methode. Allerdings wird statt des scharfen Ergebniswerts selbst der Name der Fuzzy-Menge ausgegeben, zu der er gehört. Der Wert wird also wieder bezüglich der linguistischen Variable „Klasse“ fuzzyfiziert. Am Ende erhält man so zu jedem Objekt aus der Mikroskopieaufnahme dessen endgültige Klassifikation.

4.6.1. Güte der Klassifizerung

Ein Verfahren wie Kreuzvalidierung anzuwenden, um die Güte der Klassifizerung zu bewerten, ergibt keinen Sinn, da das Klassifikationsschema nicht automatisch generiert sondern explizit manuell eingeben wird. Mit den angegebenen Fuzzy-Mengen und -Regeln ließen sich ca. 85 Prozent der gefundenen Objekte korrekt klassifizieren. Würde man die Regeln erweitern oder die Mengen verfeinern, ließe sich wahrscheinlich eine noch höhere Güte erzielen. Jedoch muss man beachten, dass Ausreißer in den Daten nicht abgefangen werden können und so die Regeln verfälschen oder die Klassifikationsgüte durch *false-positives* senken.

4.7. Das Programm Pollendetektor

Im Rahmen dieser Arbeit entstand das Programm *Pollendetektor*. Es stellt die Funktionalität bereit, die zum Lösen der Klassifikationsaufgabe benötigt wird. Das Programm besteht aus einem Hauptfenster zum Laden und Verarbeiten der Bilder und einem Fenster, in welchem die Wissensbasis angepasst werden kann. Existiert bereits eine Wissensbasis, besteht die generelle Vorgehensweise zum Bearbeiten einer Aufnahme aus den Schritten `Laden des Bildes`, `Segmentierung` und `Klassifikation`.

Sollte noch keine Wissensbasis existieren, so muss zunächst eine Trainingsmenge von Aufnahmen geladen, segmentiert und ihre Merkmale exportiert werden. Letzteres geschieht automatisch standardmäßig in die Datei *C:\statfile.csv*. Eine andere Ausgabedatei kann unter dem Menüpunkt *Datei* → *Ausgabedatei festlegen* gewählt werden. Des Weiteren wird jedes bei der Segmentierung gefundene Objekt als Bilddatei gespeichert.

Nun muss außerhalb des Pollendetektors manuell klassifizert werden. Hilfreich ist hier, die Tabelle in ein Tabellenkalkulationsprogramm wie *OpenOffice.org Calc* oder *Microsoft Excel* zu importieren und um eine Spalte `Klasse` zu erweitern. Mit Hilfe von Statistik- oder Data-Mining-Tools wie *R* beziehungsweise *RapidMiner* lassen sich nun signifikante Merkmale finden und mögliche Fuzzy-Zerlegungen abschätzen.

Diese manuell gewonnenen Erkenntnisse lassen sich nun wieder im Programm Pollendetektor im Fenster der Wissenbasis in Fuzzy-Mengen und Fuzzy-Regeln umsetzen. Eine genaue Beschreibung der Fenster und Funktionen folgt weiter unten.

Es ist zu beachten, dass ein Großteil der Funktionen Parameter benötigt. Da es sich beim Pollendetektor nicht um ein Produkt sondern um eine Testanwendung handelt, sind viele dieser Parameter nicht über Benutzerdialoge einstellbar, sondern werden im Quelltext des Programms definiert.

4.7.1. Funktion

An dieser Stelle soll kurz der Aufbau und die Funktionsweise des Programms Pollendetektor anhand seiner Fenster erläutert werden.

Segmentierung

Das Hauptfenster in Abbildung 4.25 besteht aus einem Anzeigebereich für das aktuell bearbeitete Bild, einem Log-Bereich, welcher die durchgeführten Operationen auflistet, einer Liste der gefundenen Segmente und den Knöpfen *Original zeigen*, *Segmentiere*, *Klassifiziere* und *Log löschen*, sowie einer Menü- und einer Statusleiste.

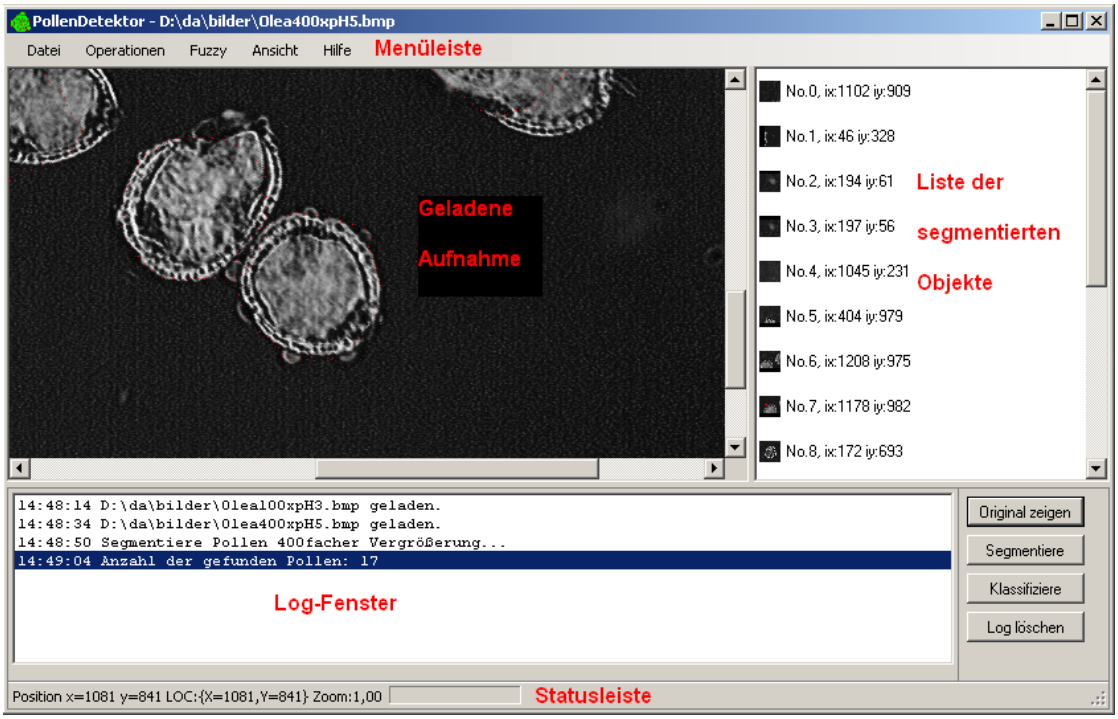


Abbildung 4.25.: Hauptfenster des Programms Pollendetektor

Der Knopf *Segmentiere* führt eine Segmentierung für Aufnahmen in 400facher Vergrößerung durch. Alle gefundenen Objekte werden rechts mit einem Bild und der Position ihres Mittelpunkts aufgelistet. Diese kann man durch Anklicken näher betrachten. Bleibt man mit der Maus darüber stehen, wird ein Tooltip mit den berechneten Merkmalen angezeigt.

Nach der Segmentierung kann man aus der Liste der segmentierten Objekte beliebig viele auswählen und über den Knopf *Klassifiziere* per Mamdami-Controller klassifizieren lassen. Dafür muss allerdings eine Wissenbasis angelgt oder geladen worden sein.

In der Statusleiste wird immer die aktuelle Position der Maus im Bild angegeben. Bei länger andauernden Operationen wie der Segmentierung wird auch ein Fortschrittsbalken angezeigt.

Weitere Operationen sind über das Menü verfügbar, jedoch nicht relevant zur Lösung der Klassifikationsaufgabe. Daher kann man in Abschnitt 4.7.3 eine vollständige Auflistung finden.

Wissensbasis

Die Fuzzy-Mengen und -Regeln, welche der Klassifikation dienen, können im Fenster *Wissensbasis* bearbeitet werden. Dieses wiederum besitzt zwei Karteikarten, eine für die Fuzzy-Mengen und eine für die darauf aufbauenden Fuzzy-Regeln. Siehe dazu auch die Abbildungen 4.26 und 4.27.

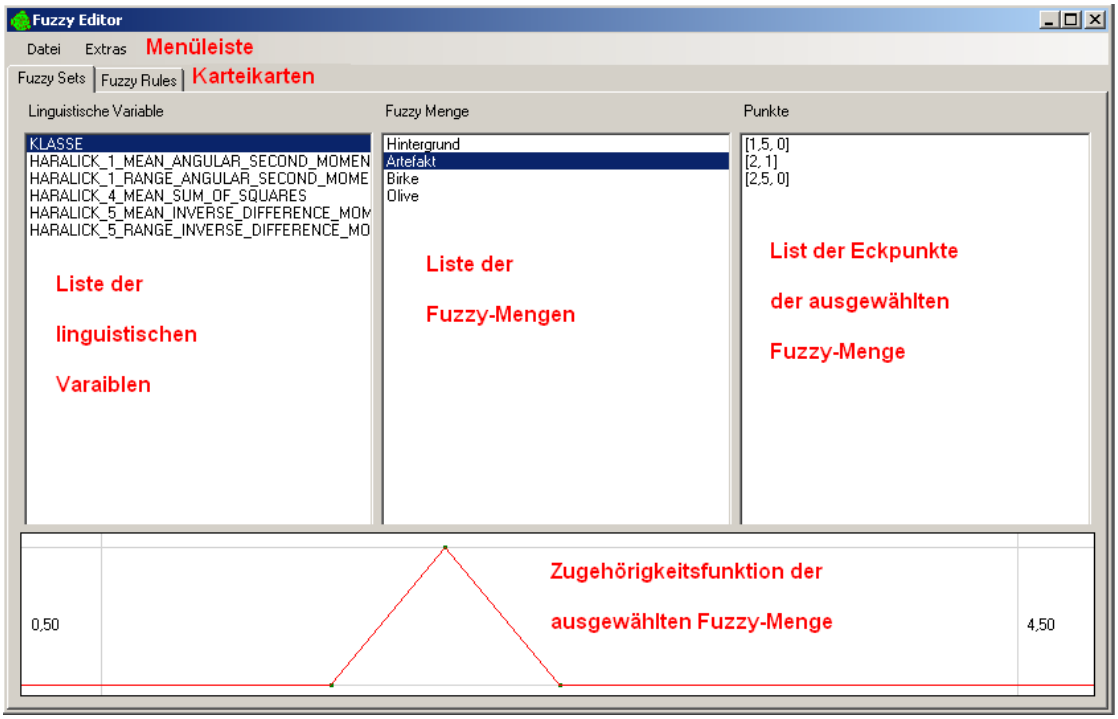


Abbildung 4.26.: Fenster der Wissensbasis, Karteireiter der Fuzzy-Mengen

Auf der Karteikarte *Fuzzy-Mengen* können neue linguistische Variablen oder Fuzzy-Mengen erstellt und bestehende manipuliert werden.

Wählt man eine der linguistischen Varaiblen aus, werden in der Liste rechts daneben alle zugehörigen Fuzzy-Mengen angezeigt. Wählt man eine Fuzzy-Menge aus, werden wiederum in der Liste rechts daneben alle zugehörigen Punkte angezeigt.

Zum Erstellen einer neuen linguistischen Variable, Fuzzy-Menge oder eines Punktes, klickt man jeweils doppelt in die entsprechende Liste und gibt dann einen Namen oder Koordinaten ein. Da dem Benutzer die internen Namen aller Merkmale nicht unbedingt klar sind, kann man diese über den Menüpunkt *Extras* → *Hole Features* in die Liste der linguistischen Variablen einfügen.

Im unteren Teil des Fensters wird die Zugehörigkeitsfunktion der ausgewählten Fuzzy-Menge auch grafisch dargestellt. Dabei entspricht die linke Grenze der kleinsten x-Koordinate eines Punktes über alle Fuzzy-Mengen der ausgwählten linguistischen Variable und die rechte Grenze der größten x-Koordinate. Ein Doppelklick erstellt einen weiteren Eckpunkt an der Position des Cursors. Ein Rechtsklick auf einen bestehenden Punkt

löscht diesen. Klickt man mit der Maus doppelt in den Bereich links beziehungsweise rechts der linken beziehungsweise rechten Grenze, so kann man einen neuen Wert für diese Grenze festlegen und somit den sichtbaren Bereich vergrößern oder verkleinern.

Die Karteikarte *Fuzzy-Regeln*, Abbildung 4.27, erlaubt es dem Benutzer, neue Regeln zu erstellen oder bestehende zu manipulieren. Im oberen Listenfeld werden alle existierenden Regeln angezeigt. In den beiden Feldern darunter sind jeweils Auswahlfelder für die Konsequenz und Prämissen der jeweiligen Regel zu sehen.

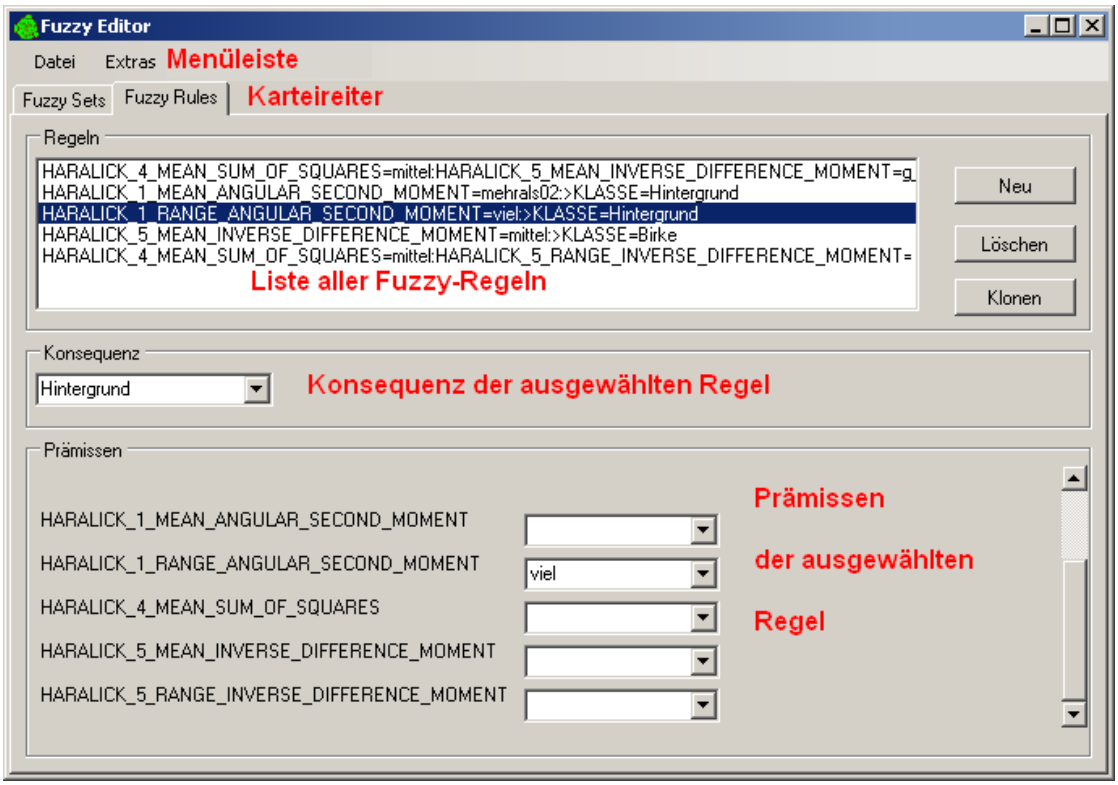


Abbildung 4.27.: Fenster der Wissensbasis, Karteireiter der Fuzzy-Regeln

Über einen Doppelklick in das Listenfeld kann eine neue Regel angelegt werden. Diese ist zunächst leer. Daher wählt man aus dem oberen Listfeld eine Konsequenz und aus allen Auswahlfeldern für die Prämissen die jeweilig benötigten. Für jede linguistische Variable wird ein Auswahlfeld mit allen Fuzzy-Mengen angezeigt. Somit sind nur Regeln möglich, welche aus einer Konjunktion der einzelnen Prämissen bestehen. Eine Disjunktion ist realiserbar, indem man mehrere Regeln anlegt, welche später vom Mamdani-Controller automatisch per Aggregation zusammengefasst werden. Dies entspricht dann einer Disjunktion.

