Segmentierung medizinischer Bilddaten mit dem Schwerpunkt auf statistischen Klassifikatoren

Diplomarbeit

zur Erlangung des akademischen Grads Diplomingenieur
an der Naturwissenschaftlichen Fakultät der
Universität Salzburg

eingereicht von
Franz Wilhelmstötter

Mai 2003

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Motivation und Übersicht

Bildgebende Verfahren spielen eine zentrale Rolle in der modernen Medizin. Seit der Entdeckung der Röntgenstrahlen vor rund einhundert Jahren fand eine rasante Entwicklung statt: Erste nuklearmedizinische Bildgebung und Ultraschall in den 1950er Jahren, Echtzeitultraschall in den 1960er Jahren, Computertomographie in den 1970er Jahren, digitale Radiographie, Doppler-Ultraschall, Kernspintomographie, Positronenemissionstomographie und Videoendoskopie in den 1980er Jahren und funktionelle 3D-Bildgebung in den 1990er Jahren.

Diese Entwicklung wurde begleitet von einem stetig wachsenden Anteil digitaler Rohbilddaten, gefolgt von der Notwendigkeit nach einer automatisierten Aufbereitung und Auswertung. Diese Aufgabe wird nun von der medizinischen Bildbearbeitung übernommen. Sie stellt Methoden und Verfahren zur Verfügung, die bei der Erstellung von diagnostischen und therapeutischen Maßnahmen helfen. (In neuerer Zeit gewinnen hier insbesondere Verfahren für die 3D-Operationsplanung und der computergestützten Chirurgie zunehmend an Bedeutung.) Die medizinische Bildverarbeitung läßt sich in

- *Bildanalyse*,
- *Mustererkennung* und
- *Visualisierung*

einteilen [8]. Dabei hat die Bildanalyse die Extraktion von Objekten und ihre quantitative Beschreibung zum Ziel. Das Thema der vorliegenden Diplomarbeit (die Segmentierung) bildet bei dieser Aufgabe einen wichtigen Teilschritt. Die Aufgabe der Mustererkennung ist die Analyse, Bewertung und Interpretation, der durch die Bildanalysealgorithmen extrahierten, Objekt- und Bildmerkmale. Die

Visualisierung soll die von der Bildanalyse und Mustererkennung erzeugten Ergebnisse in einer Form darstellen, die sie für den Menschen, bzw. den behandelnden Arzt, leicht lesbar machen. Dazu werden bekannte Techniken aus dem Bereich der Computergraphik verwendet.

Die bei der Segmentierung von medizinischen Bilddaten verwendeten Methoden stammen zu einem großen Teil aus der *klassischen* Bildbearbeitung. Die wichtigsten Verfahren werden in Kapitel 2 dargestellt. Ein paar ausgewählte Verfahren, die nicht unbedingt in die klassische Bildbearbeitung passen — wenn man eine etwas engere Definition dieses Begriffs verwendet —, aber in der Segmentierung angewendet werden, werden in Kapitel 3 beschrieben. Den Schwerpunkt dieser Diplomarbeit bildet die Segmentierung mit Hilfe von statistischen Klassifikatoren (Kapitel 4).[1] Hier werden nach einem Überblick über die vorhandenen statistischen Werkzeuge zwei Algorithmen (ML-Klassifikation mittels *k-means* Algorithmus, Kapitel 4.5 und Maximum *a posteriori* Segmentierung, Kapitel 4.6) genauer beschrieben. Die Implementierung dieser Algorithmen ist dann das Thema von Kapitel 5. Im abschließenden Kapitel 6 werden die Segmentierleistungen der implementierten Algorithmen verglichen.

Bevor wir jedoch auf die Segmentierung eingehen, wollen wir in diesem einleitenden Kapitel noch einen kurzen Überblick über die Physik und Funktionsweise zweier wichtiger bildgebender Verfahren in der Medizin geben; diese sind die Röntgen-Computertomographie und die Magnetresonanz-Tomographie.

1.2 Bildgebende Verfahren in der Medizin

In diesem Abschnitt werden zwei wichtige bildgebende Verfahren der Medizin vorgestellt: die Röntgen-Computertomographie (RCT) und die Magnetresonanztomographie (MRT). Beide Verfahren sind Techniken, mit deren Hilfe das Innere eines Körpers nichtinvasiv untersucht werden kann. Sowohl die RCT als auch die MRT haben sich in der Praxis bereits etablieren und sind in der Diagnostik nicht mehr wegzudenken.

[1] Die beschriebenen Segmentierverfahren lassen sich prinzipiell auch auf andere Problembereiche anwenden, wie z. B. Auswertung von Satelitenaufnahmen. Der Schwerpunkt der beiden implementierten Algorithmen liegt jedoch eindeutig in der Segmentierung von T_1 gewichteten MR-Bildern (Abschnitt 1.2.2.1). Die Beurteilung der Segmentierergebnisse erfolgte deshalb auch auf solchen Datensätzen.

1.2.1 Röntgen-Computertomographie

1.2.1.1 Physik

Die für die Röntgen-Computertomographie (RCT oder CT) benötigten Röntgenstrahlen werden technisch in Vakuumröhren erzeugt. Dabei treten durch den thermoelektrischen Effekt Elektronen aus der Kathode aus und werden durch Hochspannung in Richtung Anode beschleunigt. Beim Aufprall der Elektronen auf das Anodenmaterial treten verschiedene Effekte auf:

- Das beschleunigte Elektron kollidiert mit einem Elektron aus der Atomhülle; ist die Stoßenergie groß genug, wird das getroffene Elektron aus der Hülle herausgeschlagen und kann seinerseits mit anderen Elektronen kollidieren.

- Das getroffene Elektron wird nicht vollständig aus der Hülle gestoßen, sondern nur auf ein höheres Energieniveau befördert. Dadurch entstehen Löcher in der Elektronenbesetzung. Elektronen aus höheren Energieniveaus können diese Löcher besetzen. Die daraus resultierende Energiedifferenz wird in Form von elektromagnetischer Strahlung (Photonen) abgegeben. Die Frequenz der abgegebenen Strahlung hängt stark vom verwendeten Anodenmaterial ab.

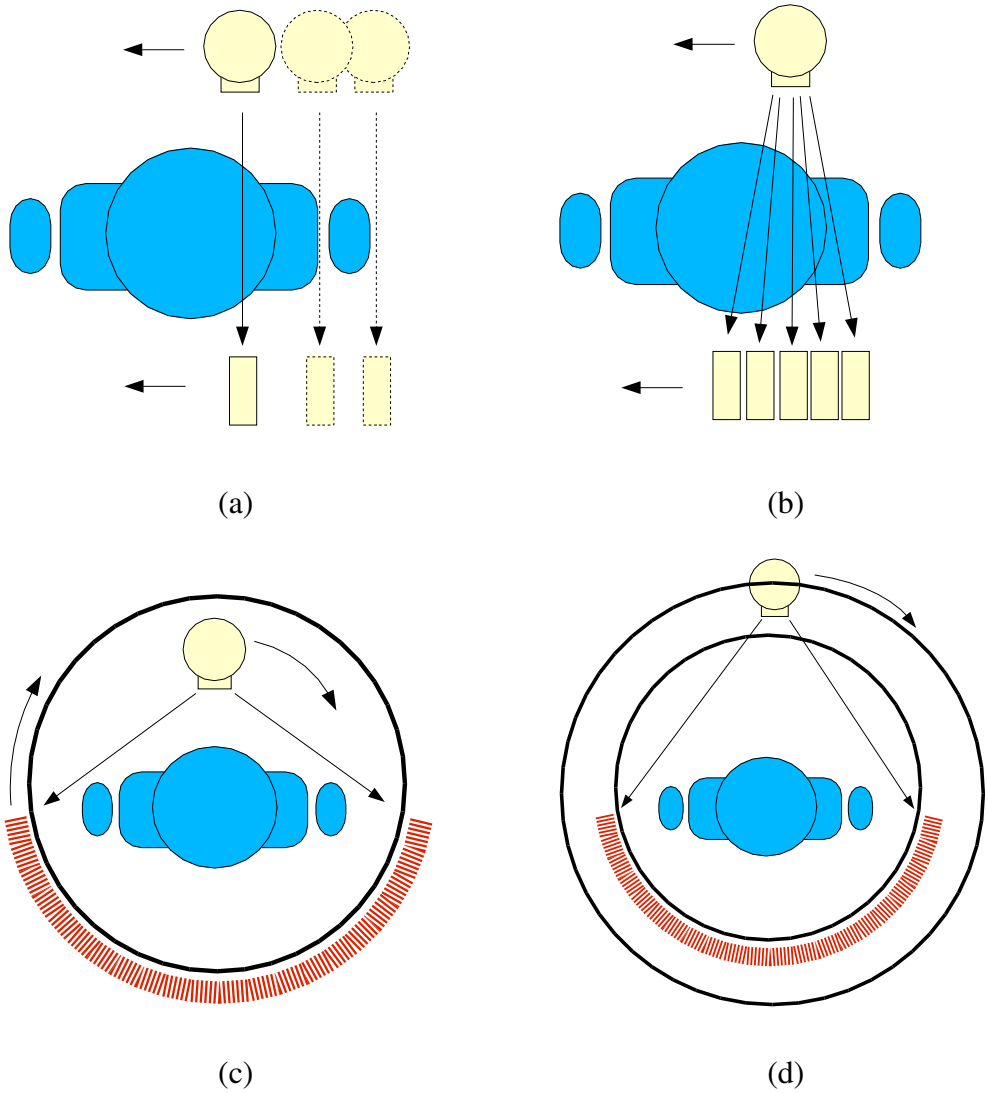
- Die meisten der beschleunigten Elektronen fliegen jedoch in mehr oder weniger großem Abstand an den Elektronen bzw. Kernen vorbei, dabei wird die Richtung der Elektronen geändert und eine elektromagnetische Bremsstrahlung abgegeben.

Die freigesetzte Strahlung ist die Überlagerung der oben genannten Effekte und somit über ein weites Frequenzspektrum verteilt.

Für ein bildgebendes Verfahren ist die Wechselwirkung der Röntgenquanten mit der zu untersuchenden Materie von Interesse. Die verschiedenen Arten der Wechselwirkung werden in einem Schwächungskoeffizienten μ (mit der Einheit m^{-1}) zusammengefaßt. Für ein ausreichend dickes homogenes Material gilt:

$$N = N_0 \cdot e^{-\mu \cdot d}, \tag{1.1}$$

wobei N_0 die Anzahl der auf das Material auftreffenden Quanten pro Sekunde, N die Anzahl der das Material verlassenden Quanten und d die Dicke des Materials ist.



(a) (b)

(c) (d)

Abbildung 1.1: Diese Abbildung zeigt die schematische Darstellung der verschiedenen Scannergenerationen: (a) 1. Generation, (b) 2. Generation, (c) 3. Generation und (d) 4. Generation.

1.2.1.2 Geräte

Die verschiedenen (bis heute entwickelten) Geräte lassen sich in vier Generationen einteilen (siehe Abbildung 1.1):

1. In der ersten Generation wurde einer einzelnen Röntgenröhre genau ein Detektor gegenübergestellt. Durch eine Transversalbewegung wurde das gesamte Objekt durchleuchtet und man erhielt somit eine eindimensionale Verteilung des mittleren Schwächungskoeffizienten. Nach abgeschlossener Aufnahme wurde die Röhre und der Detektor um einen Winkel gedreht und die Aufnahme wiederholt. Die Nachteile dieses Scannertyps waren die hohe Strahlenbelastung für den Patienten und die lange Aufnahmezeit.