4.7.2. Dateiformate

Das Programm Pollendetektor verwendet zwei Dateiformate. Zum einen, um die Ergebnisse der Segmentierung und Merkmalsgewinnung abzuspeichern und zum anderen für die Wissensbasis mit ihren Fuzzy-Mengen und -Regeln.

Merkmale

Die während der Segmentierung errechneten Merkmale jedes einzelnen Segments werden zur Weiterverarbeitung in eine Tabelle in eine *.csv*-Datei geschrieben. Die Endung *.csv* steht für *Comma Separated Value*. Diese Dateien sind so aufgebaut, dass eine Zeile einer Datenreihe entspricht und die einzelnen - meist numerischen - Werte durch ein bestimmtes Zeichen voneinander getrennt werden. Hier wird das Semikolon „;“ als Trennzeichen verwendet.

Die Namen der Spalten der Tabelle ergeben sich aus dem Dateinamen des Segments und den errechneten Merkmalen. Folgen Segmente aus verschiedenen Aufnahmen aufeinander, werden diese durch eine Leerzeile und eine weitere Zeile mit allen Spaltennamen getrennt.

Die verwendeten Spaltennamen lauten `DATEI`, `KLASSE` sowie die Namen aller berechneten Merkmale, exakt wie in Tabelle 4.4. Ein sehr knappes Beispiel mit nur drei Merkmalen wäre

```
DATEI;KONTUR_UMFANG;KONTUR_FLAECHE;KONTUR_FLAECHE_DURCH_UMFANG
Olea400xpH5.bmp_seg000.bmp;126,260375933295;1271;10,066499411276
Olea400xpH5.bmp_seg001.bmp;238,373419688102;3798;15,9329844953748
Olea400xpH5.bmp_seg002.bmp;184,957413444754;2155;11,6513307569782
Olea400xpH5.bmp_seg003.bmp;191,960616406044;2430;12,6588466191417
```

Wissensbasis

Die Wissensbasis wird in einer frei wählbaren Datei mit der Endung *.fkb* für *Fuzzy-Knowledge-Base* hinterlegt. Die Datei wird als ASCII-Text gespeichert und ist folgendermaßen aufgebaut:

```
#FUZZY-KB, V1.0}
#SETS:{n=Anzahl der Mengen}
{n Zeilen mit einer Fuzzy-Menge pro Zeile}
#RULES: {m=Anzahl der Regeln}
{m Zeilen mit einer Fuzzy-Regel pro Zeile}
```

Zeilen, welche eine Fuzzy-Menge beschreiben, haben das Format

```
{Name der ling. Variable}:{Name der Menge}:{Anzahl der Punkte}:
{X1}:{Y1}: {...} : {Xn}:{Yn}:
```

Da der Doppelpunkt „:“ als Trennzeichen verwendet wird, dürfen die Namen der linguistischen Variablen und der Fuzzy-Mengen keine Doppelpunkte enthalten.

Die Fuzzy-Regeln werden in der Form

```
{Prämisse 1}: ... :{Prämisse n}:>{Konsequenz}
```

gespeichert. Eine einzelne Prämisse und die Konsequenz werden je als

```
{linguistische Variable}={Name}
```

geschrieben.

Ein sehr einfaches aber korrektes Beispiel mit zwei Fuzzy-Mengen und einer Fuzzy-Regel sieht so aus:

```
#FUZZY-KB, V1.0
#SETS:2
KLASSE:Hintergrund:3:0,5:0:1:1:1,5:0:
HARALICK_1_MEAN_ANGULAR_SECOND_MOMENT:mehrals02:2:0,15:0:0,2:1:
#RULES:1
HARALICK_1_MEAN_ANGULAR_SECOND_MOMENT=mehrals02:>KLASSE=Hintergrund
```

4.7.3. Menüstruktur

Die Menüstruktur des Programms Pollendetektor ist wie folgt gegliedert:

- **Datei** Grundlegende Ein- und Ausgabebefehle
  - **Öffnen** Zeigt einen Dialog zum Öffnen einer Bilddatei an und lädt diese. Unterstützte Formate sind *Bitmap*, *JPEG* und *PNG*. Beim Laden werden die Intensitätswerte des Bildes invertiert.
  - **Testbild laden** Zum schnelleren Zugriff. Lädt ein fest verdrahtetes Testbild.
  - **Ausgabedatei festlegen...** Öffnet einen Dialog, in welchem eine Datei ausgewählt werden kann, in die alle Merkmalswerte exportiert werden.
  - **Bild speichern unter** Erlaubt es, den aktuellen Bearbeitungszustand des geladenen Bildes in eine Datei zu speichern.
  - **Beenden**
- **Operationen** Befehle zur Manipulation des Bildes oder Informationsgewinnung
  - **Reset** Löscht alle Ergebnisse und lädt das ursprüngliche Bild
  - **Gauß-Tiefpass** Filtert das Bild mit einem Gauß-Tiefpassfilter durch Multiplikation im Frequenzbereich. Die Standardabweichung σ kann eingegeben werden.
  - **Invertieren** Invertiert die Intensitätswerte des Bildes.
  - **Median** Wendet einen Medianfilter zu Glättung auf das Bild an. Der Radius des Filters kann eingegeben werden.
  - **Sobel** Wendet den Sobel-Kantenfilter auf das Bild an.
  - **Canny** Wendet den Canny-Kantenfilter auf das Bild an. Die Schwellwerte für die Hysterese sind festgelegt als $t_h = 0,6$ und $t_l = 0,2$.

  - **Faltung mit Kreis** Faltet das Bild mit dem Bild eines weißen Kreises auf schwarzem Hintergrund mit den Abmessungen $r = 55$ und $m = n = 200$.
  - **Kontur zusammenziehen** Testfunktion, welche eine Snake um ein Objekt zusammenzieht. Die Parameter zu Lage, Gewichten und Anzahl der Iterationen können nur im Quelltext geändert werden.
  - **Thresholding** Setzt alle Bildpunkte auf 0, deren Intensität kleiner als ein Schwellwert ist. Der Schwellwert wird nach dem Mixture-Modeling-Verfahren ermittelt.
  - **Segmentiere Pollen 400** Segmentiert eine Mikroskopieaufnahme in 400facher Vergrößerung. Die genaue Beschreibung des Verfahrens findet sich in Kapitel 4.2. Die segmentierten Objekte werden als Bitmaps unter dem Pfad „*Pfad zur Aufnahme*_seg*Nummer des Objekts*.bmp“ abgespeichert. Außerdem werden alle beschreibenden Merkmale berechnet und in der festgelegten Ausgabedatei gespeichert, standardmäßig `C:\statfile.csv`.
  - **Segmentiere Pollen 100** Genau wie der vorherige Punkt, nur für Aufnahmen in 100facher Vergrößerung.
  - **Laws** Berechnet verschiedene Texturmerkmale nach Laws, siehe [Law80b]. Die Resultate werden in Dateien `C:\out-laws0.bmp` bis `C:\out-laws15.bmp` gespeichert.
  - **Haralick** Berechnet die ersten 13 Texturmerkmale nach Haralick, wie in Kapitel 2.2.13 beschrieben, und zeigt das Ergebnis im Statusfenster an.
  - **Sinuswelle** Hilfsfunktion. Erzeugt ein Bild, welches aus der Überlagerung verschiedener Sinuswellen entsteht. Anzahl, Amplitude, Richtung und Phasenverschiebung der Sinuswellen sind im Quelltext festgelegt.

- **Fuzzy** Zugang zu Fuzzy-Mengen und Wissensbasis, sowie Klassifikation.
  - **Wissensbasis** Öffnet das Fenster zur Bearbeitung der Wissensbasis.
  - **Klassifiziere** Klassifizert das in der Liste ausgewählte Objekt mit Hilfe eines Mamdani-Controllers und der geladenen Wissensbasis.

- **Ansicht** Manipulation der Ansicht des Bildes
  - **Zoom +** Zoom in das Bild hinein. Multipliziert den Vergrößerungsfaktor mit $\sqrt{2}$.
  - **Zoom -** Zoom aus dem Bild heraus. Multipliziert den Vergrößerungsfaktor mit $\frac{1}{\sqrt{2}}$.
  - **Reset Zoom** Setzt den Vergrößerungsfaktor auf $1,0$ zurück.

- **Hilfe** Informationen zum Programm Pollendetektor
  - **Hilfe** Zeigt die Programmdokumentation an.
  - **Über Pollendetektor** Zeigt ein Informationsfenster an.

Das Fenster der Fuzzy-Wissensbasis enthält andere Menüpunkte:

- **Datei** Grundlegende Ein- und Ausgabebefehle
  - **Neu** Legt eine neue Wissensbasis an.
  - **Laden** Lädt eine Wissensbasis aus einer *.fkb*-Datei.
  - **Speichern** Speichert die Wissensbasis.
  - **Speichern unter...** Speichert die Wissensbasis in eine neue Datei.
  - **Schliessen** Schließt das Wissenbasis-Fenster .
- **Extras** Zusatzfunktionen
  - **Hole Features** Lädt eine Liste aller verfügbaren Merkmale der segmentierten Objekte und fügt diese als neue linguistische Variablen ein.
  - **Bild LV** Hilfsfunktion. Erzeugt zu der ausgewählten linguistischen Variable eine Bilddatei, die alle zugehörigen Fuzzy-Menge überlagert zeigt.

4.7.4. Dokumentation

Das Programm wurde komplett in *C#* geschrieben und nutzt die *Microsoft .NET*-Laufzeitumgebung. Für die Fourier-Transformation wurden die *FFTW*-Bibliothek [FJ05] und die freie Wrapper-Klasse für *C#* von Tamas Szalay [Sza] genutzt. Für die Matrizenoperationen wurden die *DotNetMatrix*-Klassen [Sel] von Paul Selormey eingebunden welche eine eins-zu-eins-Portierung der *JAMA*-Bibliothek von *The MathWorks, Inc.* sind. Eine umfangreiche Dokumentation aller Klassen und Methoden mit Aufrufdiagrammen findet sich auf der separaten CD. Im nächsten Abschnitt werden nur die wichtigsten Klassen und Methoden näher erläutert.

4.7.5. Klassen

Die wichtigste Klasse im Programm ist die Klasse `ComplexPicture`. Diese repräsentiert ein Graustufenbild mit einer Reihe von Merkmalen und einer Kontur. Um Operationen im Frequenzbereich zu erleichtern werden die Intensitätswerte als komplexe Zahlen gespeichert. Der Imaginärteil kann auch als Zwischenspeicher genutzt werden.

ComplexPicture

Die Klasse `ComplexPicture` wird sowohl zu Repräsentation einer Aufnahme als auch der einzelnen segmentierten Objekte verwendet. Normalerweise ist der Wertebereich des Real- und Imaginärteils $[0, 1]$, andere Werte sind aber auch möglich. Jedoch muss dann vor einer Ausgabe des Bildes normalisiert werden. Die Attribute und ihre Funktion lauten:

- `double [,] m_Re` Ein zweidimensionales Feld der Abmessung `m_Width` $\times$ `m_Height`, welches den Realteil der Intensitätswerte speichert.

- `double [,] m_Im` Ein zweidimensionales Feld der Abmessung `m_Width` × `m_Height`, welches den Imaginärteil der Intensitätswerte speichert.
- `int m_Width` Die Breite des Bildes
- `int m_Height` Die Höhe des Bildes
- `Kontur m_Kontur` Die Kontur des segmentierten Objekts.
- `SortedList<string,double> m_Features` Eine Liste der beschreibenden Merkmale. Jedes Merkmal ist über seinen Namen zugänglich.

Da die Attribute alle als geschützt (`protected`) deklariert sind, existieren öffentliche *Properties* als *Getter*-Methoden.

Um die Klasse *ComplexPicture* zu initialisieren oder auszugeben, stehen diverse Konstruktoren und Methoden zum Laden und Konvertieren bereit. Diese sind:

- `public ComplexPicture()` Erzeugt ein neues 1×1 `ComplexPicture` und setzt alle Attribute auf Standardwerte.
- `public ComplexPicture(double[,] real)` Erzeugt ein neues `ComplexPicture` mit Standardwerten und lädt das übergebene Bild in den Realteil.
- `public ComplexPicture(double[][] real)` Erzeugt ein neues `ComplexPicture` mit Standardwerten und lädt das übergebene Bild in den Realteil. Alle Zeilen des Parameters sollten gleich lang sein.
- `public ComplexPicture(double[,] real, double[,] imaginary)` Erzeugt ein neues `ComplexPicture` mit Standardwerten und lädt die übergebenen Bilder in den Real- und Imaginärteil.
- `public ComplexPicture(double[][] real, double[][] imaginary)` Erzeugt ein neues `ComplexPicture` mit Standardwerten und lädt die übergebenen Bilder in den Real- und Imaginärteil. Alle Zeilen der Parameter sollten gleich lang sein.
- `public ComplexPicture(ComplexPicture real, ComplexPicture imaginary)` Erzeugt ein neues `ComplexPicture` mit Standardwerten und lädt die übergebenen Bilder in den Real- und Imaginärteil, aber nur, wenn beide Bilder gleich groß sind.
- `public ComplexPicture(string file, bool invert)` Erzeugt ein neues `ComplexPicture` und lädt das Bild aus einer Datei in den Realteil. Der zweite Parameter gibt an, ob das Bild beim Laden invertiert wird.
- `public ComplexPicture(string file)` Gleiches Verhalten wie `ComplexPicture(file, true)`.
- `public ComplexPicture(int w, int h)` Erzeugt ein schwarzes (Intensität = 0) Bild angegebener Breite und Höhe.

- `public ComplexPicture(int w, int h, int radius, double fg, double bg)` Erzeugt ein Bild mit einem Kreis in der Mitte des Real-Anteils.
- `public ComplexPicture(Bitmap bitmap)` Lädt ein Bild aus dem übergebenen `Bitmap`-Objekt.
- `public void LoadImage(string file, bool invert)` Lädt den Inhalt der angegebenen Bilddatei in das `ComplexPicture`. Der Einsatz eines *Look-up-Tables* wie bei den unten stehenden `BitmapFrom...`-Methoden brachte hier keinerlei Geschwindigkeitszuwachs.
- `public static ComplexPicture Gauss(int w, double sigma)` Erzeugt ein quadratisches Bild der Kantenlänge `w` mit einer von der Mitte ausgehenden Gaußverteilung.
- `public Bitmap BitmapFromRe()` Gibt den Realteil des Bildes als `Bitmap`-Objekt zurück. Zur Beschleunigung wurde bei allen `BitmapFrom...`-Methoden ein *Look-up-Table* für die Umrechnung von 2D-`double`-Array zu `Bitmap`-Objekt verwendet.
- `public Bitmap BitmapFromIm()` Gibt den Imaginärteil des Bildes als `Bitmap`-Objekt zurück.
- `public Bitmap BitmapFromReIm()` Gibt den Realteil des Bildes als `Bitmap`-Objekt zurück und markiert alle Punkte rot, deren Imaginärteil $>= 1{,}0$ ist.
- `public int[,] ToIntArray2D()` Gibt den Realteil des Bildes als zweidimensionales Integer Array Feld zurück. Vorher wird mit 255 skaliert.