2. Die zweite Generation brachte eine Erhöhung der Detektoren mit sich (10-100). Der Strahl wurde dabei aufgefächert. Durch diese Parallelisierung konnte die Schrittweite der Translations- und der Rotationsbewegung vergrößert werden; dies führte zu einer schnelleren Aufnahme.

3. In der dritten Generation wurde der Röntgenstrahl nochmals aufgeweitet. Das Strahlenbündel überdeckte nun den gesamten Patienten, was die Transversalbewegung überflüssig machte. Die Vergrößerung der Anzahl der benutzten Detektoren brachte nicht nur eine verbesserte Auflösung mit sich, sondern erhöhte auch die Wahrscheinlichkeit einer korrekten Rekonstruktion des Objekts. Außerdem wurde die Strahlendosis für den Patienten weiter verringert.

4. Die Geräte der vierten Generation arbeiten mit einem feststehenden Detektorring. Der Grundgedanke ist eine Röntgenröhre, die sich um den Patienten bewegt, sich aber innerhalb des Detektorringes befindet.

1.2.1.3 Bildrekonstruktion

Ziel der Computer-Tomographie ist es, Schichtbilder zu erzeugen, die sich zu einem 3D-Bild des Patienten bzw. Objektes zusammenfügen lassen. Zur Veranschaulichung dieses Ansatzes wählen wir einen nadelförmigen Strahl, der das Objekt durchdringt. Das Signal, welches gemessen wird, ist die transmittierte Röntgenintensität bezogen auf die eingestrahlte Röntgenintensität [5]:

$$J = J_0 \cdot e^{-\int_{s_1}^{s_2} \mu(s) ds}, \tag{1.2}$$

$$\ln\left(\frac{J_0}{J}\right) = \int_{s_1}^{s_2} \mu(s)\, ds, \tag{1.3}$$

wobei J_0 die gesamte Strahlenleistung und J die gesamte durchgetretene Strahlenleistung bezeichnet.

Bei der CT wird die Funktion $\mu(x,y)$ gesucht.[2] *Da unterschiedliche Materialien (Gewebetypen) unterschiedliche Schwächungskoeffizienten besitzen, kann so ein Bild des Körperinneren erzeugt werden.*

Die Rekonstruktion der Funktion $\mu(x,y)$ aus den Linienintegralen $\int_{s_1}^{s_2} \mu(s)\,ds$ erfolgt durch Anwenden der RADON-TRANSFORMATION und des sogenannten FOURIER-SCHEIBENTHEOREMS.

Radon-Transformation Bei der Radon-Transformation geht es darum, eine beliebige, integrierbare Funktion $f : \mathbb{R}^2 \to \mathbb{R}$ durch alle geraden Linienintegrale über das Definitionsgebiet von f zu beschreiben.

$$p(\theta, r) := \int_{-\infty}^{+\infty} f(x(s), y(s))\,ds, \tag{1.4}$$

mit $x(s) := r \cdot \cos\theta - s \cdot \sin\theta$ und $y(s) := r \cdot \sin\theta + s \cdot \cos\theta$ (siehe Abbildung 1.2). Werden alle Winkel $\theta \in [0, \pi]$ und alle Werte für $r \in [r_{min}, r_{max}]$ der Reihe nach durchlaufen, so erhält man alle Linienintegrale $p(\theta, r)$ über die Funktion f. Die Funktion p wird als die Radontransformation von f bezeichnet. Eine Linie der Radontransformierten mit $\theta = \text{const}$ bezeichnet man als die Projektion $p_\theta(r)$.

Fourier-Scheibentheorem Durch das Fourier-Scheibentheorem ist es möglich die Funktion $f(x,y)$ aus ihrer Radontransformation $p(\theta, r)$ wiederherzustellen. Dafür definieren wir zunächst die 1D- und 2D-Fouriertransformation und deren Inverse.

Definition 1.2.1 *Sei $f : \mathbb{R} \to \mathbb{R} : x \mapsto f(x)$ eine integrierbare Funktion. Die* 1D-FOURIERTRANSFORMATION *$F : \mathbb{R} \to \mathbb{C} : u \mapsto F(u)$ ist gegeben durch:*

$$F(u) := \int_{-\infty}^{+\infty} f(x) \cdot e^{-j \cdot 2\pi \cdot ux} dx. \tag{1.5}$$

Die zugehörige inverse Fouriertransformation ist:

$$f(x) := \int_{-\infty}^{+\infty} F(u) \cdot e^{j \cdot 2\pi \cdot ux} du. \tag{1.6}$$

Sei $f : \mathbb{R}^2 \to \mathbb{R} : (x,y) \mapsto f(x,y)$ wieder eine integrierbare Funktion, die 2D-FOURIERTRANSFORMATION *$F : \mathbb{R}^2 \to \mathbb{C} : (u,v) \mapsto F(u,v)$ ist gegeben durch:*

$$F(u,v) := \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x,y) \cdot e^{-j \cdot 2\pi \cdot (ux+vy)} dx dy. \tag{1.7}$$

[2] Dies ist der Röntgenschwächungskoeffizienten als Funktion des Ortes für eine „Scheibe".

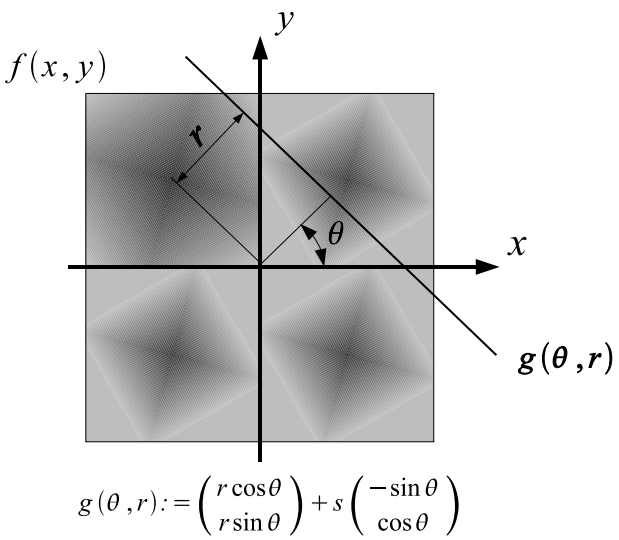


Abbildung 1.2: Diese Abbildung zeigt die Systematisierung der Geraden, die den gesamten Bildraum abdecken. Entlang diesen Geraden wird (für die Radon-Transformation) integriert.

Die zugehörige inverse Fouriertransformation ist hier:

$$f(x,y) := \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} F(u,v) \cdot e^{j \cdot 2\pi \cdot (ux+vy)} dudv. \tag{1.8}$$

Theorem 1.2.1 FOURIER-SCHEIBENTHEOREM

Sei $f(x,y)$ gegeben und $F(u,v)$ deren 2D-Fouriertransformierte. Weiters sei $p_\theta(r)$ eine Projektion von $f(x,y)$ und $P_\theta(w)$ deren 1D-Fouriertransformierte. Dann beschreibt $P_\theta(w)$ die Werte von $F(u,v)$ auf einem Radialstrahl zum Winkel θ (siehe Abbildung 1.3).

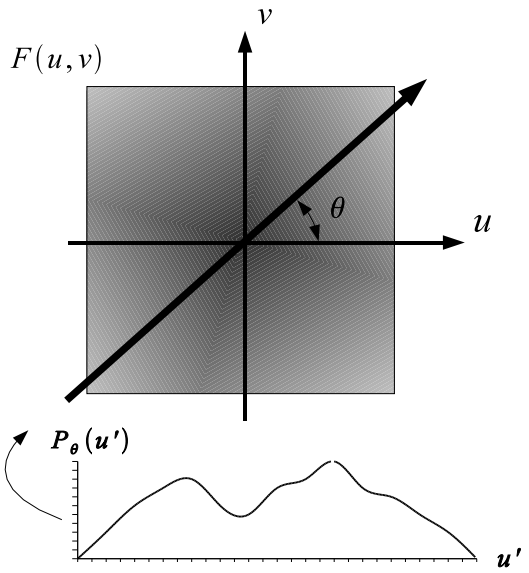


Abbildung 1.3: Diese Abbildung stellt das Fourier-Scheibentheorem dar, für einen beliebigen Winkel θ.

Rekonstruktion Für die Rekonstruktion werden möglichst viele Projektionen $p_\theta(r)$ mit einer möglichst großen Zahl von Meßpunkten aufgenommen. Danach werden die fouriertransformierten Projektionen $P_\theta(r)$ in eine (komplexe) Matrix $F_{u,v}$ eingetragen. Dabei muß von den bekannten Werten der Radialstrahlen auf die benötigten Werte in der Matrix interpoliert werden. Durch eine (diskrete) inverse Fouriertransformation von $F_{u,v}$ wird schließlich das CT-Bild gewonnen.

1.2.2 MR-Tomographie

1.2.2.1 Physik

Bei der Magnetresonanz-Tomographie (MRT) nutzt man die Tatsache, daß Protonen einen Spin besitzen und sich damit wie magnetische *Kreisel* verhalten. Wird ein rotierender Kern in ein statisches magnetisches Feld B_0 gebracht, so richtet sich dieser nach B_0 aus. Durch das Ausrichten beginnt der Kern mit einer Präzessionsbewegung[3] — die Rotationsachse des Kerns wird in Rotation versetzt (Abbildung 1.4). Die Präzessionsbewegung tritt jedesmal dann auf, wenn der Kern

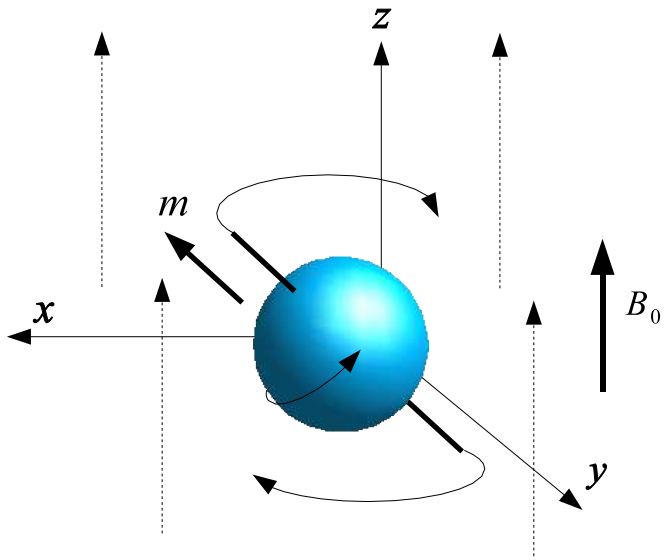


Abbildung 1.4: Diese Abbildung zeigt die Präzessionsbewegung der Kerndrehachse in einem äußeren Magnetfeld B_0. Dabei bezeichnet m das magnetische Moment des Kerns.

aus seiner Ruhelage gebracht wird. Wird das äußere Feld wieder abgestellt, so fällt der Kern in seine ursprünglich Lage (thermisches Gleichgewicht) zurück[4]. Wird ein zweites Feld (Transversalfeld) B_T angelegt, welches normal zum ersten steht,

[3] Als Präzessionsbewegung bezeichnet man die „Taumelbewegung“ der Rotationsachse des rotierenden Kerns (Abbildung 1.4).

[4] Die durch die Drehung aufgenommene Energie wird in Form von elektromagnetischer Strahlung wieder abgegeben.

beginnt der Kern wieder zu präzedieren — ebenso wenn das Feld wieder abgestellt wird. Um die Kerne dauerhaft zur Präzession anzuregen, ist dieses zweite Feld ein hochfrequentes Wechselfeld (HF-Feld) und rotiert in der *xy*-Ebene (Abbildung 1.5).

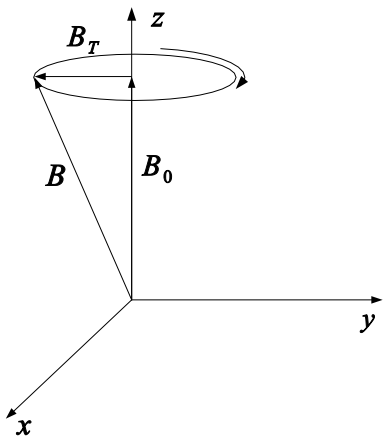


Abbildung 1.5: Diese Abbildung zeigt das resultierendes Magnetfeld B aus dem statischen Feld B_0 und dem rotierenden Transversalfeld B_T.

Für die Präzessionsbewegung existiert eine Resonanzfrequenz. Bei Atomkernen wird diese Eigenfrequenz *Lamorfrequenz* genannt. Diese ist abhängig von der Stärke des eingeprägten Magnetfeldes und vom Aufbau des Kerns. Durch die Wahl der Stärke des ersten (statischen) Feldes B_0 und die Wahl der Frequenz des Transversalfeldes B_T kann sehr genau bestimmt werden, welche Kerne in Resonanz geraten sollen. Durch diesen Resonanzeffekt wird das magnetische Moment m des Kerns um 90° in die *xy*-Ebene gekippt und rotiert präzedierend mit dem Transversalfeld.

Wird das transversale Wechselfeld, welches das magnetische Moment m eines Kerns um 90° gekippt hat, abgeschaltet, so rotiert der Kern weiter in der *xy*-Ebene. Bringt man nun eine Spule in die Nähe des rotierenden magnetischen Moments, so wird in diese eine Spannung induziert. Da die Meßspulen gewöhnlich normal auf die *xy*-Ebene stehen, ist die gemessene Spannung proportional zur Quermagnetisierung m_T des magnetischen Momentes m (Abbildung 1.6).

Mit einer Folge von HF-Impulsen des Transversalfeldes in einem Körper, der in einem starken Magnetfeld liegt, kann eine *rotierende* Quermagnetisierung M_T erzeugt werden, welche sich aus den Quermagnetisierungen m_T der einzelnen

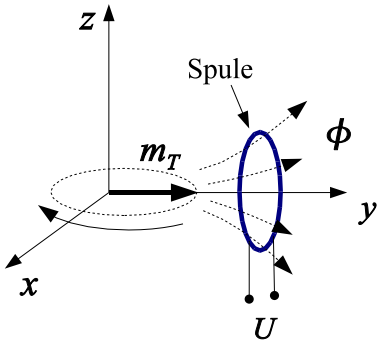


Abbildung 1.6: Der magnetische Fluß des rotierenden Dipols induziert in der Meßspule eine Spannung U. Die induzierte Spannung ist dabei proportional zur Quermagnetisierung m_T (des magnetischen Momentes m), wenn die Spule normal zur xy-Ebene orientiert ist.

Kerne zusammensetzt. Diese Quermagnetisierung ist vom Ort und vom Gewebetyp abhängig.

Das Ziel der MR-Tomographie ist die Erzeugung von Schichtbildern der Quermagnetisierung $M_T(x,y)$.

Spin-Gitter-Relaxation (Längsrelaxation T_1) Wie bereits dargelegt, können Kerne durch Einbringen eines Hochfrequenzsignals zur Präzession gebracht werden. Wird dieses Signal in der xy-Ebene ausreichend lange eingebracht, präzedieren alle Kerne in der xy-Ebene; die z-Komponente der Magnetisierung nimmt den Wert Null an. Stellt man das Signal ab, so kommt es durch Wechselwirkung mit den umgebenden Atomen zu einer Relaxation (Spin-Gitter-Relaxation), d. h. die Magnetisierungsvektoren richten sich wieder entlang des statischen Feldes B_0 aus. Diese Ausrichtung erfolgt exponentiell:

$$M_z(t) = M_0 \cdot \left(1 - e^{-\frac{t}{T_1}}\right), \tag{1.9}$$

wobei M_0 die Stärke der Magnetisierung in Richtung von B_0 im Gleichgewichtszustand ist. Die Zeit bis die z-Komponente ca. 63% ihres Ausgangswertes wieder erreicht hat, nennt man Spin-Gitter-Relaxationszeit oder auch T_1-Zeit (Abbildung 1.7).

Flüssigkeiten haben wesentlich kürzere T_1-Zeiten als Festkörper, der Grund

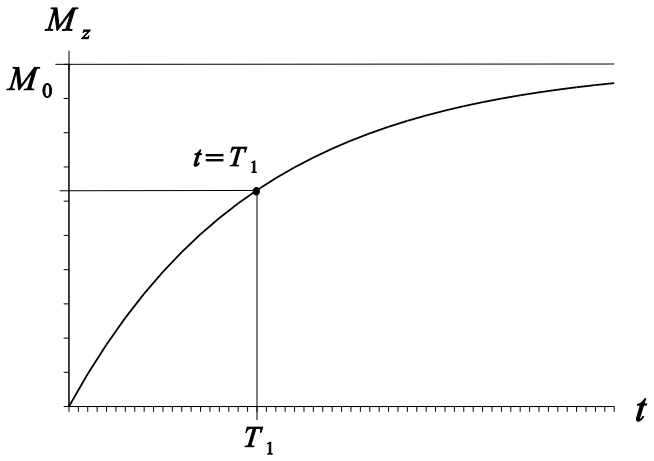


Abbildung 1.7: Diese Abbildung zeigt die Spin-Gitter-Relaxationszeit.

dafür ist, daß die Moleküle in Flüssigkeiten, aufgrund ihrer größeren Beweglichkeit, schneller ihre Energie über kinetische Stöße an andere Moleküle/Atome abgeben können; der Gleichgewichtszustand wird somit schneller wieder erreicht.[5]

Spin-Spin-Relaxation (Querrelaxationzeit T_2) Die Quermagnetisierung eines Spin-Ensembles zerfällt nun ähnlich wie die M_z-Komponente, durch Wechselwirkung mit benachbarten Atomen. Hier ist es allerdings die Spin-Spin-Wechselwirkung, die für die Dephasierung verantwortlich ist. (Dies ist die Wechselwirkung mit anderen magnetischen Kreiseln.) Der Zerfall läßt sich wieder durch eine Exponentialfunktion darstellen, jedoch mit einer anderen Zeitkonstante T_2:

$$M_T(t) = M_0 \cdot e^{-\frac{t}{T_2}}. \tag{1.10}$$

Da bei erreichtem Gleichgewichtszustand die Quermagnetisierung null ist, muß T_2 immer kleinergleich T_1 sein. In Wirklichkeit nimmt die Quermagnetisierung in der xy-Ebene viel schneller ab, als durch die Spin-Spin-Wechselwirkung erklärbar ist. Die Ursache liegt darin, daß bei einer MR-Aufnahme über ein Volumenelement gemittelt wird. Nach Wegnahme des HF-Signals verschieben sich die Phasen der Kerne untereinander und die xy-Komponenten der einzelnen Kerne

[5] Mit T_1 gewichteten MR-Bildern kann daher sehr gut der Unterschied zwischen Festkörpern und Flüssigkeiten visualisiert werden.

laufen auseinander. Tabelle 1.1 zeigt typische Werte für die Längs- und Querrelaxationszeit.

Tabelle 1.1: Diese Tabelle zeigt typische Werte für die Längsrelaxationszeit T_1 und der Querrelaxationszeit T_2 [5].

Gewebe	T_1 (ms)	T_2 (ms)	Gewebe	T_1 (ms)	T_2 (ms)
Muskel	730 ±130	47 ±13	Milz	680 ±190	62 ±27
Herz	750 ±120	57 ±16	Fett	240 ±70	84 ±36
Leber	420 ±90	43 ±14	Graue Masse	810 ±140	101 ±13
Niere	590 ±160	58 ±24	Weiße Masse	680 ±120	92 ±22

1.2.2.2 Kodierung und Bildrekonstruktion

Jedes Volumenelement besitzt seine eigene Quermagnetisierung M_T und alle Quermagnetisierungen tragen zum — in der Meßspule aufgenommenen — Signal bei. Die Schwierigkeit der Tomographie ist es nun, die Signale der einzelnen Volumenelemente so zu kodieren, daß daraus das Bild der Quermagnetisierung einer einzelnen Schicht $M_T(x,y)$ rekonstruiert werden kann. Die Kodierung erfolgt auf drei unterschiedliche Arten:

1. Selektive Anregung,
2. Phasenkodierung,
3. Frequenzkodierung.

Diese drei Kodierungsarten zusammengenommen ermöglichen die Rekonstruktion des MR-Bildes.

Selektive Anregung Bei der selektiven Anregung wird während des HF-Impulses, der die Spins umklappen soll, sodaß eine meßbare Quermagnetisierung entsteht, ein Gradientenfeld eingeschaltet. Im Falle eines Gradientenfeldes $G_z = \frac{\partial B_z}{\partial z}$ hängt die Lamorfrequenz[6] von der Position entlang der z-Achse ab. Man kann die Signale einer Schicht empfangen, indem man ein schmales Frequenzband als HF-Impuls benutzt. Werden zusätzlich Gradientenfelder G_x und G_y eingesetzt, so ist es möglich einzelne Volumenelemente direkt anzuregen und deren Quermagnetisierung zu messen. Die direkte Anregung eines Volumenelementes ist jedoch sehr zeitaufwendig.

[6] Dies ist die Resonanzfrequenz des Kerns.

Phasenkodierung Eine Phasenkodierung erreicht man durch Gradientenfelder, die zwischen der HF-Anregung und dem Auslesen der Signale eingeschaltet werden. Dabei wird meist ein Gradientenfeld G_y verwendet. Durch das Einschalten des Feldes G_y werden die Kerne, die durch den HF-Impuls angeregt wurden, aus der Phase gebracht. Auch hier wird durch ein Gradientenfeld G_z nur eine ausgewählte Schicht angeregt. Nach Abschalten des Feldes G_y rotieren die Kerne für unterschiedliche y-Koordinaten mit unterschiedlichen Winkeln, aber mit gleicher Geschwindigkeit, weiter. Die Phasenverschiebung hängt von der Stärke des eingebrachten Feldes G_y und der Dauer, die das Feld anliegt, ab. Von den Meßspulen wird immer nur die resultierende Quermagnetisierung

$$M_T(y) \quad = \quad M_{T_0}(y) \cdot e^{-j\gamma \cdot G_y \cdot y \cdot T_y} \tag{1.11}$$

gemessen. Dabei bezeichnet γ den gyromagnetischen Faktor[7] und T_y die Dauer, die das Feld G_y anliegt. M_{T_0} ist die Stärke der Quermagnetisierung vor Abschalten des HF-Impulses. Die Messungen werden mehrmals mit unterschiedlichen Gradientenfeldstärken durchgeführt.

Frequenzkodierung Eine Frequenzkodierung wird erreicht, wenn *während* des Auslesens des Signals ein Gradientenfeld G_x angelegt wird. Dabei nehmen wir an, daß wir durch einen HF-Impuls und eine anschließende Phasenkodierung ein Muster von Quermagnetisierung $M_T(y)$ erzeugt haben. Wiederum wird durch ein Gradientenfeld G_z eine einzelne Schicht fokussiert. Die transversale Magnetisierung während des Auslesens läst sich folgendermaßen schreiben:

$$M_T(x,t) = M_{T_0}(x) \cdot e^{-j\gamma \cdot G_x \cdot x \cdot t}. \tag{1.12}$$

Diese Gleichung hat große Ähnlichkeit mit Gleichung (1.11). Der Unterschied besteht hier, daß bei der Phasenkodierung durch Abschalten des Gradientenfeldes G_y die Zeit t *angehalten* wird. Bei der Frequenzcodierung läuft die Zeit t weiter und sendet während der Messung eines Volumenelements mit einer anderen Frequenz. Die Ortsinformation ist somit über die Frequenz kodiert.