Zum Bearbeiten des Bildes existieren zahlreiche Methoden. Generell erzeugt jede Funktion dabei ein neues `ComplexPicture`-Objekt, welches das Ergebnis beinhaltet. Es ist aber genauso möglich *in-place* zu arbeiten, das heißt die Daten des Objekts direkt zu manipulieren ohne ein neues Objekt zu erzeugen. Diese Methoden sind durch das Präfix `Do` kenntlich gemacht. Zum Beispiel

```
public ComplexPicture CannyEdge(double sigma, double t_h, double t_l)
```

und

```
public void DoCannyEdge(double sigma, double t_h, double t_l)
```

Hier seien jetzt nur die Methoden aufgelistet, die nicht *in place* arbeiten:

- `public ComplexPicture FFT()` Führt eine schnelle diskrete Fourier-Transformation durch.
- `public ComplexPicture FFTInv()` Führt eine inverse schnelle diskrete Fourier-Transformation durch.
- `public ComplexPicture Multiply(ComplexPicture b)` Multipliziert pixelweise zwei Bilder miteinander.

- `public ComplexPicture ScaledBy(double factor)` Skaliert alle Werte des Realteils mit `b`.
- `public ComplexPicture FaltungFFT(ComplexPicture womit)` Faltet zwei Bilder miteinander, indem zunächst beide in den Frequenzbereich transformiert und dort multipliziert werden und das Ergebnis zurücktransformiert wird.
- `public ComplexPicture KorrelationFFT(ComplexPicture womit)` Korreliert zwei Bilder miteinander, indem zunächst beide in den Frequenzbereich transformiert und dort konjugiert und dann multipliziert werden und das Ergebnis zurücktransformiert wird.
- `public ComplexPicture FaltungRe(ComplexPicture filter)` Faltet das Bild mit dem `filter`-Objekt im Ortsbereich.
- `public ComplexPicture Resize(int newWidth, int newHeight)` Gibt das Bild in neuer Größe zurück, wobei nicht skaliert sondern abgeschnitten oder mit Nullen aufgefüllt wird.
- `public ComplexPicture Normalized()` Begrenzt den Wertebereich von Real- und Imaginärteil auf das Intervall $[0, 1]$. Größere Werte werden auf 1, kleinere auf 0 gesetzt.
- `public ComplexPicture Normalized2()` Begrenzt den Wertebereich von Real- und Imaginärteil auf das Intervall [0,1]. Zuerst werden der kleinste und der größte vorkommende Werte ermittelt. Danach werden alle Werte durch die Differenz zwischen diesen beiden geteilt und das Minimum abgezogen.
- `public ComplexPicture Conjugate()` Konjugiert das komplexwertige Bild. Alle Werte des Imaginärteils werden mit -1 multipliziert.
- `public ComplexPicture AddCP(ComplexPicture what)` Addiert die Intensitätswerte zweier Bilder.
- `public ComplexPicture CannyEdge(double sigma, double t_h, double t_l)` Führt den Kantenfilter nach Canny aus. Genaue Beschreibung siehe Kapitel 2.2.10.
- `public int[,] FindeMaxima(double tol)` Sucht alle regionalen Maxima. Genaue Beschreibung siehe Kapitel 2.2.8.
- `public ComplexPicture Exp()` Berechnet für jeden Pixel im Realteil $exp(Pixel)$.
- `public ComplexPicture Log()` Berechnet für jeden Pixel im Realteil $log(Pixel)$.
- `public ComplexPicture Shift(int dx, int dy)` Verschiebt das Bild periodisch um dx Pixel nach rechts und dy Pixel nach unten.
- `public ComplexPicture Ausschnitt(int mx, int my, int radius)` Liefert einen Ausschnitt des Bildes der Größe $2radius \times 2radius$.

- `public ComplexPicture AusschnittPadded(int x, int y, int w, int h)` Liefert einen Ausschnitt des Bildes der Größe $w \times h$, dessen linke obere Ecke am Punkt (x, y) liegt. Sollten Parameter außerhalb des Bildes liegen, wird mit Nullen aufgefüllt.

- `public ComplexPicture Median(double radius)` Wendet den Medianfilter auf das Bild an. Siehe auch Kapitel 2.2.11.

- `public ComplexPicture Sobel()` Berechnet ein einfaches Gradientbild des Bildes, indem mit dem senkrechten und dem waagerechten Sobel-Operator gefaltet und die Länge der resultierenden „Vektoren“ berechnet wird.

- `public int[] Histogramm()` Berechnet das Histogramm des Realteils des Bildes bei 256 Quantisierungsstufen.

- `public List<int> HistoMax()` Liefert eine Liste der lokalen Maxima des oben ermittelten Histogramms.

- `public List<int> HistoMax(int grad)` Liefert eine Liste der regionalen Maxima des oben ermittelten Histogramms, die mindestens `grad` höher sind als ihre Nachbarn.

- `public int MixtureModeling()` Liefert den Schwellwert nach dem Mixture-Modeling-Verfahren. Siehe auch Kapitel 2.2.9.

- `public ComplexPicture Threshold(double t)` Wendet einen Schwellwert auf das Bild an. Alle Werte, die kleiner sind, werden auf 0 gesetzt.

- `public ComplexPicture InversRe()` Invertiert den Realteil des Bildes nach der Formel $m\_Re[x, y] = 1,0 - m\_Re[x, y]$.

- `public bool ImBild(int x, int y)` Gibt an, ob die genannten Koordinaten im Bild liegen oder außerhalb.

Zum Berechnen der beschreibenden Merkmale stehen noch die Methoden

- `public double[, ,] FilterHaralickMasked(int dist_max, int grayleves)` Berechnet die Texturmerkmale nach Haralick für den Bereich des Bildes, der innerhalb der Kontur liegt.

- `public List<ComplexPicture> Filter_Laws_texture()` Berechnet die Texturmerkmale nach Laws [Law80b].

- `public double GetMeanRe()` Berechnet den Mittelwert des Realteils.

- `public double[] FeaturesKontur()` Berechnet Merkmale bezüglich der Kontur (Umfang, Fläche, Fläche/Umfang,Umfang/Durchmesser und mittlerer Krümmungswinkel)

- `public void ComputeFeatures()` Stößt die Berechnung aller Merkmale an.

zur Verfügung. Diese werden allerdings automatisch beim Zugriff auf die Eigenschaft `Features` aufgerufen, mit Ausnahmen von `Filter_Laws_texture()`.

Kontur

Die Klasse `Kontur` entspricht zunächst einer Liste von Punkten. Sie erbt von der Klasse `List<Points>` und wurde lediglich um einige Methoden und Attribute erweitert, die für das Snake-Verfahren aus Kapitel 2.2.12 benötigt werden. Im Speziellen sind dies

- `protected double[,] m_Gradient` Verweis auf das Gradientenbild für die bildbezogene Energie der Snake.
- `protected int m_Width` Breite des Gradientenbildes
- `protected int m_Height` Höhe des Gradientenbildes
- `protected double m_CurvatureWeight` Gewichtung der Krümmungsenergie
- `protected double m_ContinuityWeight` Gewichtung der Stetigkeitsenergie
- `protected double m_GradientWeight` Gewichtung der Bildenergie
- `protected int m_CurveBehavior` Gibt an, ob sich die Snake zusammenziehen oder ausdehnen soll.

Diese sind alle über Properties zugänglich.

Es wurden zwei verschiedene Snake-Verfahren implementiert. Ein Verfahren nach Mark Schulze [Sch] betrachtet für jeden Punkt der Kontur mögliche alternative Positionen in einer definierten Umgebung und ermittelt die Position mit der geringsten Energie. Das andere Verfahren wurde strikt nach dem Artikel von Kass, Witkin und Terzopoulus [KWT88] implementiert und arbeitet mit Matrizenmultiplikation und dem Lösen linearer Gleichungssysteme.

Es werden einige private Variablen als Zwischenspeicher benutzt:

- `private GeneralMatrix ms_AgammaIinv` Die Matrix $A + \gamma I$
- `private GeneralMatrix ms_Px` Horizontale Bildenergie P_x.
- `private GeneralMatrix ms_Py` Vertikale Bildenergie P_Y.
- `private GeneralMatrix ms_x_t` Neue berechnete x-Koordinaten x_t.
- `private GeneralMatrix ms_x_t_1` Alte x-Koordinaten der vorherigen Iteration x_{t-1}.
- `private GeneralMatrix ms_y_t` Neue berechnete y-Koordinaten x_t

- `private GeneralMatrix ms_y_t_1` Alte y-Koordinaten der vorherigen Iteration y_{t-1}.
- `private double ms_gamma` Zwischenspeicher für die Schrittweite gamma

Zuerst muss die Snake über die Methode `public void InitSnake(double gamma)` initialisiert werden. Diese berechnet die Vorbelegung der Matrix $A + \gamma I$ auf Basis der vorher festgelegten Gewichte für Stetigkeit und Steifigkeit. Mit der Methode *public int SnakeStep()* wird dann jeweils eine Iteration der Snake durchgeführt. Sollten sich Punkte beim Zusammenziehen überlappen, werden Duplikate automatisch entfernt.

MaxPoint

Die Klasse `MaxPoint` ist ein zweidimensionaler Punkt mit ganzzahligen Koordinaten, der eine Fließkommazahl als Intensität besitzt. Entsprechend sind die Attribute

- `public double value` Der (Intensitäts-) Wert des Punktes
- `public int x` Die x-Koordinate des Punktes
- `public int y` Die y-Koordinate des Punktes

Als Methoden besitzt `MaxPoint` nur einen Konstruktor und den Vergleichsoperator `CompareTo(object o)`, so dass im Quelltext zwei `MaxPoint`-Objekt mit `<` verglichen werden können.

FuzzySet

Die Klasse `FuzzySet` repräsentiert eine Fuzzy-Menge und stellt einige Operationen auf dieser zur Verfügung. Die Zugehörigkeitsfunktion der Fuzzy-Menge ist durch Punkte gegeben, zwischen denen linear interpoliert wird. Die drei Attribute lauten

- `public String name` Der Name der Fuzzy-Menge.
- `public String lingVar` Der Name der linguistischen Variable, zu der diese Fuzzy-Menge gehört.
- `public SortedList<double, double> points` Eine Liste von Punkten, welche die Zugehörigkeitsfunktion der Fuzzy-Menge festlegen.

Neben Konstruktoren und Funktionen zum Laden und Speichern existieren noch drei weitere Methoden, welche für die Verwendung in einem Mamdani-Controller nötig sind.

- `public double Evaluate(double x)` Gibt die Zugehörigtkeit des Parameters x zum FuzzySet zurück.
- `public void MinChop(double y)` Schneidet die Zugehörigkeitsfunktion auf der angebenen Höhe ab.

- `public FuzzySet MaxComposition(FuzzySet fs)` Berechnet die t-Norm Maximun nach Zadeh von dieser Fuzzy-Menge mit der angegebenen.
- `public double? GetMaximumPosition()` Ermittelt die Position des ersten Maximums oder bei einem Trapez den Mittelpunkt dieses als Maximum.
- `public double CenterOfGravityX()` Ermittelt die x-Koordinate des Schwerpunkts der Zugehörigkeitsfunktion.

FuzzyRule

Die Klasse `FuzzyRule` repräsentiert eine Fuzzy-Regel bestehend aus Prämisse und Konsequenz. Die Regel hat die Form `WENN Prämisse1 UND Prämisse2 UND ... UND PrämisseN DANN Konsequenz`. Die Prämisse wird als reine Konjunktion betrachtet. Entsprechend hat die Klasse auch nur zwei Attribute

- `public List<FuzzySet> premises` Die Liste von Fuzzy-Mengen der Prämisse der Regel
- `public FuzzySet consequence` Die Konsequenz der Regel

Die wichtigste Methode ist `public FuzzySet Evaluate(SortedList<string, double> values)`. Diese wertet die Regel für die angegeben Parameter aus. Zuerst werden die Zugehörigkeitsgrade der Fuzzy-Mengen der Prämisse ermittelt. Diese werden mit der t-Norm $Min()$ verknüpft (Aggregation), das heißt der kleinste Wert ausgewählt und die Konsequenz an diesem Wert „abgeschnitten“ (Implikation). Siehe auch Kapitel 2.3.6.

Mit der Funktion `public string GetSignature()` kann man überprüfen, ob die Regel mit den zur Verfügung stehenden Belegungen für linguistische Variablen ausgewertet werden kann. Die Signatur hat die Form

```
LINGVARp1,LINGVARp2,LINGVARp3,->LINGVARcon
```

MamdaniController

Der Mamdani-Controller dient zum einen als Container für alle Fuzzy-Mengen und Fuzzy-Regeln und zum anderen zum Auswerten der Regeln. Die beiden Attribute hierfür heißen

- `private List<FuzzyRule> m_Rules` Die Liste aller Fuzzy-Regeln
- `private List<FuzzySet> m_Sets` Die Liste aller Fuzzy-Mengen

und sind als privat deklariert, jedoch über die Eigenschaften `Rules` und `Sets` der Klasse `MamdaniController` lesend erreichbar.

Es stehen diverse Methoden für den Zugriff auf sowie das Initialisieren, Speichern und Laden von Regeln und Mengen zur Verfügung. Für die Klassifikationsaufgabe ist jedoch die Methode `public string ClassifyMax(SortedList<string, double> values, string whichLingVar)` interessant. Dieser wird eine per Name indizierte Liste von Werte übergeben sowie der Name der linguistischen Variablen, auf die klassifizert werden soll. Es

werden alle auf die Werte passenden Regeln ausgewertet und zum Schluss der Name der Fuzzy-Menge ausgegeben, in welcher der defuzzifizierte Wert liegt.

Möchte man einen scharfen Wert mit dem Mamdani-Controller ermitteln, kann man die Methode `public double AnswerCoG(SortedList<string, double> values, string whichLingVar)` benutzen.

GUI

Zur grafischen Oberfläche sei nur gesagt, dass für jedes Fenster je eine Klasse existiert, `frmMain` und `frmFuzzy`. Für die grafische Darstellung der Fuzzy-Mengen wurde das *Control* `FuzzyPointEditor` entworfen, welches das Anzeigen und Manipulieren eines `FuzzySet`-Objekts ermöglicht. Für detailliertere Informationen wird auf die Klassendokumentation auf der CD verwiesen.

5. Erweiterung auf andere Objekte

Die Einsetzbarkeit des Verfahrens zur Detektion und Klassifikation von Pollenkörnern wurde gezeigt. Interessant ist aber auch, ob sich das Verfahren direkt oder mit geringen Modifikationen für andere Belange nutzen lässt. Ein mögliches Einsatzgebiet stellt hier die Nanotechnologie dar.

Dank Unterstüzung durch *Dr. Alex von Bolen* vom *Institute for Analytical Sciences* in Dortmund standen hierfür Aufnahmen eines Rasterelektronenmikroskops zur Verfügung.

5.1. Grundlagen der Nanotechnologie

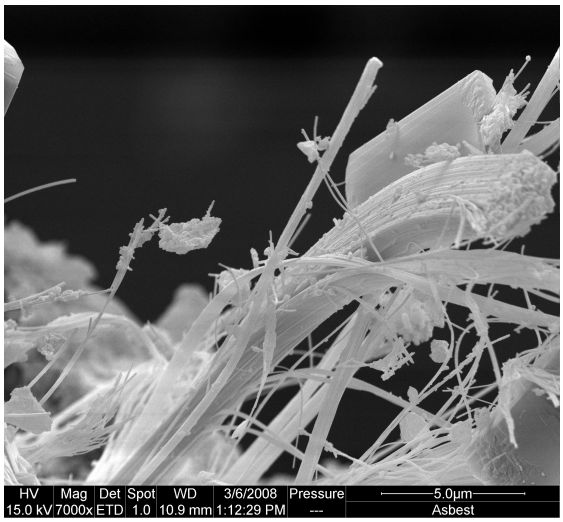


Abbildung 5.1.: Mikroskopieaufnahme von Asbest in 500facher Vergrößerung.