Bildrekonstruktion Wird mit der selektiven Anregung eine ganze Schicht im Körper angeregt, so wird in der Meßspule die Summe aller Quermagnetisierungen dieser Schicht gemessen. Durch Einbringen eines Phasenkodier- und Frequenzkodiergradienten kann das MR-Bild aus dem gemessenen Signal $S(t,T_y)$ mit den Gleichungen (1.11) und (1.12) direkt berechnet werden:

$$S(t,T_y) = \int\int M_{T_0}(x,y) \cdot e^{\left(-j\gamma \cdot G_x \cdot xt - j\gamma \cdot G_y \cdot yT_y\right)} dxdy. \tag{1.13}$$

[7] Dies ist eine vom Kern abhängige Zahl und ist mitbestimmend für die Lamorfrequenz.

Ersetzt man nun $\gamma \cdot G_x \cdot t$ durch k_x und $\gamma \cdot G_y \cdot T_y$ durch k_y, so erhält man:

$$S(k_x, k_y) = \int \int M_{T_0}(x,y) \cdot e^{\left(-jk_x x - jk_y y\right)} dxdy. \tag{1.14}$$

Theorem 1.2.2 *Gleichung (1.14) ist die Fouriertransformierte der Quermagnetisierung M_{T_0}.*

Theorem 1.2.2 ermöglicht somit die Rekonstruktion der Quermagnetisierung M_T und die Aufnahme eines MR-Bildes.

Kapitel 2

Klassische Segmentiermethoden

In diesem Kapitel werden Segmentiermethoden aus der klassischen Bildverarbeitung vorgestellt. Dabei wollen wir in Abschnitt 2.1 die Erkennung von Diskontinuitäten in digitalen Bildern behandeln. (Dazu zählen das Erkennen von Punkten, Linien und Kanten [7].) Die weiteren Methoden werden dann in kantenorientierte (Abschnitt 2.2) und in bereichsorientierte (Abschnitt 2.3) Verfahren eingeteilt.

2.1 Grundlegende Methoden

Die medizinische Bildverarbeitung muß zwei- und dreidimensionalen Bilder bearbeiten können. Zweidimensionale Bilder entstehen z. B. bei Röntgenaufnahmen oder auch bei Ultraschalluntersuchungen. Computertomographen erzeugen hingegen zweidimensionale Schichtbilder, die zu 3D-Aufnahmen zusammengesetzt werden. Die in diesem Abschnitt eingeführten Bezeichnungen und Definitionen werden zunächst für 2D-Bilder gegeben. Die Definitionen für 3D-Bilder werden nur dann angeführt, wenn die Erweiterung für drei Dimensionen nicht unmittelbar ersichtlich ist.

Ein Bild I wird als Funktion betrachtet, welche jedem Bildpunkt $p \in I_D \subset \mathbb{N}^2$ einen Farbwert $c \in C_D = \{0, 1, ..., m\}$, für $m \in \mathbb{N}$, zuordnet:

$$I : I_D \to C_D : p \mapsto I(p). \tag{2.1}$$

Der Bildbereich (*image domain*) I_D kann als zweidimensionales Koordinatensystem dargestellt werden. Jedes Bildelement p ist somit über seine Koordinaten (p_x, p_y) eindeutig bestimmt. Abbildung 2.1 zeigt die Orientierung dieses Koordinatensystems. Der Unterschied zu den bekannten kartesischen Koordinaten liegt in der Lage der Y-Achse. Diese ist von oben nach unten orientiert.

Davon ausgehend, können wir den ABSTAND zweier Bildpunkte und die NACHBARSCHAFT eines Bildpunktes definieren.

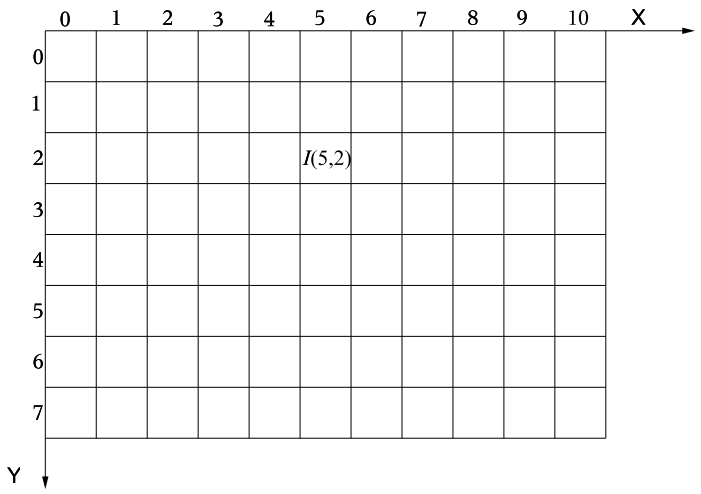


Abbildung 2.1: Darstellung der Orientierung des Bildkoordinatensystems. Im Gegensatz zur üblichen Orientierung zeigt in der Bildverarbeitung die Y-Achse von oben nach unten.

Definition 2.1.1 *Seien $p,q \in I_D$ zwei Bildpunkte. Es lassen sich damit folgende Abstände bzw. Normen definieren:*

- NORM *der* KOMPONENTEN-BETRAGSSUMME *oder* EINSER-NORM:

$$\|p-q\|_1 := |p_x - q_x| + |p_y - q_y| . \tag{2.2}$$

- EUKLIDISCHE NORM *oder* ZWEIER-NORM*:*

$$\|p-q\|_2 := \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2} . \tag{2.3}$$

- MAXIMUMS-NORM *oder* UNENDLICH-NORM*:*

$$\|p-q\|_\infty := \max\left\{ |p_x - q_x| , |p_y - q_y| \right\} . \tag{2.4}$$

Abbildung 2.2 zeigt die Werte der Abstände für die verschiedenen Normen.

Mit Hilfe des Abstands zweier Bildpunkte läßt sich nun die Nachbarschaft eines Bildpunktes definieren.

Definition 2.1.2 *Seien $p,q \in I_D$ wieder zwei Bildpunkte.*

4	3	2	3	4
3	2	1	2	3
2	1	0	1	2
3	2	1	2	3
4	3	2	3	4

(a)

$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$
$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
2	1	0	1	2
$\sqrt{5}$	$\sqrt{2}$	1	$\sqrt{2}$	$\sqrt{5}$
$\sqrt{8}$	$\sqrt{5}$	2	$\sqrt{5}$	$\sqrt{8}$

(b)

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

(c)

Abbildung 2.2: Drei verschiedene Abstandsdefinitionen bzw. Normen: (a) Einser-Norm, (b) Zweier-Norm (euklidischer Abstand), (c) ∞-Norm (Maximums-Norm)

- 4ER-NACHBARSCHAFT*:*

$$N_4(p) := \{q \in I_D \setminus \{p\} \mid \|q-p\|_2 \leq 1\}. \tag{2.5}$$

- 8ER-NACHBARSCHAFT*:*

$$N_8(p) := \{q \in I_D \setminus \{p\} \mid \|q-p\|_\infty \leq 1\}. \tag{2.6}$$

Für 3D-Bilder sind drei weitere Nachbarschaftsdefinitionen gebräuchlich.

- 6ER-NACHBARSCHAFT*:*

$$N_6(p) := \{q \in I_D \setminus \{p\} \mid \|q-p\|_1 \leq 1\}. \tag{2.7}$$

- 18ER-NACHBARSCHAFT*:*

$$N_{18}(p) := \left\{q \in I_D \setminus \{p\} \mid \|q-p\|_2 \leq \sqrt{2}\right\}. \tag{2.8}$$

- 26ER-NACHBARSCHAFT*:*

$$N_{26}(p) := \left\{q \in I_D \setminus \{p\} \mid \|q-p\|_2 \leq \sqrt{3}\right\}. \tag{2.9}$$

In Abbildung 2.3 sind die einzelnen Nachbarschaftsbeziehungen in 3D dargestellt.

Bildverarbeitung kann als die Manipulation der Farbwerte eines Bildes betrachtet werden. Dies kann prinzipiell auf zwei Arten erfolgen: die Neuberechnung der Farbwerte geschieht entweder aufgrund einer (a) *globalen* Betrachtung des Bildes oder die Farbwerte werden durch (b) *lokale* Bildeigenschaften neu bestimmt. Eine globale Operation ist z. B. die sogenannte Hough-Transformation.

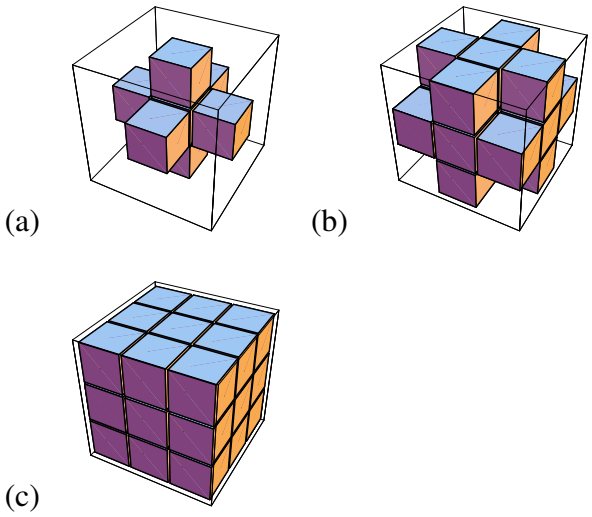


Abbildung 2.3: 3D-Nachbarschaften: Abbildung (a) zeigt die 6er-Nachbarschaft eines Bildpunktes p (p sitzt im Zentrum), (b) die 18er-Nachbarschaft und (c) die 26er-Nachbarschaft.

Diese wird zur Erkennung von parametrisierbaren geometrischen Objekten (Geraden, Kreise, Ellipsen usw.) eingesetzt. Lokale Operationen werden durch Anwenden von *Masken* auf das Bild umgesetzt. Dabei wird für die Neuberechnung des Farbwertes eines Bildpunktes nur die Nachbarschaft des Punktes berücksichtigt. Eine beliebte Größe für eine Maske M bildet die 8er-Nachbarschaft N_8.

Die Bearbeitung des Bildes durch einen *lokalen* Operator erfolgt nun so, daß das zentrale Feld der Maske über den zu bearbeiteten Bildpunkt gelegt wird. Der neue Farbwert des aktuellen Bildpunktes ergibt sich dann aus der gewichteten Summe der Farbwerte seiner Nachbarn. Die Gewichte werden dabei durch die Maske M festgelegt. Die allgemeine Form einer solchen Maske und die Art wie sie über das Bild gelegt wird, ist in Abbildung 2.4 zu sehen. Der neue Farbwert berechnet sich also wie folgt:

$$R = w_1 I(p_1) + w_2 I(p_2) + \ldots + w_9 I(p_9) = \sum_{i=1}^{9} w_i I(p_i), \tag{2.10}$$

wobei R und w_i beide in $\mathbb{R}$, sowie p_i jenem Bildpunkt entspricht, der durch die Maskenzelle w_i überdeckt wird. Der Wert R wird dann dem Bildpunkt unter der Zelle w_5, also p_5, zugewiesen. Auf diese Weise wird für jeden Bildpunkt ein neuer Farbwert berechnet. Der veränderte Farbwert wird dann in einer Kopie des Bildes

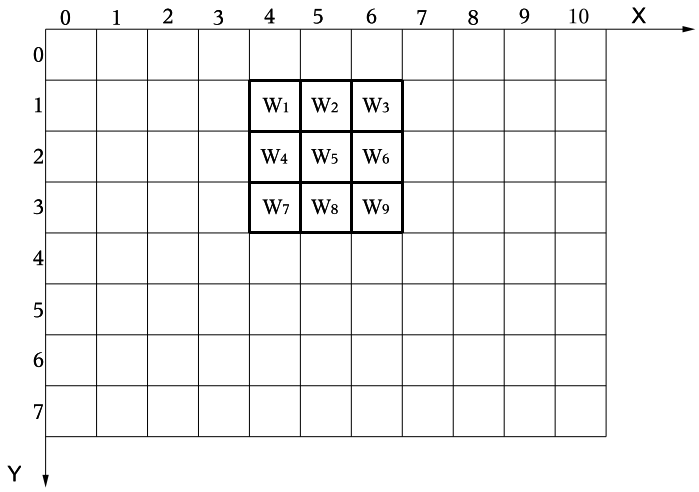


Abbildung 2.4: Allgemeine Form einer Maske M, wie sie über einen zu bearbeitenden Bildpunkt gelegt wird (in diesem Fall der Punkt $(5,2)$). Diese Maske berücksichtigt für die Neuberechnung eines Bildpunktes die 8er-Nachbarschaft des Bildpunktes.

gespeichert; ansonsten wäre das Ergebnis von der Reihenfolge, in der die Maske über das Bild geführt wird, abhängig.

2.1.1 Punkterkennung

Die Erkennung eines isolierten Punktes ist vergleichsweise einfach. Abbildung 2.5

-1	-1	-1
-1	8	-1
-1	-1	-1

Abbildung 2.5: Maske für die Erkennung eines isolierten Punktes auf einem Hintergrund mit konstanten Grauwerten.

zeigt die Maske M, welche für die Punkterkennung auf das Bild anzuwenden ist. Es werden alle jene Bildpunkte als *Punkte* erkannt, für die gilt:

$$|R| > T,$$

wobei $T \in \mathbb{R}^+$ ein benutzerdefinierter Schwellwert ist.

2.1.2 Linienerkennung

Die nächste Stufe der Segmentierung bildet die Erkennung von Linien. Wird die Maske (a) in Abbildung 2.6 auf das Bild angewendet, so werden Bereiche, die einer horizontalen Linie (mit einer Bildpunktbreite eins) entsprechen, hervorgehoben. Maske (b) und (d) verstärken Linien in einem Winkel von $\pm\frac{\pi}{4}$ und Maske (c) verstärkt vertikale Linien.

(a)

-1	-1	-1
2	2	2
-1	-1	-1

(b)

-1	-1	2
-1	2	-1
2	-1	-1

(c)

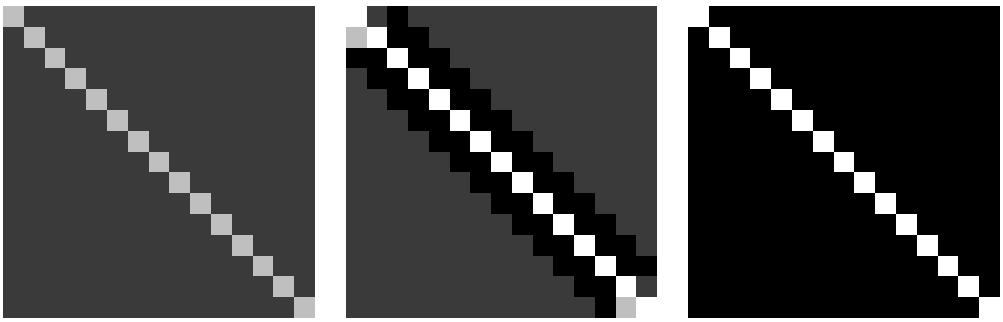
-1	2	-1
-1	2	-1
-1	2	-1

(d)

2	-1	-1
-1	2	-1
-1	-1	2

Abbildung 2.6: Diese Abbildung zeigt die Masken für die Linienerkennung. (a) horizontal, (b) $+\frac{\pi}{4}$, (c) vertikal, (d) $-\frac{\pi}{4}$.

Die Eigenschaften der beschriebenen Masken können sehr leicht mit Hilfe eines Beispiels überprüft werden. Abbildung 2.7 zeigt, wie die Grauwerte eines



(a) (b) (c)

Abbildung 2.7: Wendet man auf das Bild (a) die Maske in Abbildung 2.6 (d) an, so ergibt sich das Bild (b). Bild (c) ist schließlich durch Anwenden eines Schwellwertfilters auf Bild (b) entstanden. Man sieht, daß die hellgraue schräge Linie weiß wird und der dunkelgraue Hintergrund schwarz.

Bildes, die eine um $-\frac{\pi}{4}$ gedrehte Linie zeigt, durch Anwendung der Maske in Abbildung 2.6 (d) verändert werden. Wendet man auf das bearbeitete Bild noch einen Schwellwert an, so werden nur solche Punkte gezeichnet, welche auf der Linie liegen. Eine deutliche Hervorhebung der Linie wird sichtbar.

2.1.3 Kantenerkennung

Kanten eines Bildes sind dadurch charakterisiert, daß sie einen relativ großen Unterschied der Grauwerte zu ihrer Umgebung aufweisen. Bildet man nun den Gradienten des Bildes, so ist die Norm des Gradienten genau in jenen Gebieten hoch, welche eine starke Grauwertänderung aufweisen. Der Gradient eines 2D-Bildes ist ein zweidimensionaler Vektor, bestehend aus den partiellen Ableitungen:

$$\nabla g = \begin{pmatrix} G_x \\ G_y \end{pmatrix} = \begin{pmatrix} \frac{\partial I}{\partial p_x} \\ \frac{\partial I}{\partial p_y} \end{pmatrix}. \tag{2.11}$$

Die Norm des Gradienten wird als Magnitude bezeichnen und ist wie folgt definiert:

$$\nabla I = mag(\nabla g) := \sqrt{G_x^2 + G_y^2}. \tag{2.12}$$

Die Magnitude entspricht der Grauwertzunahme pro Bildpunkt in Richtung der maximalen Steigung ∇g. Die Berechnung der Magnitude ist jedoch relativ teuer, deshalb wird die Magnitude meist durch die Summe der Absolutbeträge der

partiellen Ableitungen angenähert:

$$\nabla I \approx |G_x| + |G_y| . \tag{2.13}$$

Diese Näherung von ∇I läßt sich sehr leicht durch Betrachtung der 8er-Nachbarschaft eines Bildpunktes berechnen. Sei

p_1	p_2	p_3
p_4	p_5	p_6
p_7	p_8	p_9

die 8er-Nachbarschaft des Bildpunktes p_5. Eine Näherung der Beträge der partiellen Ableitung berechnet sich dann folgendermaßen:

$$|G_x| \approx |(I(p_7)+I(p_8)+I(p_9)) - (I(p_1)+I(p_2)+I(p_3))| , \tag{2.14}$$

$$|G_y| \approx |(I(p_3)+I(p_6)+I(p_9)) - (I(p_1)+I(p_4)+I(p_7))| . \tag{2.15}$$

Die Gleichungen für G_x und G_y lassen sich nun aber auf die Anwendung von jeweils einer Maske zurückführen. Die benötigten Masken werden in Abbildung 2.8 gezeigt und werden auch als PREWITT-OPERATOREN bezeichnet. Die Prewitt-

$|G_x|$:

-1	-1	-1
0	0	0
1	1	1

$|G_y|$:

-1	0	1
-1	0	1
-1	0	1

Abbildung 2.8: Diese Abbildung zeigt die Masken für die Berechnung der Beträge der partiellen Ableitungen G_x und G_y. Diese beiden Masken werden auch als PREWITT-OPERATOREN bezeichnet

Operatoren haben den Nachteil, daß sie nicht nur die Kanten hervorheben, sondern auch das vorhandene Bildrauschen verstärken. Um das Rauschen des Bildes zu vermindern, muß es vorher geglättet werden. Deswegen wird zur Kantenerkennung auch meist nicht der Prewitt-, sondern der SOBEL-OPERATOR verwendet, da durch diesen das Bild gleichzeitig geglättet wird. Abbildung 2.9 zeigt die beiden Masken des Sobel-Operators.

2.2 Kantenorientierte Verfahren

Kantenorientierte Segmentierverfahren verwenden Informationen, die von Operatoren zur Kantendetektion gewonnen wurden. Diese Operatoren erkennen Diskontinuitäten in der Helligkeit, Farbe und/oder Textur. Das Ergebnis einer Kantenerkennung kann meist noch nicht als fertige Segmentierung verwendet werden.

$|G_x|$:

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

$|G_y|$:

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

Abbildung 2.9: Diese Abbildung zeigt die Masken des SOBEL-OPERATORS.

In einer Nachbearbeitung werden gefundene Kanten zu Ketten zusammengesetzt. Erst diese Ketten werden dann als Grenzen von Bereichen und die Bereiche als Teile von Objekten (Klassen) interpretiert. Bei den kantenorientieren Verfahren unterscheidet man noch zwischen lokalen und globalen Methoden. Zu den lokalen Verfahren zählen die KANTENRELAXATION und die KANTENVERFOLGUNG und zu den globalen Verfahren zählt die HOUGH-TRANSFORMATION.

2.2.1 Kantenrelaxation

Um die Qualität der Kantenerkennung (Abschnitt 2.1.3) zu erhöhen, werden bei der Kantenrelaxation die Kanteneigenschaften der Nachbarbildpunkte mit berücksichtigt. Dabei wird eine Kante als zwischen den Bildpunkten liegend betrachtet. (Diese werden dann auch als *crack edges* bezeichnet, siehe Abbildung 2.10.)

Jede dieser Kanten wird anhand ihrer Beziehung zu ihren Nachbarkanten bewertet. Die Kantenstärke, bzw. die Wahrscheinlichkeit, daß diese Kante Teil einer Objektgrenze ist, steigt, wenn die Kante zwischen zwei starken Kanten liegt. Liegt eine starke Kante zwischen zwei schwachen, so verliert die starke Kante an Gewicht, da sie vermutlich kein Teil einer Objektgrenze ist.

Zum Finden der Objektgrenzen müssen alle möglichen Kantennachbarn von e betrachtet werden. Der Zweck der Kantenrelaxation ist die Konstruktion von durchgehenden Objektgrenzen, durch Vergleich der Kantenmuster, die wir in einer lokalen Nachbarschaft von e finden können. Abbildung 2.11 zeigt einige Kantenmuster mit dazugehörigem Kantentyp. Die Kantentypen haben dabei folgende Bedeutung:

- **0-0**: isolierte Kante, mit negativem Einfluß auf die Kantenstärke.
- **0-2**, **0-3**: totes Ende, mit negativem Einfluß auf die Kantenstärke.
- **0-1**: unbestimmt, mit schwach positivem oder keinem Einfluß auf die Kantenstärke.
- **1-1**: Fortsetzung einer Kante, mit stark positivem Einfluß auf die Kantenstärke.

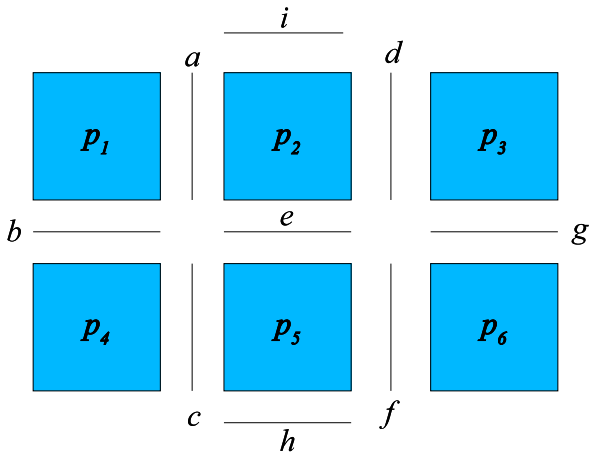


Abbildung 2.10: Diese Abbildung zeigt die zentrale Kante *e* (als *crack edge*) mit den drei möglichen Kantenfortsetzungen auf jeder Seite.