Die Nanotechnologie ist grob gesagt die *Wissenschaft von den sehr kleinen Teilchen.* Sehr klein bedeutet, dass Partikel betrachtet, analysiert, verwendet und modifiziert werden, welche einen Durchmesser von wenigen Mikrometer (νm) bis hinunter zu einigen Nanometern (nm) haben.

Anwendungsgebiete der Nanotechnologie sind reichlich vorhanden. Hier seien nur einige Beispiele genannt:

- Produktion von Medikamenten, welche so fein sind, dass sie von der (Schleim-) Haut absorbiert werden können

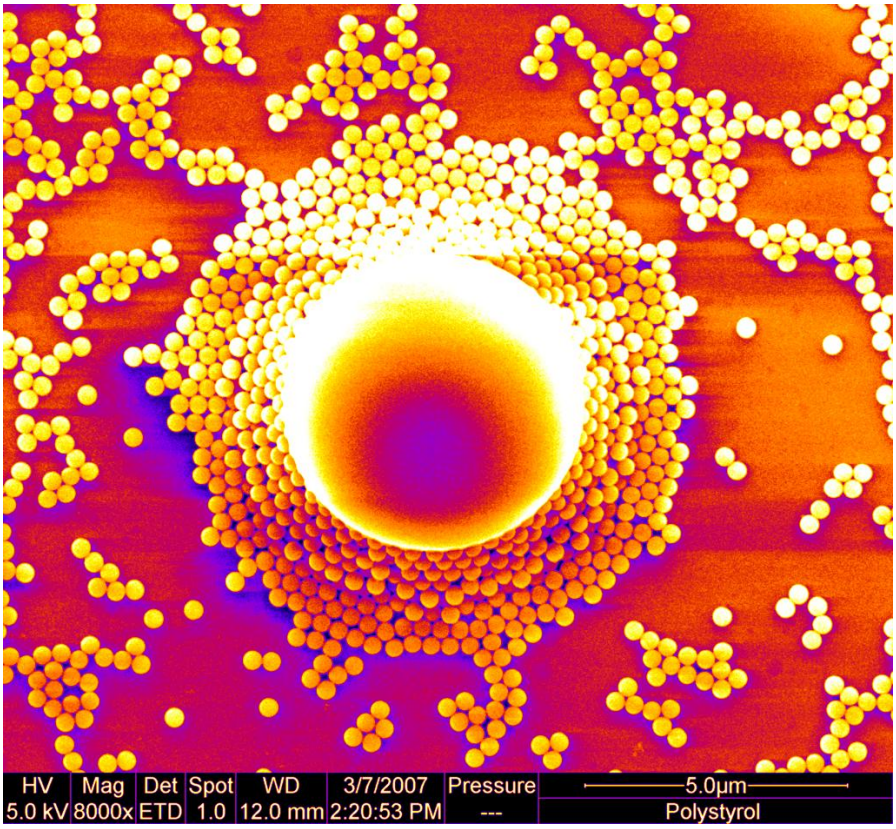


Abbildung 5.2.: Polystyrolkugeln unterschiedlicher Größe in Falschfarbendarstellung

- Analyse der Zusammensetzung bestimmter Stoffe, zum Beispiel zur Echtheitsprüfung eines Gemäldes oder historischer Instrumente
- Lab-On-A-Chip-Systeme zur chemischen Diagnostik / Analyse von Stoffen auf kleinstem Raum
- Farben und Beschichtungen mit Lotos-Effekt

Um diese kleinsten Partikel, die mit dem menschlichen Auge natürlich nicht ohne Weiteres sichtbar sind, handhaben zu können, bedarf einer Apparatur zum Vergrößern dieser Partikel. Da eine rein optische Vergrößerung hier nicht mehr ausreicht, wird dazu meist ein Rasterelektronenmikroskop, kurz REM, verwendet, dessen Funktionsweise im nächsten Abschnitt erläutert wird.

5.1.1. Funktionsweise eines Rasterelektronenmikroskops

Das Wort Raster im Namen deutet schon darauf hin, dass beim REM das Objekt zeilenweise Punkt für Punkt abgetastet wird. Ein guter Vergleich hierzu ist der Bildaufbau beim Röhrenfernseher. Einen einzelnen Bildpunkt erhält man nun, indem der Punkt

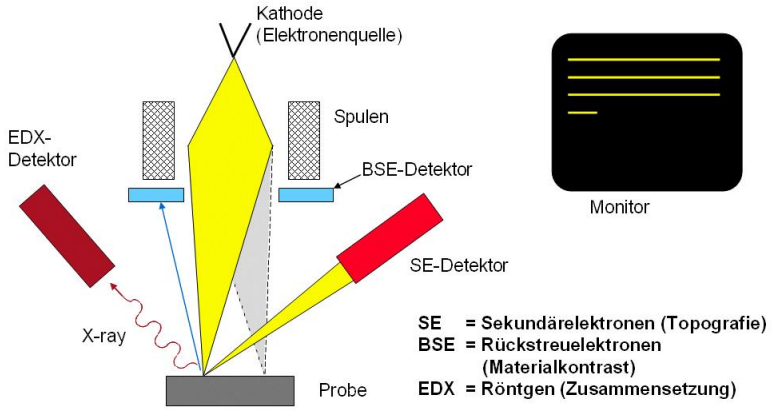


Abbildung 5.3.: Der von der Kathode erzeugte Elektronenstrahl wird durch eine Spule gebündelt und beschleunigt und trifft auf das Material. Dieses emittiert Sekundärelektronen und Röntgenstrahlen, welche je von einem Sensor erfasst werden. Quelle: de.wikipedia.org.

auf dem Objekt mit einem sehr feinen Elektronenstrahl beschossen wird und ein Sensor registriert, wie stark Sekundärelektronen und Röntgenstrahlen vom bestrahlten Punkt emittiert werden.

5.2. Erweiterung auf Nanotechnologie

5.2.1. Problemstellung

Als mögliche Aufgabenstellung kam die Analyse von künstlich erzeugten Polystyrolkugeln hinzu. Für die Praxis interessant sind hier einige statistische Maße bezüglich Größe und Form und auch Lage der einzelnen Kugeln. Dies sind unter anderem

1. Durchmesser beziehungsweise Größe der Kugeln
2. Anzahl der Kugeln
3. Normabweichung der Größe
4. Unrundheit
5. Anzahl und Vorhandensein von Kon- / Agglomeraten

Für die Punkte 1. und 3. stehen bereits Methoden zur Verfügung. Der 2. Punkt lässt sich durch einfaches Abzählen lösen und auch der 4. Punkt lässt sich nach einer Segmentierung aller Kugeln leicht berechnen. Für den 5. Punkt, das Auffinden von Konglomeraten, ist die Implementierung eines einfachen Clustering-Verfahrens hilfreich.

5.2.2. Vorgehen

Da es sich bei den Polystyrolkugeln um runde Objekte handelt, kann das in Kapitel 4.2 ausführlich beschriebene Verfahren zur Segmentierung eingesetzt werden. Einzig die Parameter müssen ein wenig an die Größe der Kugeln angepasst werden. Gute Ergebnisse wurden erzielt mit den Werten

- Radius des Kreises mit dem gefaltet wird = 25
- Toleranz der Maximasuche = 0
- Radius der initalen Snake = 30
- Gewichte der Snake:
  - Stetigkeitsenergie = 1,5
  - Krümmungsenergie = 1,0
  - Bildenergie = 1,1

Das Ergebnis einer Segmenitierung nach diesem Verfahren kann man in Abbildung 5.4 sehen.

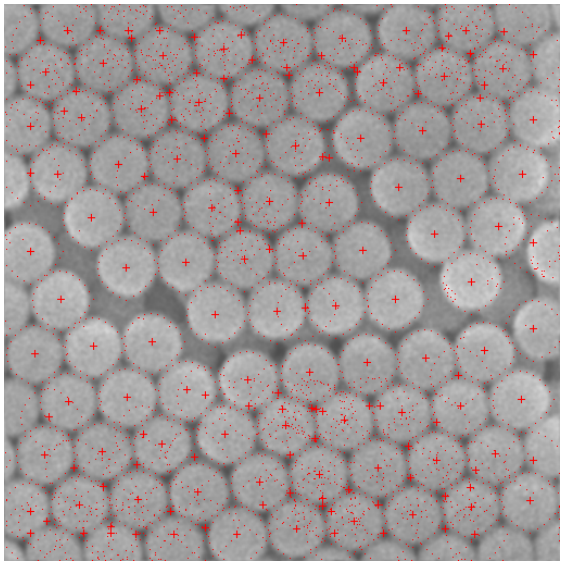


Abbildung 5.4.: Segmentierungsergebnis für den Ausschnitt einer Mikroskopieaufnahme von Polystyrolkugeln

Da aber auch hier Artefakte segmentiert werden, macht es Sinn vor einer Berechnung der statistischen Maße zwischen Kugeln und Artefakten zu unterscheiden. Hierfür ist das in Kapitel 2.3.6 beschriebene Fuzzy-Logik-System geeignet. Zur Unterscheidung zwischen Artefakt und Polystyrolkugeln werden die beiden linguistischen Variablen HARALICK_4_MEAN_SUM_OF_SQUARES und HARALICK_13_RANGE_-

INFMEAS_CORR_COEFF_2 benutz mit den in Abbildung 5.5 und 5.6 erkennbaren Fuzzy-Mengen. Die Fuzzy-Menge der Klasse ist in Abbildung 5.7 zu sehen.

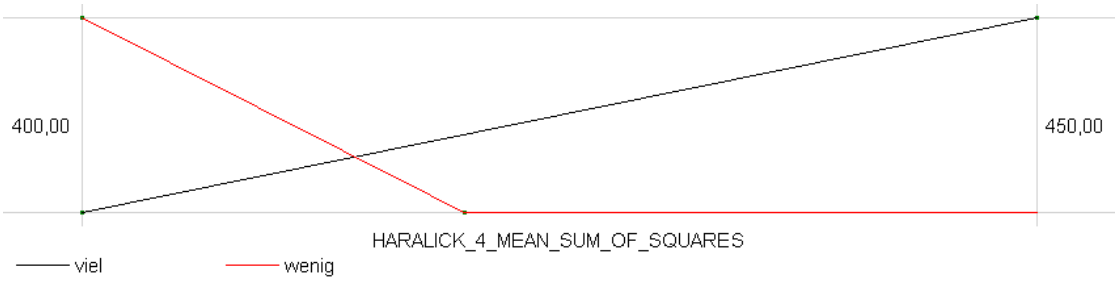


Abbildung 5.5.: Die Fuzzy-Mengen der linguistischen Variable HARALICK_4_MEAN_SUM_OF_SQUARES

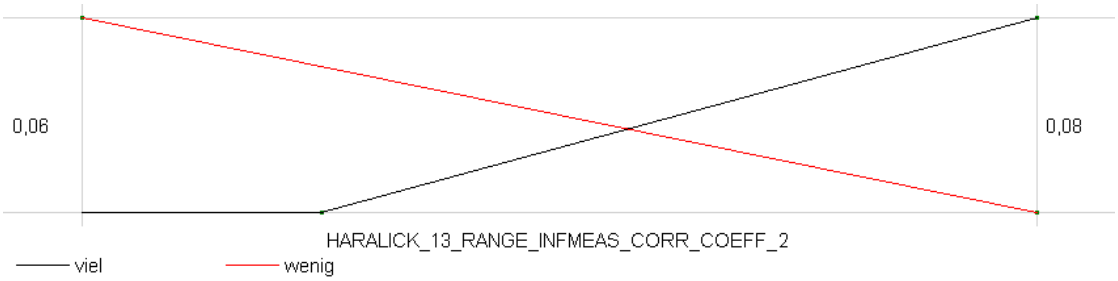


Abbildung 5.6.: Die Fuzzy-Mengen der linguistischen Variable HARALICK_13_RANGE_INFMEAS_CORR_COEFF_2

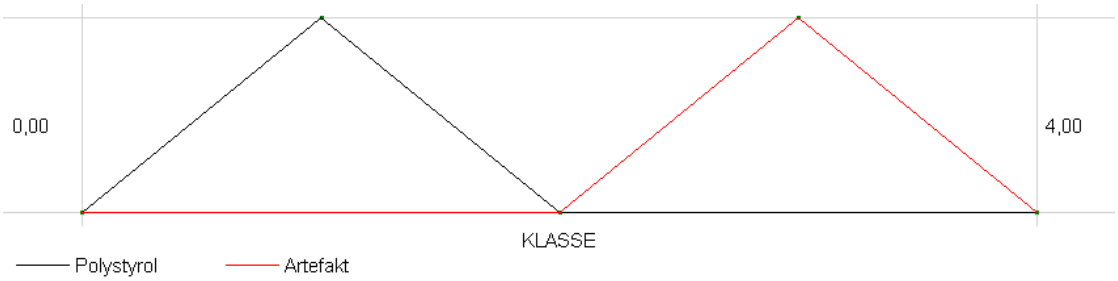


Abbildung 5.7.: Die Fuzzy-Mengen der linguistischen Variable KLASSE

Die entsprechenden Regeln zur Klassifikation der Kandidaten als Artefakte oder Polystyrolkugeln lauten:

- HARALICK_13_RANGE_INFMEAS_CORR_COEFF_2 = wenig
  HARALICK_4_MEAN_SUM_OF_SQUARES = viel
  => KLASSE = Polystyrol

- HARALICK_13_RANGE_INFMEAS_CORR_COEFF_2 = viel
  => KLASSE = Polystyrol

- HARALICK_13_RANGE_INFMEAS_CORR_COEFF_2 = wenig
  HARALICK_4_MEAN_SUM_OF_SQUARES = wenig
  => KLASSE = Artefakt

Da die Aufnahmen alle von stark schwankender Qualität sind, lassen sich mit Sicherheit noch robustere Regeln als die oben genannten finden. Nachdem die Artefakte mit Hilfe des Fuzzy-Logik-System aussortiert wurden, können weitere Verfahren folgen, welche die gewünschten statistischen Maße oder Agglomerationen ermitteln. Dem Programm *Pollendetektor* wurde zur Demonstration der Menüpunkt *Operationen -> Segmentiere Polystyrol* hinzugefügt.

6. Zusammenfassung und Ausblick

In dieser Arbeit wurde ein Verfahren entwickelt, mit dem es möglich ist, Pollen aus Mikroskopieaufnahmen verschiedener Auflösungen zu segmentieren und zu klassifizieren.

Zunächst wurden mögliche Objekte aus den Aufnahmen segmentiert. Bei der Segmentierung wurde auf bekannte Methoden zurückgegriffen und diese zu einem neuen Verfahren kombiniert. Für die so segmentierten Objekte wurden beschreibende Merkmale berechnet, welche zum einen von der Form und zum anderen von der Textur des Objekts abhängen. Um diese Objekte an Hand ihrer Merkmale zu klassifizieren wurde ein Fuzzy-Logik-System verwendet. Bevor dieses benutzt werden konnte, musste manuell eine Wissenbasis aufgebaut werden. Danach konnten mit dem Fuzzy-Logik-System die verschiedenen Arten von Objekten beziehungsweise Pollen erfolgreich unterschieden werden.

Bei der Güte der Klassifikation durch das Fuzzy-Logik-System ist man auf die Exaktheit der Wissensbasis angewiesen. An dieser Stelle ließe sich durch Überarbeiten der Fuzzy-Mengen und Fuzzy-Regeln sicherlich noch eine bessere Klassifikation erzielen. Vorteile des Fuzzy-Logik-Systems sind klar die einfache Implementierung sowie die intuitive Benutzung durch fast natürlichsprachliche Beschreibung.