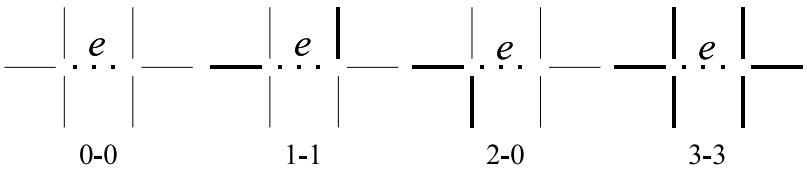


Abbildung 2.11: Diese Abbildung zeigt einige Kantenmuster mit dazugehörigem Kantentyp. Die Kantenstärke wird durch die Strichdicke repräsentiert.

- **1-2**, **1-3**: Fortsetzung zu einer Überschneidung, mit mittlerem positivem Einfluß auf die Kantenstärke.
- **2-2**, **2-3**, **3-3**: Brücke zwischen zwei Objektgrenzen, nicht notwendig für die Segmentierung und daher keinen Einfluß auf die Kantenstärke.

Die Methode der Kantenrelaxation ist ein iterativer Vorgang, wobei die Kantenstärke entweder gegen Eins (die Kante ist Teil einer Objektgrenze) oder gegen Null konvergiert.

Vor der ersten Iteration wird die Kantenstärke auf den normierten Wert der Magnitude des Bildes gesetzt. Die Normierung kann entweder durch das globale Maximum der Magnitude erfolgen oder durch ein lokales Magnitudenmaximum eines Bildausschnittes. Die zweite Methode hat den Vorteil, daß der Einfluß von sehr hohen, globalen Magnituden beschränkt bleibt. Algorithmus 1 zeigt den iterativen Ablauf der Kantenrelaxation.

Algorithm 1 Algorithmus der KANTENRELAXATION. Die Stärke einer Kante wird mit $c^{(i)}(e)$ bezeichnet. Das hochgestellte i in den runden Klammern kennzeichnet die i-te Iteration.

---

1. Initialisieren der Kantenstärke $c^{(1)}(e)$ für alle Kanten des Bildes.
2. Finden des Kantentyps jeder Kante anhand der Kantenstärke und seiner Nachbarschaft.
3. Aktualisieren der Kantenstärke $c^{(i+1)}(e)$ gemäß des Kantentyps und der vorherigen Kantenstärke $c^{(i)}(e)$.
4. Wiederholen von Schritte (2) und (3), bis alle Kantenstärken entweder gegen Eins oder gegen Null konvergiert sind.

Die beiden Hauptschritte aus Algorithmus 1 sind die Berechnung des Kantentyps (Schritt (2)), gefolgt von der Neuberechnung der Kantenstärke (Schritt (3)). Die Berechnung des Kantentyps der Kante e erfolgt durch die Bestimmung der Knotenarten der beiden Knoten von e. Die Knotenart (oder der Knotentyp) $t(v) \in \{0,1,2,3\}$ eines Knotens v ist wie folgt definiert:

$$t(v) := \text{argmax}_{k\in\{0,1,2,3\}} u(k), \tag{2.16}$$

wobei

$$u(k) := \begin{cases} (m-a)(m-b)(m-c) & \text{wenn} \quad k=0, \\ a(m-b)(m-c) & \text{wenn} \quad k=1, \\ ab(m-c) & \text{wenn} \quad k=2, \\ abc & \text{wenn} \quad k=3. \end{cases} \tag{2.17}$$

Dabei sind a, b, c die normierten Kantenstärken der angrenzenden Kanten des Knotens v (siehe Abbildung 2.10) und

$$m = \max\{a, b, c\}. \tag{2.18}$$

Der passende Kantentyp ist durch die Verkettung der beiden Knotentypen bestimmt. Die Berechnung der Kantenstärke (Schritt (3) von Algorithmus 1) erfolgt über die beiden Gleichungen (2.19) und (2.20):

$$c^{(i+1)}(e) := \min\left\{1, c^{(i)}(e) + \delta\right\}, \tag{2.19}$$

$$c^{(i+1)}(e) := \max\left\{0, c^{(i)}(e) - \delta\right\}, \tag{2.20}$$

mit $\delta < 0$. Gleichung (2.19) wird bei Erhöhung und Gleichung (2.20) wird bei Reduzierung der Kantenstärke verwendet.

Bereits nach wenigen Iterationen verbessert sich das Kantenbild erheblich. Iteriert man jedoch weiter, so verschlechtert sich das Ergebnis wieder. Um dies zu vermeiden, muß Schritt (3) des Algorithmus erweitert werden:

$$\text{wenn } c^{(i+1)}(e) > T_1 \quad \text{dann } c^{(i+1)}(e) := 1, \tag{2.21}$$

$$\text{wenn } c^{(i+1)}(e) < T_2 \quad \text{dann } c^{(i+1)}(e) := 0, \tag{2.22}$$

wobei T_1 und T_2 benutzerdefinierte Parameter sind, die die Konvergenzgeschwindigkeit und Genauigkeit der Relaxation beeinflussen.

2.2.2 Hough-Transformation

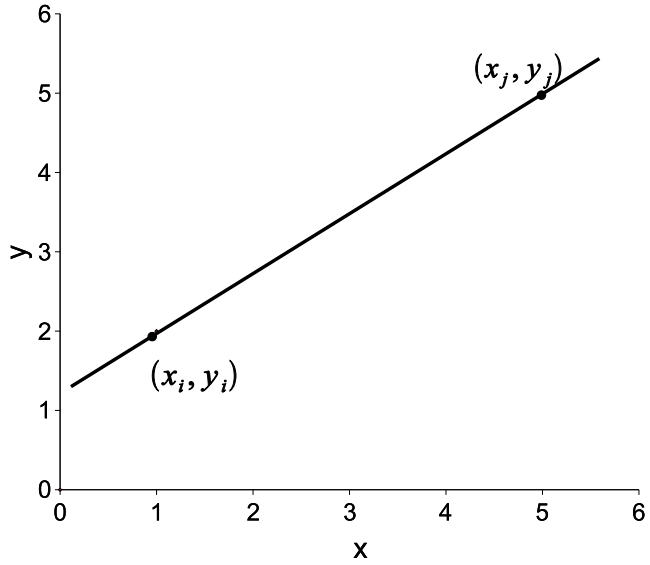
Die Hough-Transformation dient zur Erkennung von geometrischen Objekten in digitalen Bildern. Bei diesen Objekten handelt es sich meist um Geraden bzw. Geradensegmente. Für die Erkennung von Geraden betrachten wir den Punkt (x_i, y_i) und die dazugehörige Geradengleichung

$$y_i = a x_i + b. \tag{2.23}$$

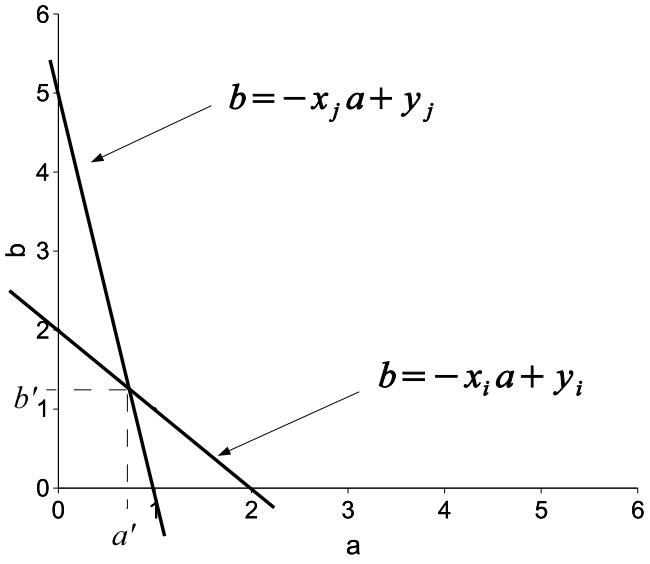
Es existieren unendlich viele Geraden, die den Punkt (x_i, y_i) schneiden und die Geradengleichung in der gegebenen Form erfüllen — für verschiedene Werte der Parameter a und b. Wir können die Gleichung (2.23) auch folgendermaßen schreiben:

$$b = -x_i a + y_i. \tag{2.24}$$

Ein Punkt im Bildraum (xy-Ebene) läßt sich im Parameterraum (ab-Ebene) durch eine Gerade darstellen. Fügt man einen zweiten Punkt $\left(x_j, y_j\right)$ hinzu, so erzeugt



(a)



(b)

Abbildung 2.12: Diese Abbildung zeigt die Hough-Transformation vom Bildraum der *xy*-Ebene (a) in den Parameterraum der *ab*-Ebene (b). Den beiden Punkten $(x_i, y_i) = (1,2)$ und $(x_j, y_j) = (5,5)$ im Bildraum entsprechen die beiden Geraden $g_1 : b = -a+2$ und $g_2 : b = -5a+5$ im Parameterraum. Aus dem Schnittpunkt der Geraden g_1 und g_2 lassen sich die Parameter a' und b' für die Gerade $\overline{(1,2)(5,5)}$ ablesen ($a' = \frac{3}{4}$ und $b' = \frac{5}{4}$).

auch dieser eine Gerade im Parameterraum. Der Schnittpunkt dieser beiden Geraden im Parameterraum kennzeichnet die Parameter der Geraden $\overline{(x_i, y_i)\,(x_j, y_j)}$ im Bildraum. In Abbildung 2.12 ist die Transformation von zwei Punkten der *xy*-Ebene im Bildraum in die zwei Geraden der *ab*-Ebene des Parameterraums dargestellt. Diese Transformation (des Bildraumes in den Parameterraum) wird als HOUGH-TRANSFORMATION bezeichnet. Führt man diese Transformation für alle Punkte des Bildbereichs durch, von denen man vermutet, daß sie auf einer Geraden liegen, so braucht man im Parameterraum nur noch nach den Schnittpunkten der Geraden suchen und die Geradenparameter (der Geraden im Bildraum) ablesen.

Die parametrisierte Darstellung einer Geraden, wie sie in Gleichung (2.23) gegeben ist, ist jedoch äußerst ungünstig, wenn wir es mit Geraden zu tun haben, die parallel oder beinahe parallel zur *y*-Achse verlaufen. In diesem Fall geht der Parameter a gegen unendlich. Aus diesem Grund wechseln wir zu folgender Darstellung:

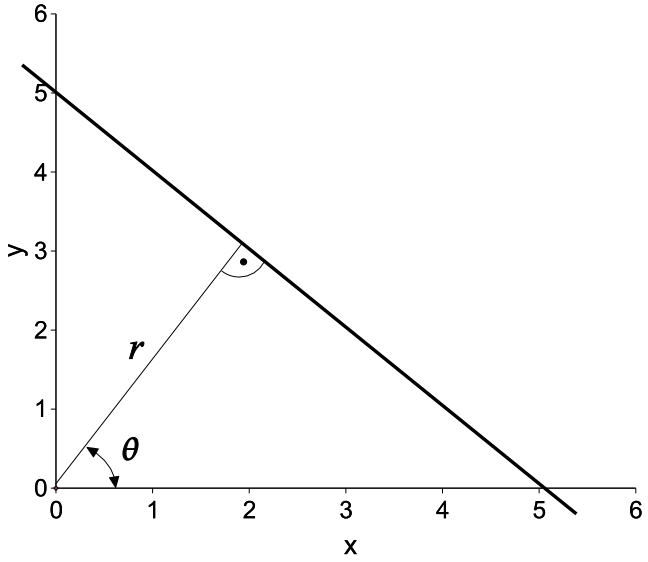
$$r = x\cos\theta + y\sin\theta, \tag{2.25}$$

wobei $\theta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$. Durch diese Einschränkung sind die Parameter der Geraden eindeutig bestimmt. Abbildung 2.13 zeigt die Hough-Transformation bei der Verwendung von Gleichung (2.25) als Parametrisierung einer Geraden im Bildbereich.

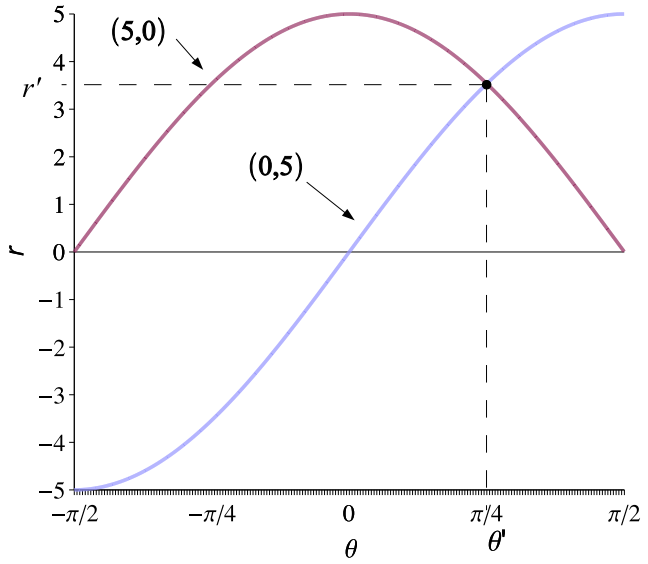
Nachdem wir nun wissen, wie man Punkte aus dem Bildraum in den Parameterraum transformiert, stellt sich nun die Frage nach der Berechnung der Parameter. Dazu nehmen wir eine Diskretisierung des Parameterraums vor, d. h. der Parameterraum wird in gleichmäßige, rechteckige Zellen unterteilt. Diese Zellen werden AKKUMULATOR-ZELLEN genannt. Für jeden Punkt aus dem Bildbereich, von dem wir vermuten, daß er auf einer Geraden liegt, berechnen wir die „diskretisierte" Sinuskurve. Dazu setzen wir diesen Punkt in die Gleichung (2.25) ein und berechnen den Wert für r für jeden diskreten Winkel θ_i. Das Ergebnis von r wird auf den nächsten erlaubten Wert r_i gerundet. Danach wird jene Akkumulator-Zelle, mit den berechneten Koordinaten (θ_i, r_i), um eins erhöht. (Die Akkumulator-Zellen werden vor Beginn der Rechnung auf Null gesetzt.) Am Ende brauchen wir nur mehr jene Parameter auszuwählen, deren Akkumulator-Zellen die größten Werte aufweisen. Je höher der Wert der Zelle, desto wahrscheinlicher ist eine Gerade mit diesem Parameter im Bildraum.

Bisher haben wir die Hough-Transformation dazu verwendet, um Geraden in einem Bild festzustellen. Die beschriebene Vorgehensweise läßt sich aber auf alle parametrisierbaren Objekte ausdehnen. Wollen wir z. B. jene Punkte feststellen, die auf einem Kreis liegen, so werden wir für einen Kreis folgende Parametrisierung verwenden:

$$c^2 = (x-a)^2 + (y-b)^2\,. \tag{2.26}$$



(a)



(b)

Abbildung 2.13: Diese Abbildung zeigt die Hough-Transformation mit der θr-Ebene als Parameterraum. Wird die Gerade, wie in Gleichung (2.25), parametrisiert, so entsprechen Punkte im Bildraum, nach der Transformation, Sinuskurven im Parameterraum. In (b) sind die zwei Sinuse für die Punkte $(0,5)$ und $(5,0)$ eingezeichnet. Daraus lassen sich die Parameter $(\theta, r) = \left(\frac{\pi}{4}, \sqrt{12.5}\right)$ ablesen.

Diese Parametrisierung induziert einen dreidimensionalen Parameterraum mit Akkumulator-Zellen in Quaderform. Hier wird für alle diskreten Parameter a und b und für alle Punkte aus dem Bildbereich der Parameter c berechnet und jene Zelle, mit den sich daraus ergebenden Koordinaten, inkrementiert. Auch hier sind große Werte in den Akkumulator-Zellen ein starkes Indiz für einen korrespondierenden Kreis (mit entsprechenden Parametern) im Bildraum.

2.3 Bereichsorientierte Verfahren

Während kantenbasierte Segmentierverfahren Regionen indirekt über ihre Begrenzung detektieren, versuchen bereichsorientierte Verfahren diese direkt zu finden. Bereichsorientierte Verfahren funktionieren üblicherweise besser in verrauschten Bildern, wenn Kanten schwierig zu finden sind. Die Ähnlichkeit oder Homogenität eines Bereichs wird bei diesen Verfahren als Hauptkriterium für die Segmentierung genutzt.

2.3.1 Schwellwertverfahren

Schwellwertverfahren (*thresholding*) sind einfache, aber sehr wichtige Methoden in der Bildsegmentierung. Die Zuordnung eines Bildpunktes zu einem Objekt erfolgt aufgrund seines Grauwertes. Daraus resultiert auch die Einfachheit und Schnelligkeit des Verfahrens. Nachteil dieses Verfahrens ist, daß die räumlichen Zusammenhänge der Bildpunkte vernachlässigt werden.

Das Verfahren läuft nun so ab, daß ein Schwellwert $T \in \mathbb{R}$ und eine Entscheidungsfunktion $e : I_D \rightarrow \Omega$ definiert wird, wobei Ω die Menge der Klassen bzw. Objekte ist, in die das Bild aufgeteilt werden soll. Die Entscheidungsfunktion für zwei Klassen ω_1 und ω_2 (Hintergrund und Objekt) sieht nun folgendermaßen aus:

$$e(c) := \begin{cases} \omega_1 & \text{wenn } c \leq T, \\ \omega_2 & \text{wenn } c > T. \end{cases} \tag{2.27}$$

Globale Schwellwerte führen nur eingeschränkt zu einem guten Segmentierergebnis. Lokale Schwellen können in Teilbereichen eines Bildes angewandt werden. Sogenannte adaptive Schwellwertverfahren benutzen eine Funktion, die einen lokalen Schwellwert aus den lokalen Eigenschaften des Bildes ableitet. Damit können Helligkeitsunterschiede in den einzelnen Bildbereichen ausgeglichen werden.

In Abbildung 2.14 ist das Grauwert-Histogramm eines MR-Bildes eines Gehirns zu sehen. Man wird beim Betrachten dieses Histogramms vermuten, daß ein einziger Schwellwert nicht in der Lage ist, das Bild „optimal" zu segmentieren.

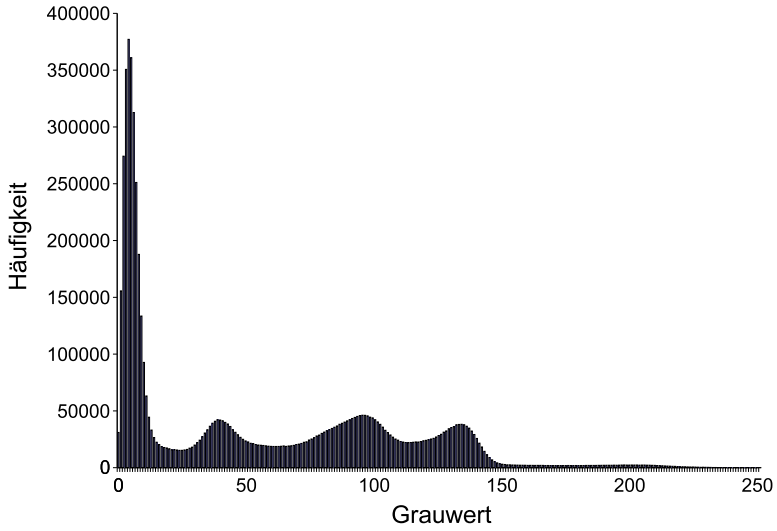


Abbildung 2.14: Diese Abbildung zeigt das Grauwert-Histogramm einer MR-Aufnahme eines Gehirns, mit Abmessungen von $181 \times 217 \times 181$.

Für mehrere Schwellwerte $T_1, T_2, ..., T_{k-1}$ läßt sich die Entscheidungsfunktion wie folgt erweitern:

$$e(c) := \begin{cases} \omega_1 & \text{wenn} \quad 0 \leq c \leq T_1, \\ \omega_2 & \text{wenn} \quad T_1 < c \leq T_2, \\ \vdots & \\ \omega_k & \text{wenn} \quad T_{k-1} < c. \end{cases} \tag{2.28}$$

Das Ergebnis eines Multi-Schwellwertverfahrens ist allerdings kein Binärbild mehr.

Neben der Wahl der richtigen Anzahl der Schwellwerte stellt sich auch die Frage nach deren Lage. Die Frage nach der Lage des optimalen Schwellwertes wollen wir für den Zwei-Klassen-Fall beantworten. Wir nehmen an, daß das Histogramm einer Mischverteilung mit zwei Verteilungsdichten gehorcht. (Vergleiche dazu Abschnitt 4.3.3.) Die Verteilungsdichte des Histogramms läßt sich also wie folgt beschreiben:

$$p(c) = p(\omega_1)\, p(c \mid \omega_1) + p(\omega_2)\, p(c \mid \omega_2), \tag{2.29}$$

wobei $p(\omega)$ die Wahrscheinlichkeit für das Auftreten der Klasse ω und $p(c \mid \omega)$ die Verteilungsdichte dieser Klasse bezeichnet. Setzt man für die klassenbeding-

ten Verteilungsdichten die Normalverteilung ein, so erhält man:

$$\begin{aligned} p(c) &= \frac{p(\omega_1)}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(c-\mu_1)^2}{2\sigma^2}\right] + \\ &\quad \frac{p(\omega_2)}{\sqrt{2\pi\sigma^2}} \exp\left[-\frac{(c-\mu_2)^2}{2\sigma^2}\right]. \end{aligned}$$

Der optimale Schwellwert ist jener, bei dem die Wahrscheinlichkeit einer Fehlklassifikation minimal ist. Sei

$$E_1(T) = \int_{-\infty}^{T} p(c \mid \omega_2)\, dc \tag{2.30}$$

der Erwartungswert der Fehlklassifikationswahrscheinlichkeit eines Hintergrundpunktes und

$$E_2(T) = \int_{T}^{\infty} p(c \mid \omega_1)\, dc \tag{2.31}$$

der Erwartungswert der Fehlklassifikationswahrscheinlichkeit eines Objektpunktes. Die Gesamtfehlklassifikationswahrscheinlichkeit ist somit

$$E(T) = p(\omega_2) E_1(T) + p(\omega_1) E_2(T). \tag{2.32}$$

Nach einigen Umformungen erhalten wir folgende quadratische Gleichung:

$$AT^2 + BT + C = 0, \tag{2.33}$$

mit

$$\begin{aligned} A &= \sigma_1^2 - \sigma_2^2, \\ B &= 2\left(\mu_1\sigma_2^2 - \mu_2\sigma_1^2\right), \\ C &= \sigma_1^2\mu_2^2 - \sigma_2^2\mu_1^2 + 2\sigma_1^2\sigma_2^2 \ln\left(\frac{\sigma_2 p(\omega_1)}{\sigma_1 p(\omega_2)}\right). \end{aligned}$$

Für gleiche Varianzen $\sigma^2 = \sigma_1^2 = \sigma_2^2$ ist ein einziger Schwellwert ausreichend:

$$T = \frac{\mu_1 + \mu_2}{2} + \frac{\sigma^2}{\mu_1 - \mu_2} \ln\left(\frac{p(\omega_2)}{p(\omega_1)}\right). \tag{2.34}$$

Die Bestimmung der Parameter der Verteilungsfunktionen kann über eine Minimierung des quadratischen Fehlers zwischen dem gemischten Modell und dem Histogramm erfolgen [7]:

$$\text{err} := \sum_{i=0}^{n} \left(p(c) - h(c)\right)^2, \tag{2.35}$$

wobei die Funktion $h(c)$ die relative Häufigkeit des Grauwertes c liefert und n dem maximalen Grauwert entspricht (siehe Abschnitt 4.3.3).