Dass das entwickelte Segmentierungsverfahren auch in anderen Bereichen Anwendungen finden kann, zeigte sich in Kapitel 5.2. Zwar kann es abhängig von Art und Qualität der Eingabedaten erforderlich sein, Vorverarbeitungsschritte einzufügen, doch das Grundprinzip ist weiter einsetzbar. Vorstellbar ist auch eine Erweiterung des Segmentierungsverfahrens auf stark ovale beziehungsweise elliptische Objekte. Allerdings müsste noch ein Arbeitschritt eingefügt werden, der die Orientierung der Objekte bestimmt, damit die Ausgangskontur für die *Snake* besser an das Objekt angepasst werden kann.

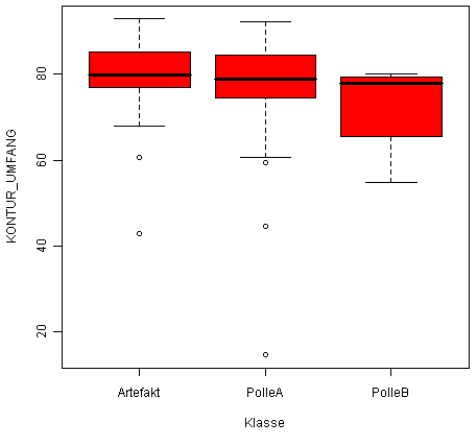
Das Expertensystem wäre noch dahingehend erweiterbar, dass zusätzlich zum üblichen `und` auch noch die Operationen `nicht` und `oder` unterstützt werden. Dies ist zwar bereits implizit dadurch möglich, entsprechende Fuzzy-Mengen und -Regeln manuell zu erzeugen, doch würde der Bedienkomfort hierdurch steigen.

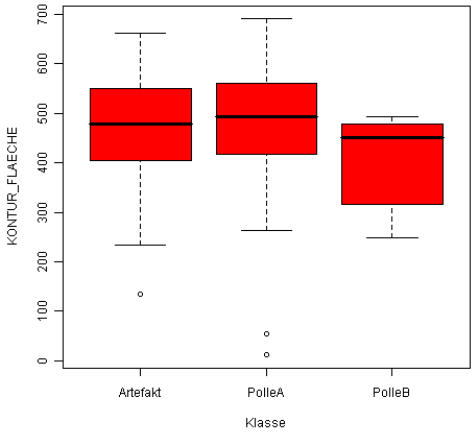
Interessant wäre es, das beschriebene Verfahren noch für andere Pollenarten als die in den Testdaten auftretenden Birken- und Olivenpollen zu testen. Leider standen von Hasel, Erle, Roggen, Beifuß oder Gräsern keine Bilder zur Verfügung, obwohl diese aus allergologischer Sicht auch von Bedeutung sind.

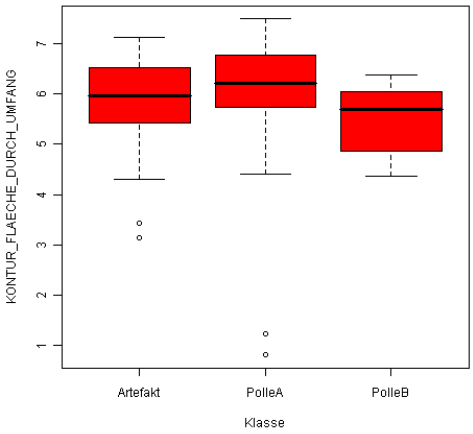
A. Statistiken

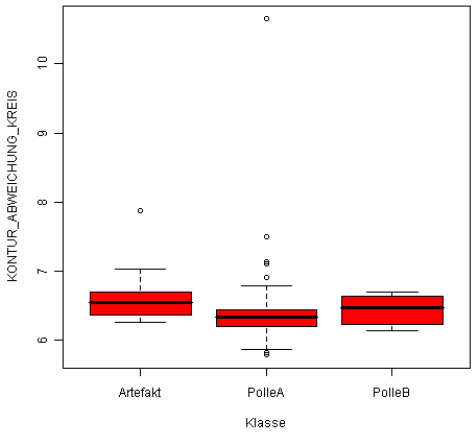
An den Grafiken in diesem Abschnitt kann man sehr gut erkennen, wie sich die Merkmalswerte auf die verschiedenen Klassen verteilen. Bei den Aufnahmen in 100facher Vergrößerung wird nur zwischen den Klassen *Artefakt*, *PolleA* und *PolleB* unterschieden. Für die Aufnahmen in 400facher Vergrößerung existieren die sechs Klassen *Hintergrund*, *Artefakt*, *Birke*, *Birke_kaputt*, *Olive* und *Olive_kaputt*. Bezüglich der Texturmerkmale konnten je die Klassen *Birke* und *Birke_kaputt* sowie *Olive* und *Olive_kaputt* zusammengefasst werden.

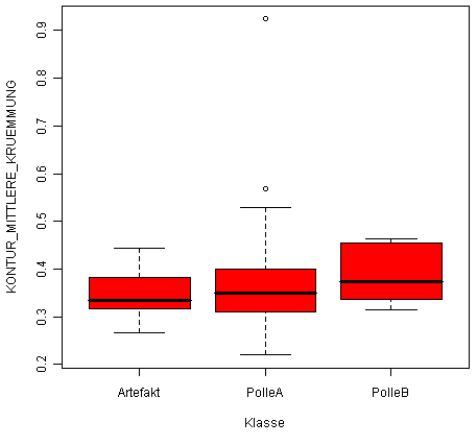
A.1. Aufnahmen in 100facher Vergrößerung

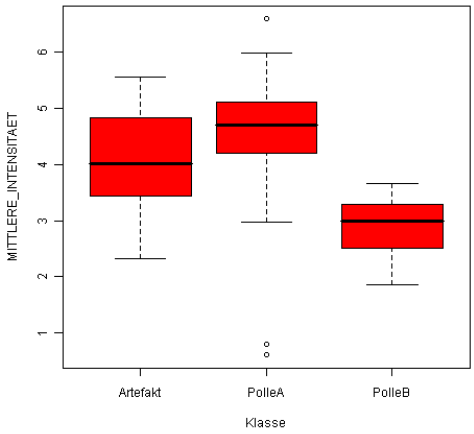


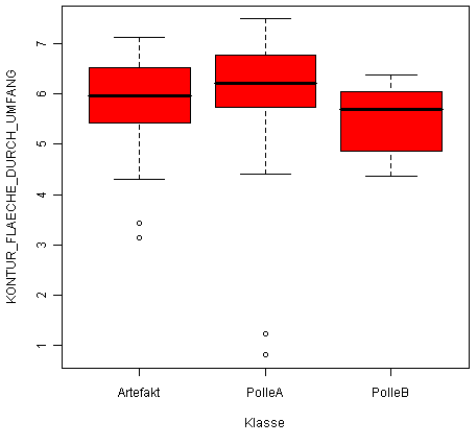


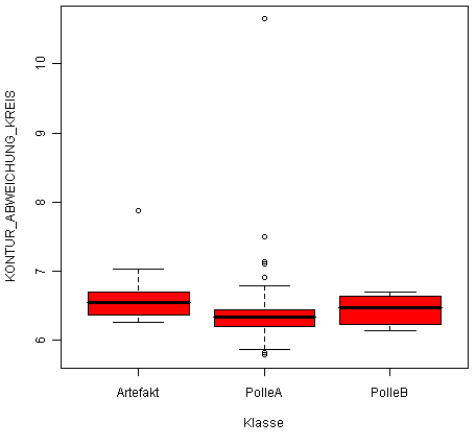


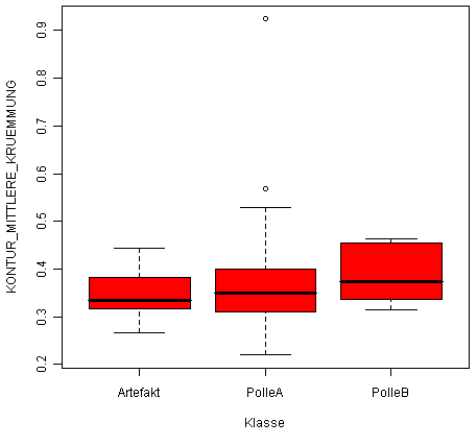


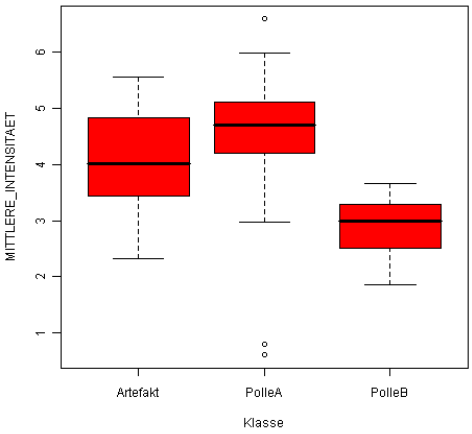


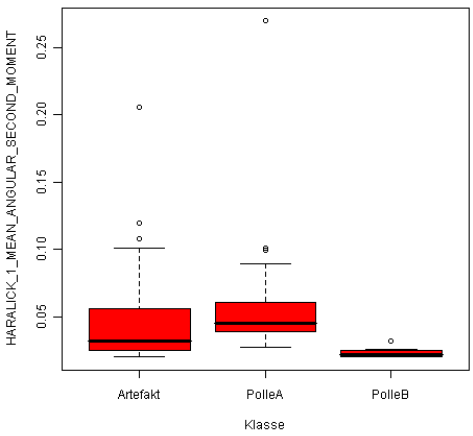


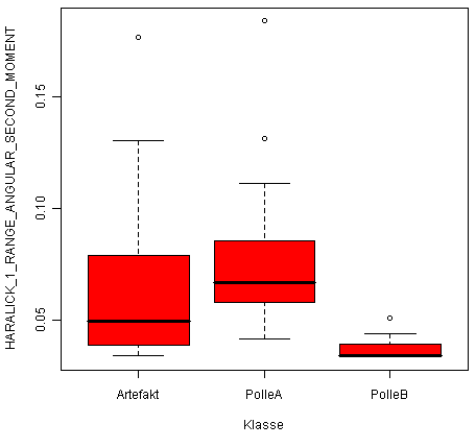

KONTUR_FLAECHE_DURCH_UMFANG
1
2
3
4
5
6
7
Artefakt
PolleA
PolleB
Klasse

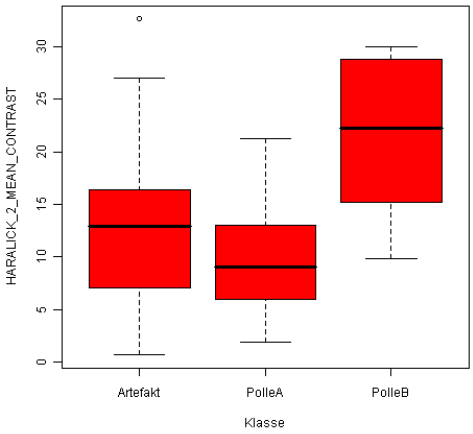

KONTUR_ABWEICHUNG_KREIS
6
7
8
9
10
Artefakt
PolleA
PolleB
Klasse

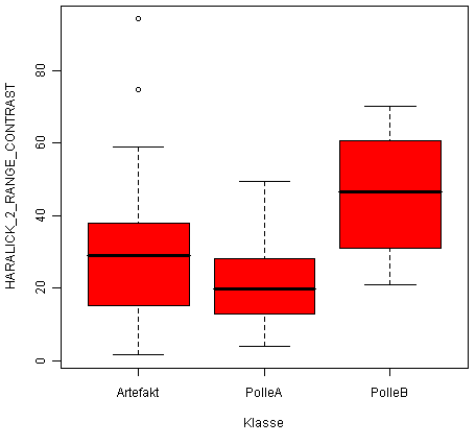

KONTUR_MITTLERE_KRUEMMUNG
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
Artefakt
PolleA
PolleB
Klasse

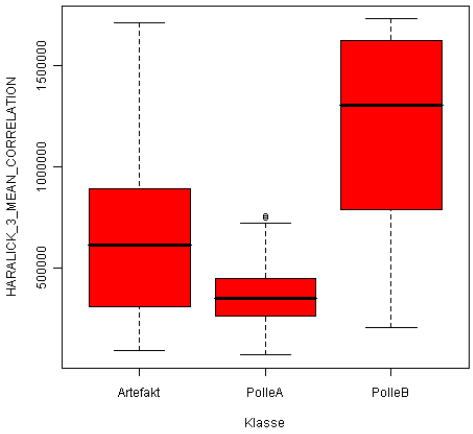

MITTLERE_INTENSITAET
1
2
3
4
5
6
Artefakt
PolleA
PolleB
Klasse

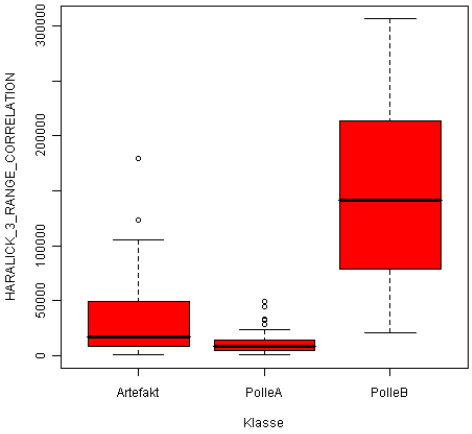

HARALICK_1_MEAN_ANGULAR_SECOND_MOMENT
0.05
0.10
0.15
0.20
0.25
Artefakt
PolleA
PolleB
Klasse

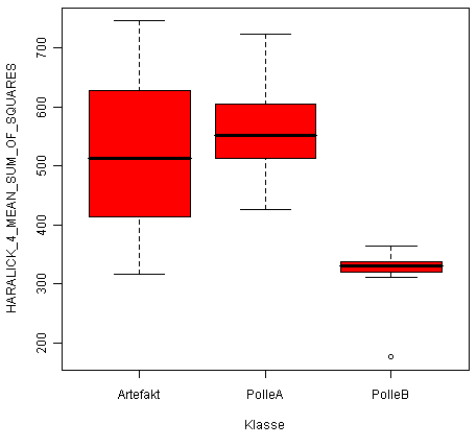

HARALICK_1_RANGE_ANGULAR_SECOND_MOMENT
0.05
0.10
0.15
Artefakt
PolleA
PolleB
Klasse

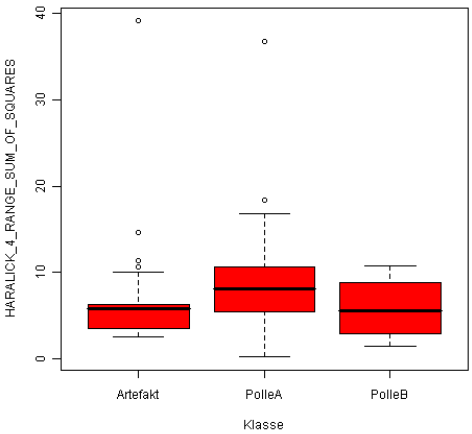

HARALICK_2_MEAN_CONTRAST
0
5
10
15
20
25
30
Artefakt
PolleA
PolleB
Klasse

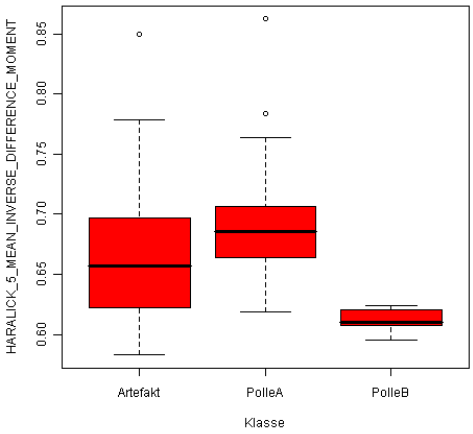

HARALICK_2_RANGE_CONTRAST
0
20
40
60
80
Artefakt
PolleA
PolleB
Klasse

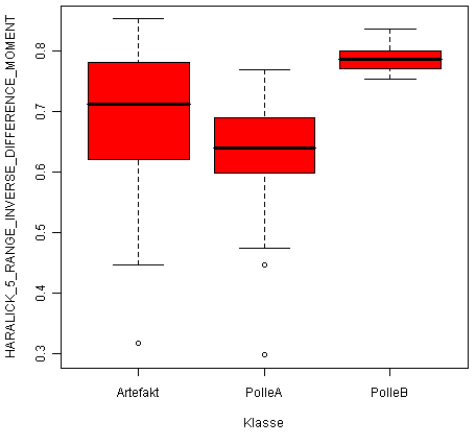

HARALICK_3_MEAN_CORRELATION
500000
1000000
1500000
Artefakt
PolleA
PolleB
Klasse

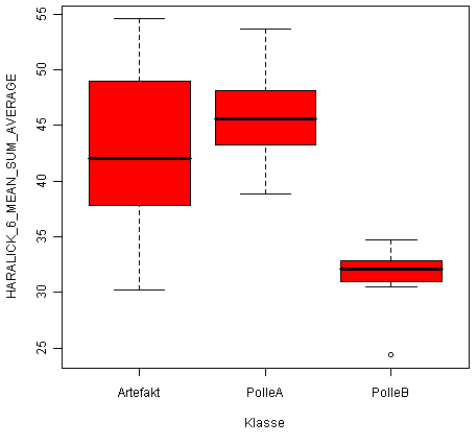

HARALICK_3_RANGE_CORRELATION
0
50000
100000
200000
300000
Artefakt
PolleA
PolleB
Klasse

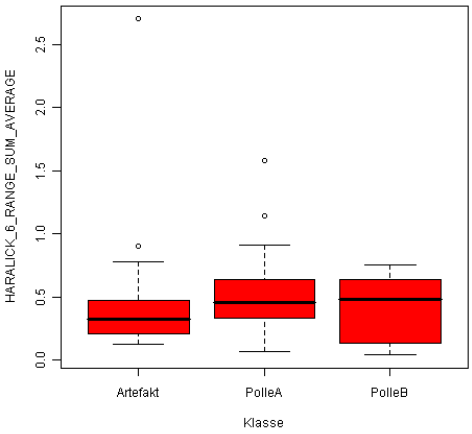

HARALICK_4_MEAN_SUM_OF_SQUARES
200
300
400
500
600
700
Artefakt
PolleA
PolleB
Klasse

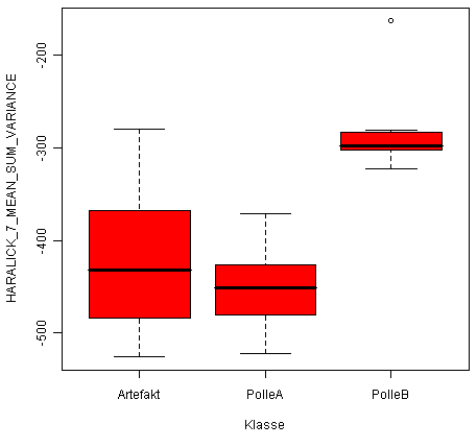

HARALICK_4_RANGE_SUM_OF_SQUARES
0
10
20
30
40
Artefakt
PolleA
PolleB
Klasse

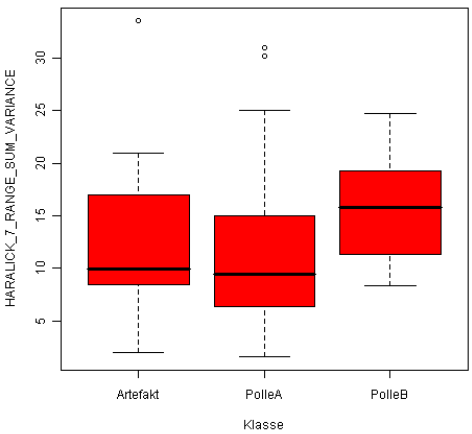

HARALICK_5_MEAN_INVERSE_DIFFERENCE_MOMENT
0.60
0.65
0.70
0.75
0.80
0.85
Artefakt
PolleA
PolleB
Klasse

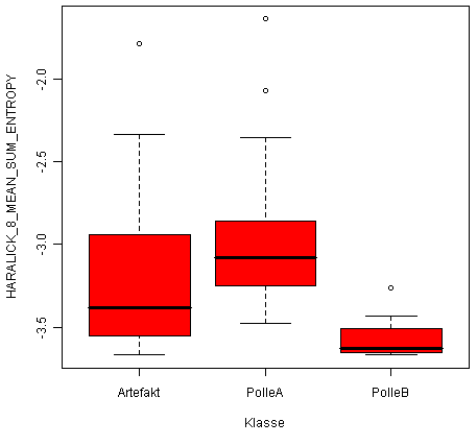

HARALICK_5_RANGE_INVERSE_DIFFERENCE_MOMENT
0.3
0.4
0.5
0.6
0.7
0.8
Artefakt
PolleA
PolleB
Klasse

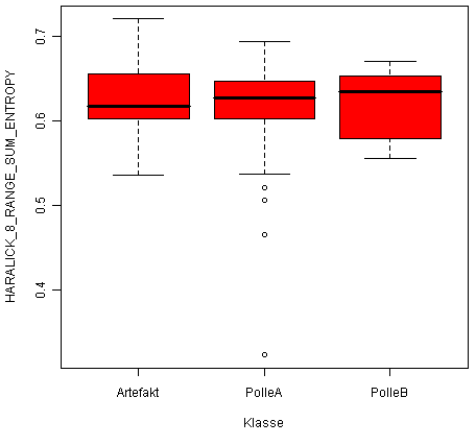

HARALICK_6_MEAN_SUM_AVERAGE
25
30
35
40
45
50
55
Artefakt
PolleA
PolleB
Klasse

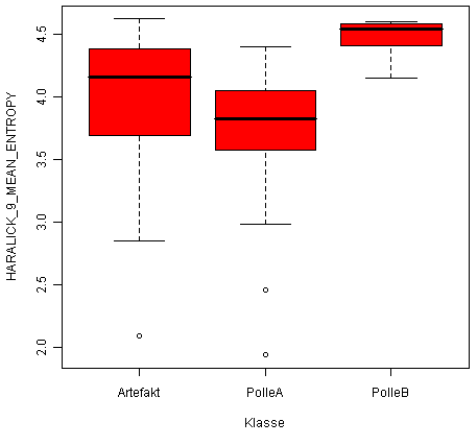

HARALICK_6_RANGE_SUM_AVERAGE
0.0
0.5
1.0
1.5
2.0
2.5
Artefakt
PolleA
PolleB
Klasse

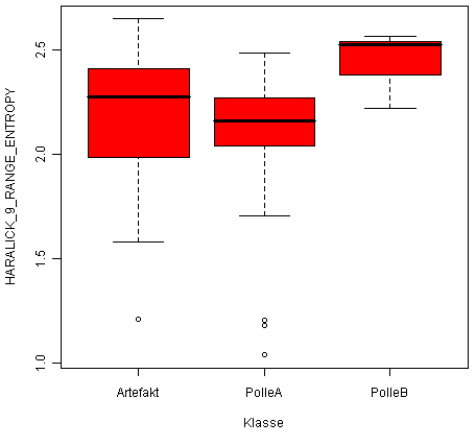

HARALICK_7_MEAN_SUM_VARIANCE
-500
-400
-300
-200
Artefakt
PolleA
PolleB
Klasse

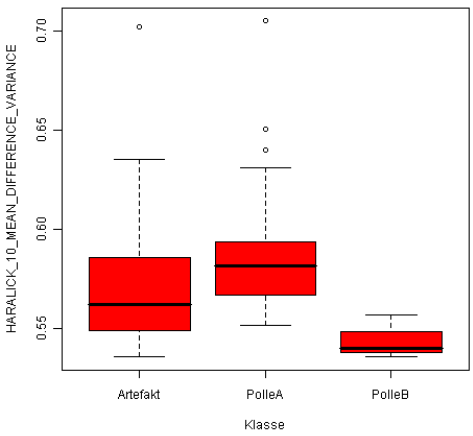

HARALICK_7_RANGE_SUM_VARIANCE
5
10
15
20
25
30
Artefakt
PolleA
PolleB
Klasse

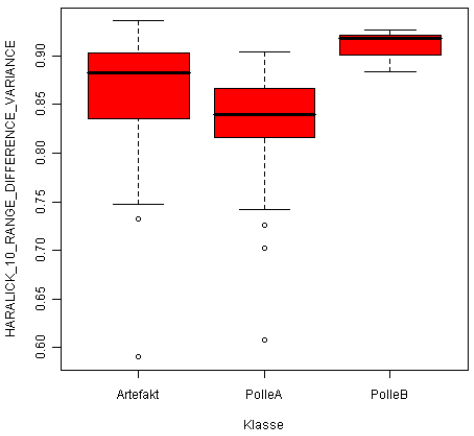

HARALICK_8_MEAN_SUM_ENTROPY
-2.0
-2.5
-3.0
-3.5
Artefakt
PolleA
PolleB
Klasse

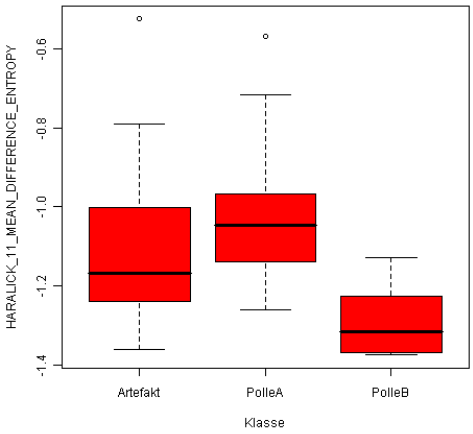

HARALICK_8_RANGE_SUM_ENTROPY
0.7
0.6
0.5
0.4
Artefakt
PolleA
PolleB
Klasse

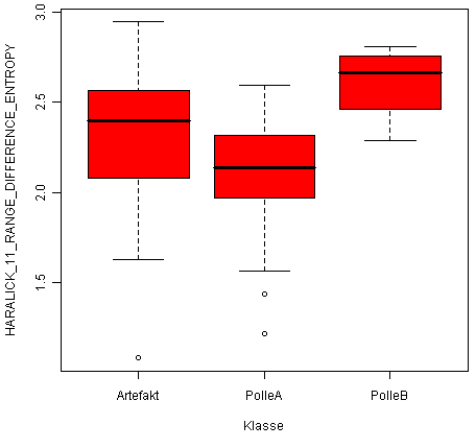

HARALICK_9_MEAN_ENTROPY
4.5
4.0
3.5
3.0
2.5
2.0
Artefakt
PolleA
PolleB
Klasse

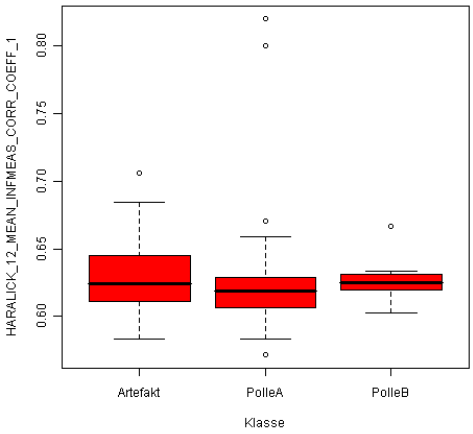

HARALICK_9_RANGE_ENTROPY
2.5
2.0
1.5
1.0
Artefakt
PolleA
PolleB
Klasse

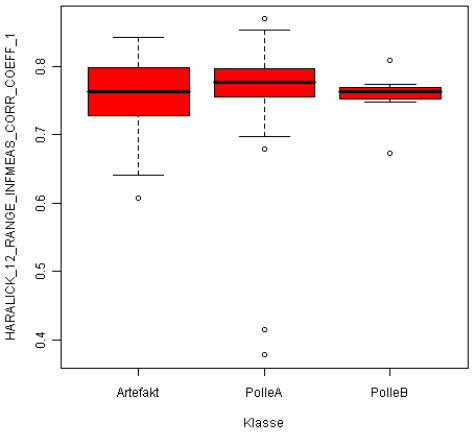

HARALICK_10_MEAN_DIFFERENCE_VARIANCE
0.70
0.65
0.60
0.55
Artefakt
PolleA
PolleB
Klasse

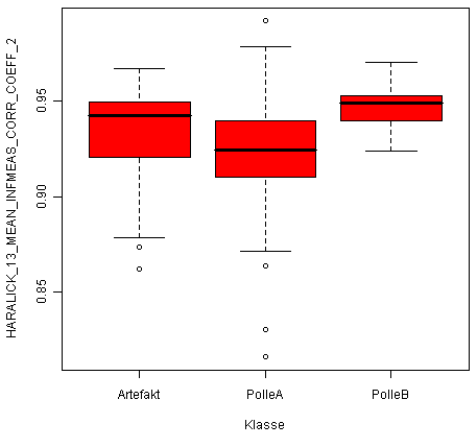

HARALICK_10_RANGE_DIFFERENCE_VARIANCE
0.90
0.85
0.80
0.75
0.70
0.65
0.60
Artefakt
PolleA
PolleB
Klasse

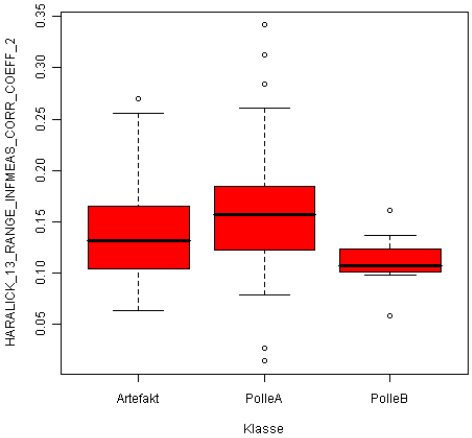

HARALICK_11_MEAN_DIFFERENCE_ENTROPY
-1.4
-1.2
-1.0
-0.8
-0.6
Artefakt
PolleA
PolleB
Klasse


HARALICK_11_RANGE_DIFFERENCE_ENTROPY
1.5
2.0
2.5
3.0
Artefakt
PolleA
PolleB
Klasse

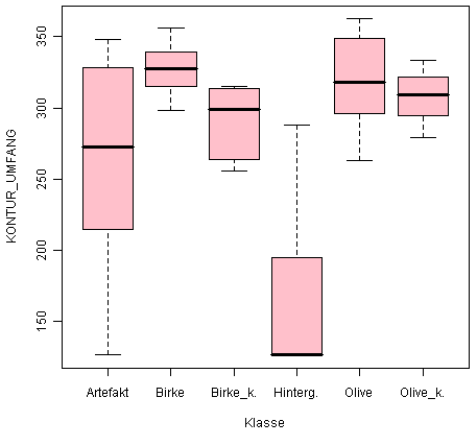

HARALICK_12_MEAN_INFMEAS_CORR_COEFF_1
0.60
0.65
0.70
0.75
0.80
Artefakt
PolleA
PolleB
Klasse

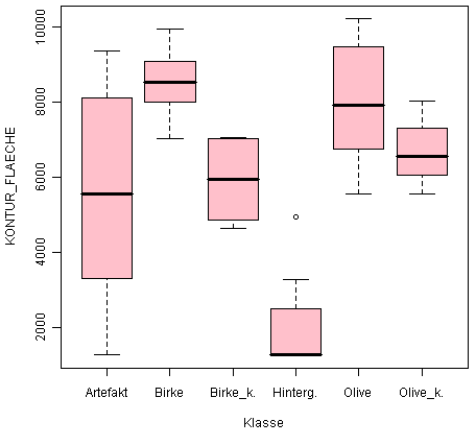

HARALICK_12_RANGE_INFMEAS_CORR_COEFF_1
0.4
0.5
0.6
0.7
0.8
Artefakt
PolleA
PolleB
Klasse

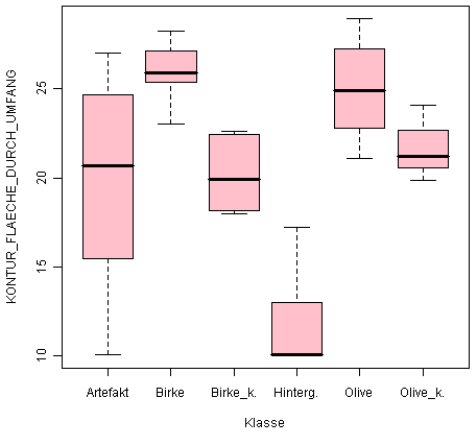

HARALICK_13_MEAN_INFMEAS_CORR_COEFF_2
0.85
0.90
0.95
Artefakt
PolleA
PolleB
Klasse

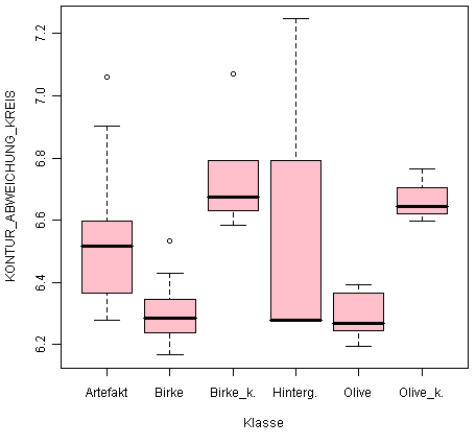

HARALICK_13_RANGE_INFMEAS_CORR_COEFF_2
0.05
0.10
0.15
0.20
0.25
0.30
0.35
Artefakt
PolleA
PolleB
Klasse

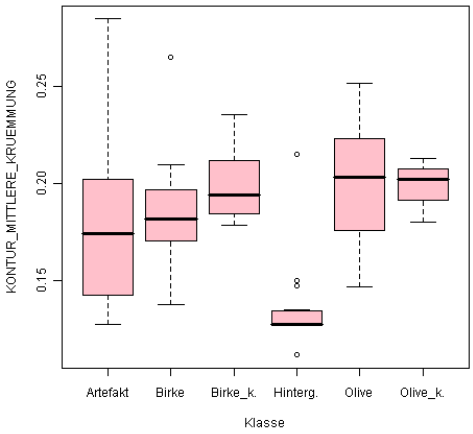
A.2. Aufnahmen in 400facher Vergrößerung

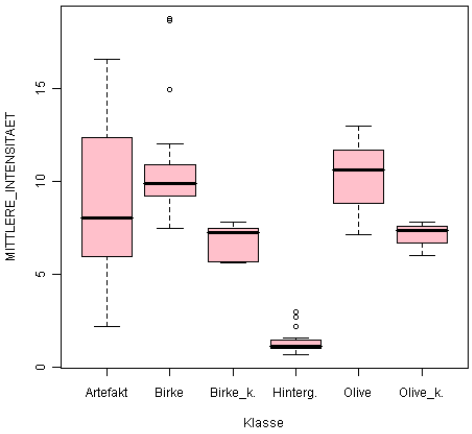


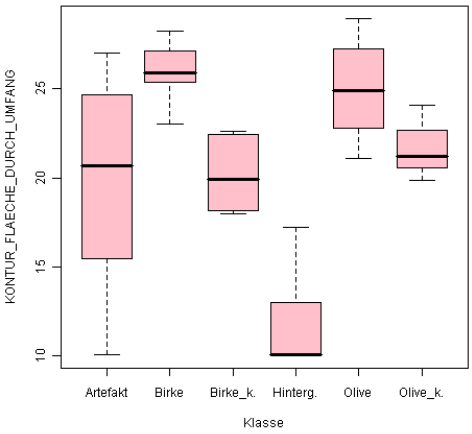


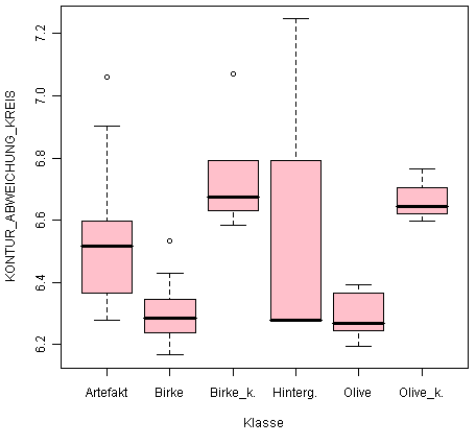


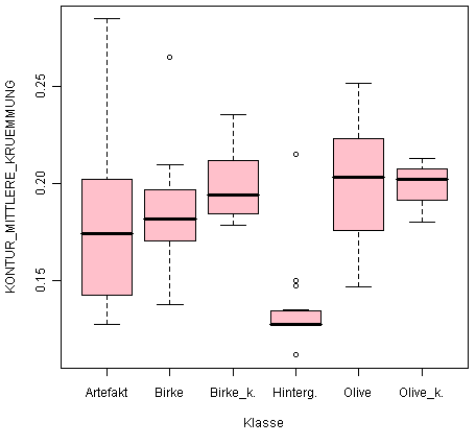


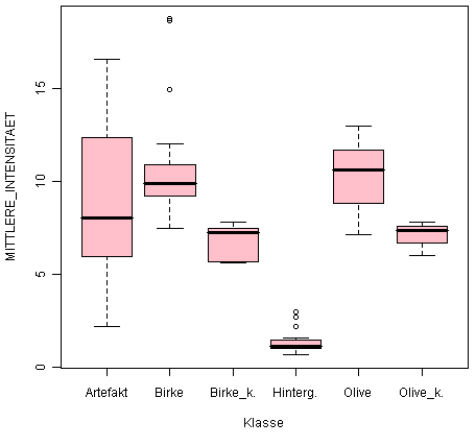


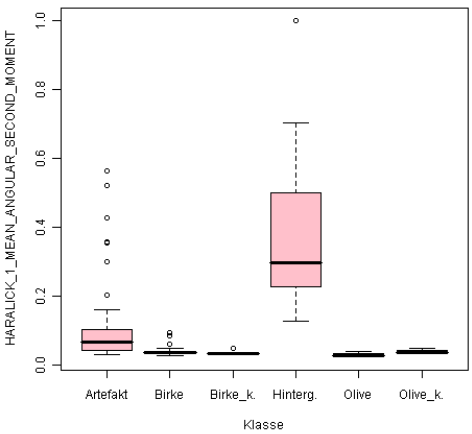


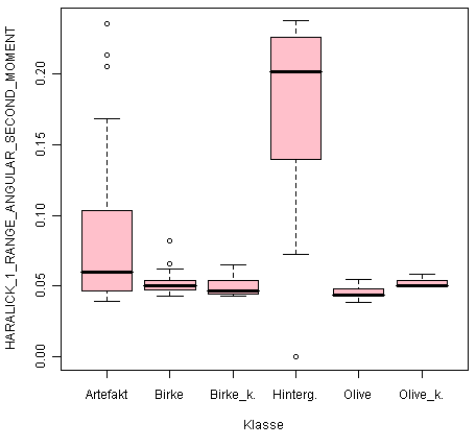

KONTUR_FLAECHE_DURCH_UMFANG
10
15
20
25
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

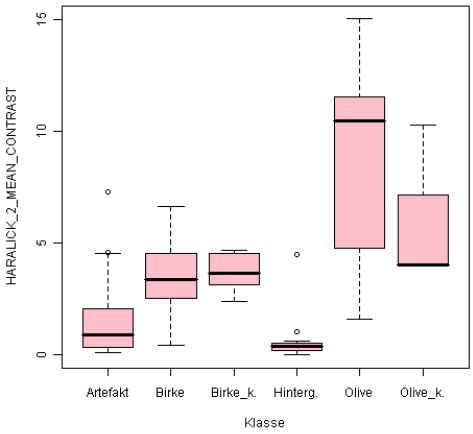

KONTUR_ABWEICHUNG_KREIS
6.2
6.4
6.6
6.8
7.0
7.2
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

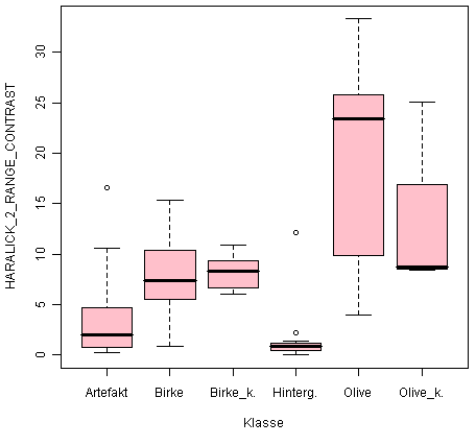

KONTUR_MITTLERE_KRUEMMUNG
0.15
0.20
0.25
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

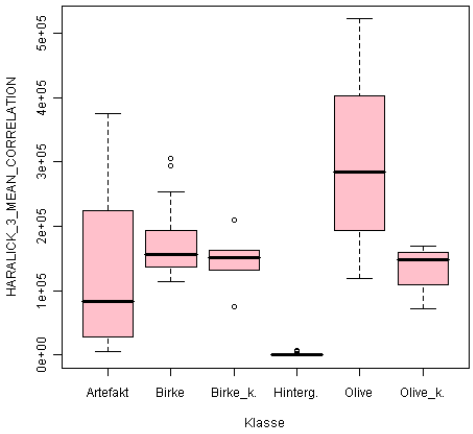

MITTLERE_INTENSITAET
0
5
10
15
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

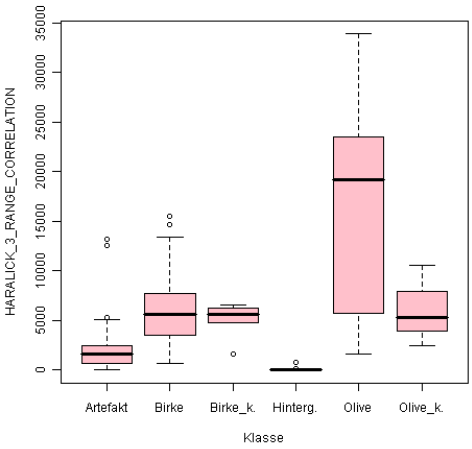

HARALICK_1_MEAN_ANGULAR_SECOND_MOMENT
0.0
0.2
0.4
0.6
0.8
1.0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

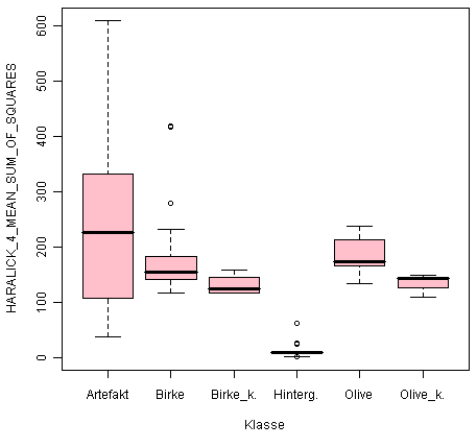

HARALICK_1_RANGE_ANGULAR_SECOND_MOMENT
0.00
0.05
0.10
0.15
0.20
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

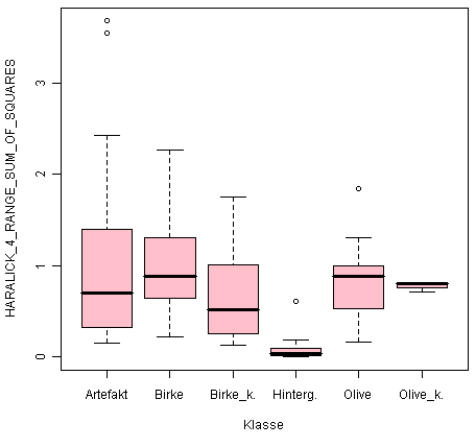

HARALICK_2_MEAN_CONTRAST
0
5
10
15
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

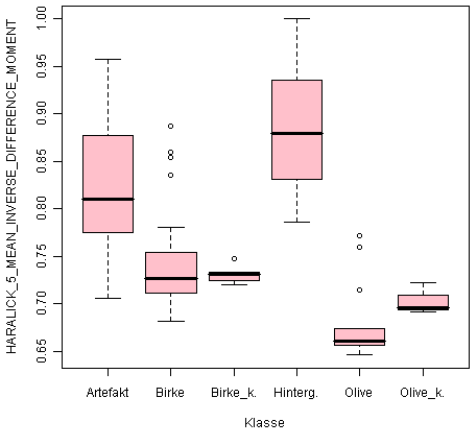

HARALICK_2_RANGE_CONTRAST
0
5
10
15
20
25
30
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

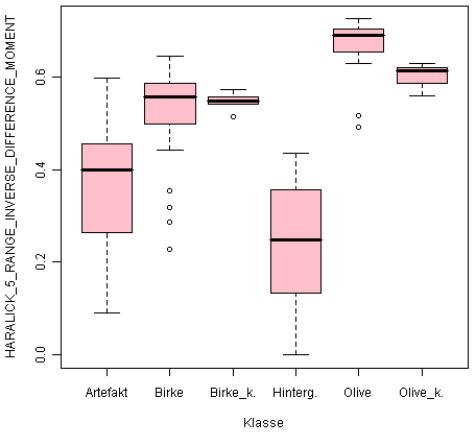

HARALICK_3_MEAN_CORRELATION
0e+00
1e+05
2e+05
3e+05
4e+05
5e+05
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

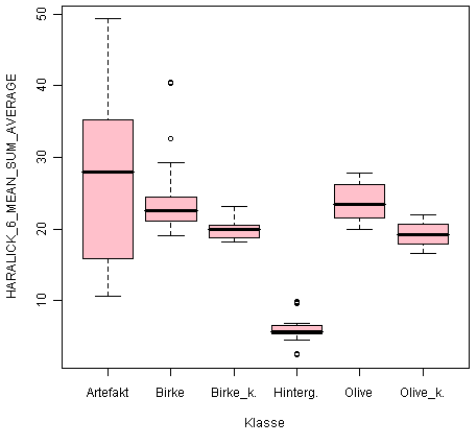

HARALICK_3_RANGE_CORRELATION
0
5000
10000
15000
20000
25000
30000
35000
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

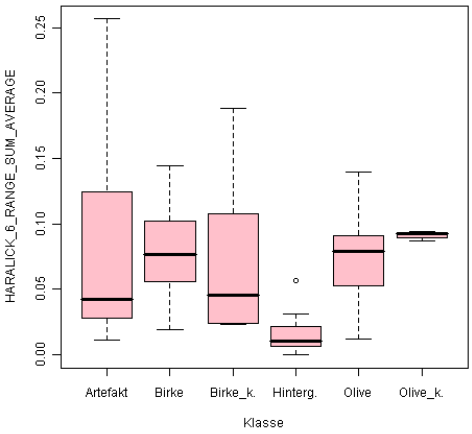

HARALICK_4_MEAN_SUM_OF_SQUARES
0
100
200
300
400
500
600
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

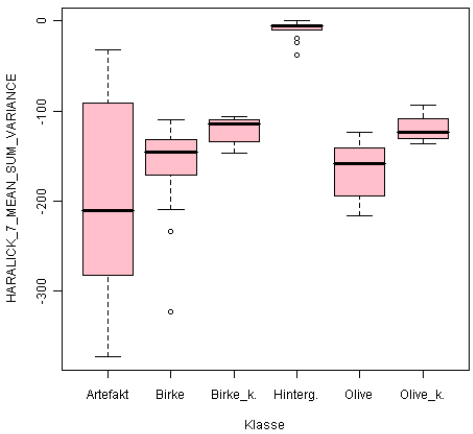

HARALICK_4_RANGE_SUM_OF_SQUARES
0
1
2
3
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

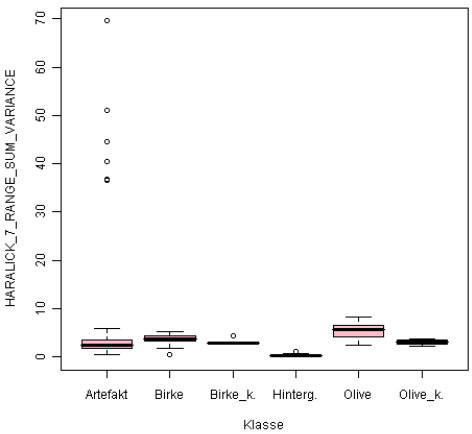

HARALICK_5_MEAN_INVERSE_DIFFERENCE_MOMENT
0.65
0.70
0.75
0.80
0.85
0.90
0.95
1.00
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

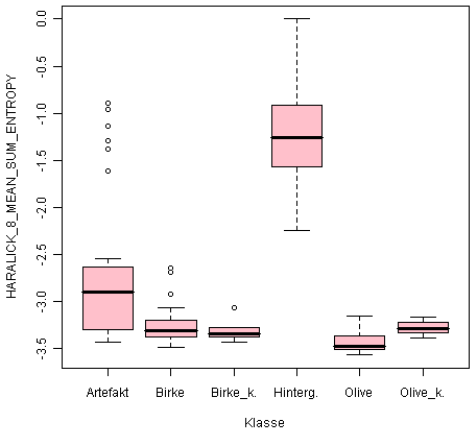

HARALICK_5_RANGE_INVERSE_DIFFERENCE_MOMENT
0.0
0.2
0.4
0.6
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

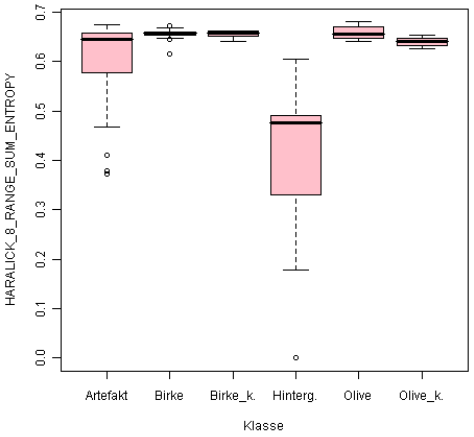

HARALICK_6_MEAN_SUM_AVERAGE
10
20
30
40
50
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

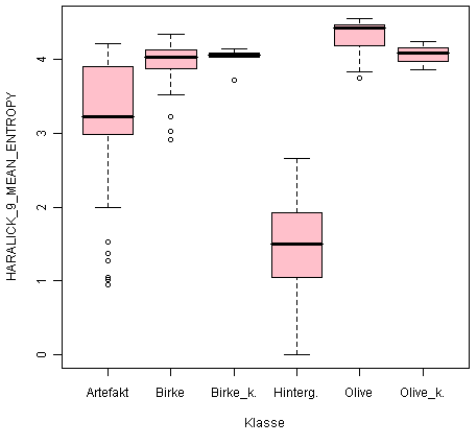

HARALICK_6_RANGE_SUM_AVERAGE
0.00
0.05
0.10
0.15
0.20
0.25
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

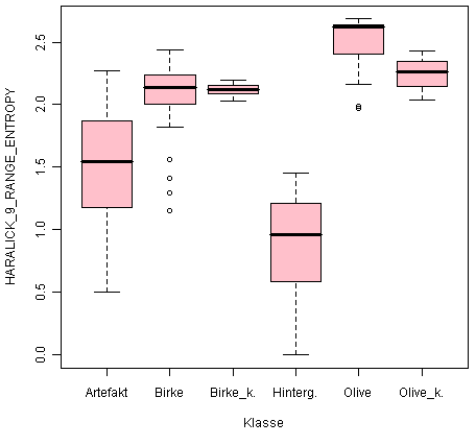

HARALICK_7_MEAN_SUM_VARIANCE
-300
-200
-100
0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

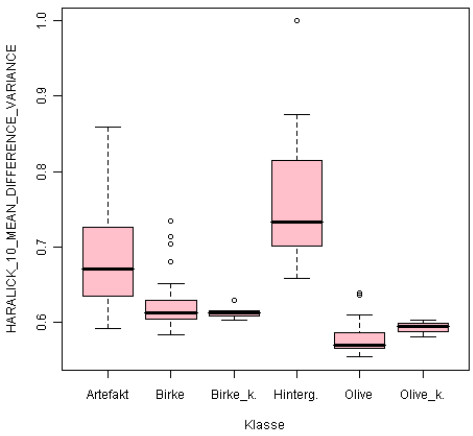

HARALICK_7_RANGE_SUM_VARIANCE
0
10
20
30
40
50
60
70
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

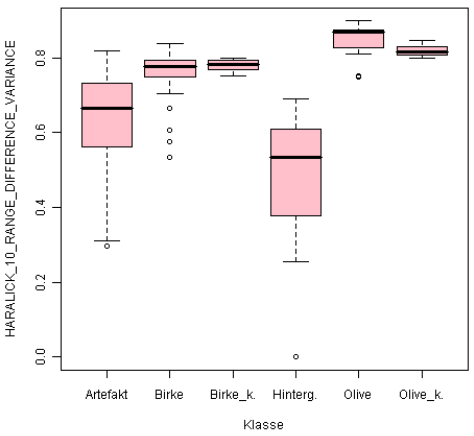

HARALICK_8_MEAN_SUM_ENTROPY
0.0
-0.5
-1.0
-1.5
-2.0
-2.5
-3.0
-3.5
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

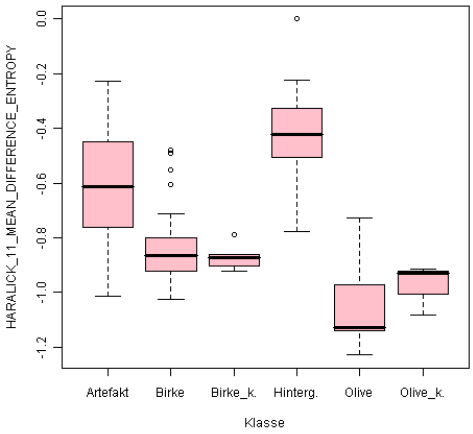

HARALICK_8_RANGE_SUM_ENTROPY
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

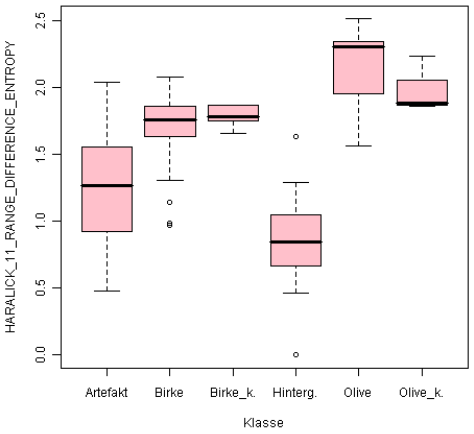

HARALICK_9_MEAN_ENTROPY
4
3
2
1
0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

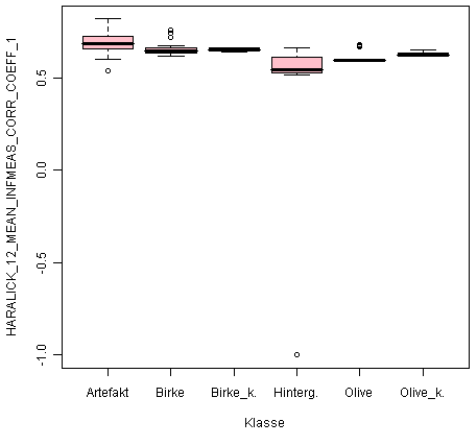

HARALICK_9_RANGE_ENTROPY
2.5
2.0
1.5
1.0
0.5
0.0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

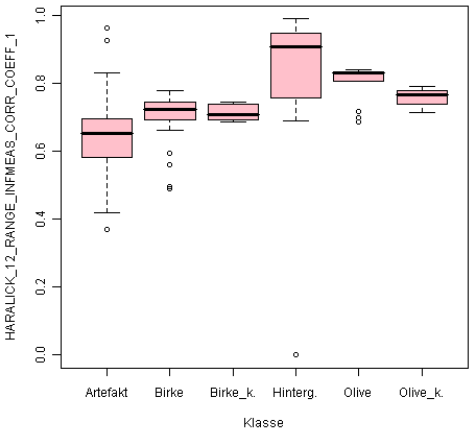

HARALICK_10_MEAN_DIFFERENCE_VARIANCE
1.0
0.9
0.8
0.7
0.6
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

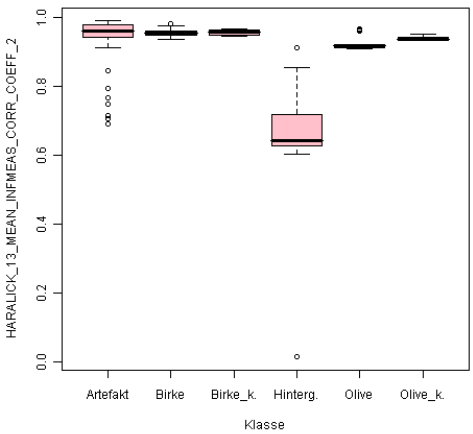

HARALICK_10_RANGE_DIFFERENCE_VARIANCE
0.8
0.6
0.4
0.2
0.0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

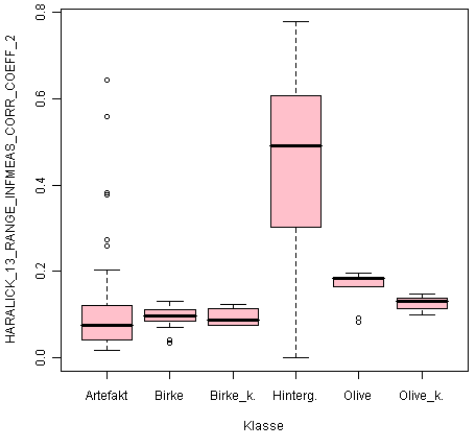

HARALICK_11_MEAN_DIFFERENCE_ENTROPY
0.0
-0.2
-0.4
-0.6
-0.8
-1.0
-1.2
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse


HARALICK_11_RANGE_DIFFERENCE_ENTROPY
2.5
2.0
1.5
1.0
0.5
0.0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse


HARALICK_12_MEAN_INFMEAS_CORR_COEFF_1
0.5
0.0
-0.5
-1.0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse


HARALICK_12_RANGE_INFMEAS_CORR_COEFF_1
1.0
0.8
0.6
0.4
0.2
0.0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse


HARALICK_13_MEAN_INFMEAS_CORR_COEFF_2
1.0
0.8
0.6
0.4
0.2
0.0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse


HARALICK_13_RANGE_INFMEAS_CORR_COEFF_2
0.8
0.6
0.4
0.2
0.0
Artefakt
Birke
Birke_k.
Hinterg.
Olive
Olive_k.
Klasse

Literaturverzeichnis

[Can86] Canny, J.: *A Computational Approach for Edge Detection.* IEEE Transitions on Pattern Analysis an Machine Intelligence, vol. 8, no. 6, bis pp. 679–698, 1986.

[FJ05] Frigo, Matteo und Steven G. Johnson: *The Design and Implementation of FFTW3.* Proceedings of the IEEE, 93(2):216–231, 2005. special issue on "Program Generation, Optimization, and Platform Adaptation".

[Fou06] European Centre For Allergy Research Foundation: *ECARF.* http://www.ecarf.org, 2006.

[GRS+06] G.P.Allen, R.M.Hodgson, S.R.Marsland, G.Arnold, R.C.Flemmer, J.Flenley und D.W.Fountain: *Automatic Recognition of Light-Microscope Pollen Images.* Image and Vision Computing New Zealand, 2006.

[GW01] Gonzalez, Rafael C. und Richard E. Woods: *Digital Image Processing.* Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[HDS73] Haralick, R. M., Dinstein und K. Shanmugam: *Textural features for image classification.* IEEE Transactions on Systems, Man, and Cybernetics, SMC-3:610–621, November 1973.

[Int96] International Telecommunications Union: *Recommendation ITU-R BT.601-5, Studio Encoding Parameters of digital television for standard 4:3 and wide-screen 16:9 aspect ratios*, 1996.

[KWT88] Kass, M., A.P. Witkin und D. Terzopoulos: *Snakes: Active Contour Models.* Int. Journal. Computer Vis.(1), No. 4, bis pp. 321–331, 1988.

[Law80a] Laws, K.: *Rapid texture identification.* In SPIE (The International Society for Optical Engineering), 238, Image Processing for Missile Guidance:376–380, 1980.

[Law80b] Laws, K.: *Textured Image Segmentation, Ph.D. Dissertation*, 1980.

[MK06] Martin Kreuzer, Stefan Kühling: *Logik für Informatiker.* Pearson Studium, München, 2006.

[Mül06] Müller: *Vorlesung Digitale Bildverarbeitung*, 2006.

[Ras07a] Rasband, W.S.: *ImageJ.* http://rsb.info.nih.gov/ij/, 1997-2007.

[Ras07b] Rasband, W.S.: *ImageJ*. U. S. National Institutes of Health, Bethesda, Maryland, USA, 1997-2007.

[RK93] Rudolf Kruse, Jörg Gebhardt, Frank Klawonn: *Fuzzy-Systeme*. Verlag B.G. Teubner, Stuttgart, 1993.

[Sch] Schulze, Mark: *Active Contours (Snakes) - A demonstration using Java*. http://www.markschulze.net/snakes/.

[Sel] Selormey, Paul: *DotNetMatrix*. http://www.codeproject.com/KB/recipes/psdotnetmatrix.aspx.

[SHS+05] Schultz, E., U. Heimann, S. Scharring, A. Brandenburg, M. von Ehr, G. Sulz, H. Burkhardt, O. Ronneberger, Q. Wang, W. Koch, W. Dunkhorst, H. Loedding, W. Müller und G. Breitfuss: *Extracting an Optical Finger-Print – A New Approach to Single Particle Analysis*. 98th Annual Conference and Exhibition of the Air and Waste Management Association, Minneapolis, Minnesota, Paper 1187, 2005.

[Sob70] Sobel, I.E.: *Camera Models and Machine Perception, Ph.D. Dissertation*, 1970.

[SP04] Schuchardt, Petra und Prof. Dr. Wilfried Probst: *Basiswissen Scuhle Biologie Abitur*. Dudenverlag Mannheim, 2004.

[Ste93] Steinbrecher, Rainer: *Bildverarbeitung in der Praxis*. R. Oldenbourg Verlag München Wien, 1993.

[Ste04] Stelzer, Roland: *Fuzzy Logic*. http://wiki.atrox.at/index.php/Fuzzy_Logic, 2004.

[Sza] Szalay, Tamas: *Using FFTW in C#*. http://www.sdss.jhu.edu/~tamas/bytes/fftwcsharp.html.

[Zad65] Zadeh, L. A.: *Fuzzy sets*. Information and Control, 8(3):338–353, June 1965.

[Zad73] Zadeh, L. A.: *Outline of a New Approach to the Analysis of Complex Systems and Decision Processes*. IEEE Transactions on Systems, Man and Cybernetics, 3(1):28–44, 1973.