2.3.2 Bereichswachstumsverfahren

Das Bereichswachstumsverfahren (*region growing*) startet an einem Saatpunkt — dieser Saatpunkt wird dem zu segmentierenden Bereich zugeordnet — und überprüft die Nachbarpunkte auf Ähnlichkeit mit diesem Punkt. Liegt die Ähnlichkeit innerhalb eines Toleranzbereichs, werden auch diese Punkte dem Bereich zugeordnet und deren Nachbarn ebenfalls auf Ähnlichkeit untersucht (siehe Algorithmus 2). Die häufigsten Kriterien, die für das Ähnlichkeitsmaß herangezogen werden, sind:

- Grauwert der Bildpunkte,
- Farbe und Textur,
- Form des Bereichs.

Algorithm 2 Dieser Algorithmus zeigt den prinzipiellen Ablauf des Bereichswachstumsverfahren. Im ersten Schritt wird der Stack S mit dem Saatpunkt p_0 initialisiert. Die Funktion $h : I_D \times I_D \rightarrow \mathbb{R} : (p,q) \mapsto h(p,q)$ bildet das Ähnlichkeitsmaß zwischen zwei Punkten und $T \in \mathbb{R}$ ist ein benutzerdefinierter Schwellwert.

---

1. S
2. Solange $S \neq \emptyset$
   (a) $p \leftarrow S.\text{pop}$
   (b) $R \leftarrow R \cup \{p\}$
   (c) Für alle $p_j \in N_8(p)$
      i. Wenn $p_j \notin R \wedge h(p_j, p) < T$
         A. $S.\text{push}(p_j)$

Der Vorteil dieses Verfahrens ist die leichte Implementierung und die Ausführungsgeschwindigkeit. Zu den Nachteilen zählt, daß das Segmentierergebnis stark von der Wahl des Saatpunktes abhängt. Probleme gibt es auch, wenn der gewählte Saatpunkt auf einer Kante, d. h. auf einer Bereichsgrenze liegt.

2.3.3 Bereichsverschmelzung

Das Verfahren der Bereichsverschmelzung (*region melting*) geht von der Idee aus, jeden Bildpunkt als Saatpunkt für einen Bereich zu verwenden. Benachbarte Be-

reiche können verschmolzen werden, wenn ihre Vereinigung ein Ähnlichkeitskriterium erfüllt. Bereiche werden z. B. durch ihre Intensitätsstatistik gekennzeichnet. Nach einem Vergleich der Statistiken wird eine Entscheidung über die Verschmelzung getroffen. Wenn ein Bereich mit keinem seiner Nachbarbereiche mehr verschmolzen werden kann, wird dieser als „endgültig" bezeichnet. Der Algorithmus terminiert, wenn alle Bereiche als „endgültig" gekennzeichnet sind.

Eine andere Methode verwendet die Kantenstärke als Kriterium für die Verschmelzung von Bereichen. Zwei Bereiche können verschmolzen werden, wenn die Kantenstärke zwischen ihnen schwach ist.

2.3.4 Wasserscheiden-Transformation

Bei der Wasserscheiden-Transformation wird die Helligkeit eines Bildpunktes als Erhebung interpretiert. Ein Grauwertbild kann somit als *Helligkeitsgebirge* gesehen werden (siehe Abbildung 2.15). „Wassertropfen", die auf dieses Gebirge fallen, streben in Richtung des größten Gefälles den lokalen Minima zu und bilden dort „Wasserbecken". Definiert man nun alle Punkte, von denen aus ein Tropfen in ein solches Becken läuft, als Einflußzone dieses Beckens, so ergibt sich die Wasserscheide als Trennlinie zwischen zwei Becken. Durch kontinuierliches Fluten des Gebirges verschmelzen die kleineren und niedriger liegenden Becken, bis schließlich nur noch ein einziger, großer See vorhanden ist. In Abbildung 2.15 wurde eine Schicht eines MR-Bildes eines Gehirns als ein solches Helligkeitsgebirge dargestellt.

Für die Wasserscheiden-Transformation gibt es unterschiedliche Definitionsmöglichkeiten. Wir wollen hier zunächst eine Definition für ein kontinuierliches Grauwertbild I geben [15]. Danach geben wir eine Wasserscheidendefinition mittels topographischen Abstandes. Die zweite Definition ermöglicht (für den diskreten Fall) eine algorithmische Behandlung des Problems .

2.3.4.1 Kontinuierliche Wasserscheide

Für den kontinuierlichen Fall wird angenommen, daß das Bild I auf dem Bereich I_D zweimal stetig differenzierbar ist. Der TOPOGRAPHISCHE ABSTAND T_I des Bildes I zwischen zwei Punkten p und q ist definiert als:

$$T_I(p,q) := \inf \int_0^1 \|\nabla I(v(s))\|\, ds, \tag{2.36}$$

wobei das Infimum über alle Pfade (glatten Kurven) $v : [0,1] \to I_D$ genommen wird, mit $v(0) = p$ und $v(1) = q$. Der topographische Abstand zwischen einem Punkt $p \in I_D$ und einem Bereich $A \subseteq I_D$ ist definiert durch:

$$T_I(p,A) : \quad = \quad \min_{a \in A} \{T_I(p,a)\}. \tag{2.37}$$

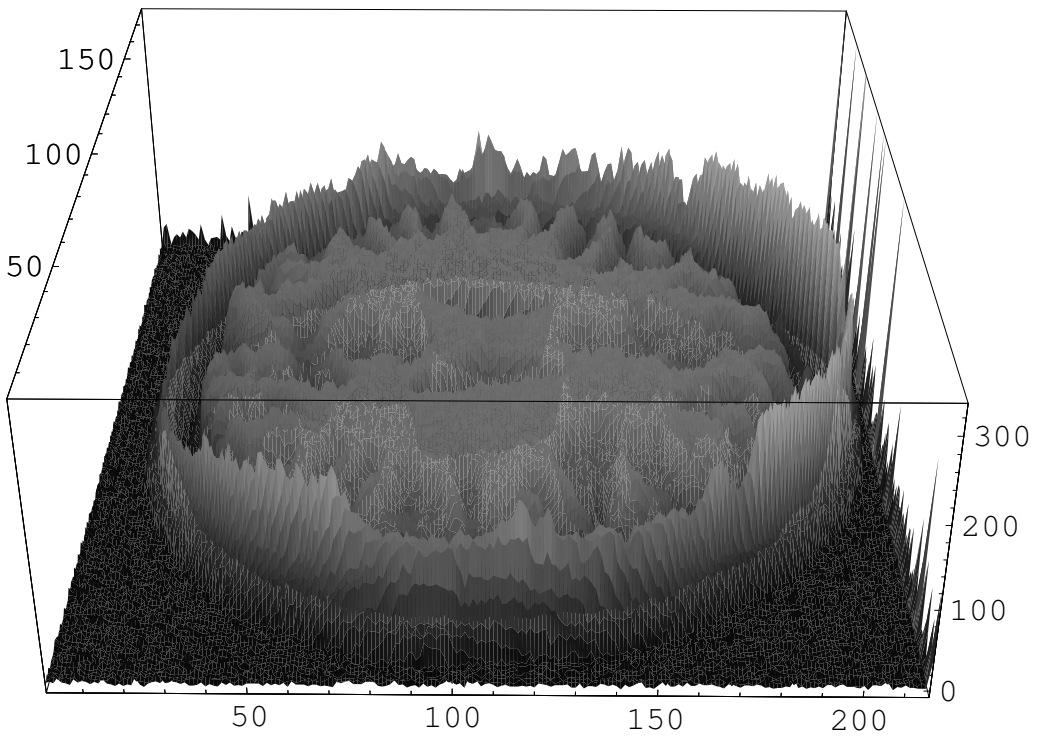


Abbildung 2.15: Diese Abbildung zeigt das zu einem Grauwertbild gehörige WASSERSCHEIDEN-GEBIRGE. Die Helligkeitswerte des Bildes — es handelt sich hierbei um ein Schichtbild einer MR-Aufnahme eines Gehirns — wurden als Höheninformation interpretiert.

Der Pfad mit dem geringsten topographischen Abstand zwischen zwei Punkten p und q wird auch als der PFAD DES GRÖSSTEN GEFÄLLES bezeichnet (*path of steepest descent*). Daraus leitet sich die Definition der Wasserscheidentransformation ab.

Definition 2.3.1 *Sei das Bild I zweimal stetig differenzierbar mit den lokalen Minima $\{m_i\}_{i\in\{1,\ldots,k\}}$. Das* AUFFANGBECKEN *(*catchment basin*) $CB(m_i)$ eines Minimums m_i ist definiert als die Menge von Punkten $x \in I_D$, deren topographischer Abstand zu m_i kleiner ist, als zu irgend einem anderen lokalen Minimum m_j.*

$$CB(m_i) := \left\{ x \in I_D \mid \forall j \in \{1,\ldots,k\} \setminus \{i\} : I(m_i) + T_I(x, m_i) < I\left(m_j\right) + T_I\left(x, m_j\right) \right\}. \tag{2.38}$$

Definition 2.3.2 *Die* WASSERSCHEIDE *W von I ist dann jene Menge von Punkten, welche keinem Auffangbecken zugeordnet sind.*

$$W(I) := I_D \cap \; comp \left(\bigcup_{i\in\{1,..,k\}} CB(m_i) \right). \tag{2.39}$$

Dabei bezeichnet $comp(X)$ die Komplementärmenge von X.

Definition 2.3.3 *Sei κ eine Markierung, $\kappa \notin \{1,\ldots,k\}$. Die* WASSERSCHEIDEN-TRANSFORMATION *ist eine Funktion $\lambda : I_D \to \{1,\ldots,k\} \cup \{\kappa\}$, die für jeden Bildpunkt p folgende Zuordnung trifft:*

$$\lambda(p) := \begin{cases} i & wenn\ p \in CB(m_i), \\ \kappa & wenn\ p \in W(I). \end{cases} \tag{2.40}$$

Die Wasserscheiden-Transformation eines Bildes I markiert jeden Bildpunkt, sodaß (a) unterschiedliche Auffangbecken eindeutig gekennzeichnet sind und (b) eine spezielle Markierung κ den Bildpunkten der Wasserscheide zugeordnet wird.

2.3.4.2 Wasserscheide mittels topographischem Abstand

Bei dieser Definition der Wasserscheiden-Transformation nehmen wir zunächst an, daß das Bild keine Plateaus aufweist, d. h. jeder Bildpunkt, der kein Minimum ist, besitzt mindestens einen Nachbarn mit einem echt kleinerem Grauwert — dies wird auch als *lower complete* Eigenschaft eines Bildes bezeichnet. Das Vorhandensein von Plateaus (Regionen mit konstantem Grauwert) stellt eine gewisse Schwierigkeit bei der Diskretisierung der topographischen Distanz (Gleichung (2.36) und (2.37)) dar. (Diese Annahme wird später wieder aufgeweicht.)

Definition 2.3.4 *Das* GERINGSTE GEFÄLLE *(*lower slope*) $LS(p)$ eines Bildes I für einen Bildpunkt p ist definiert als die maximale Neigung, welche p mit irgendeinem seiner Nachbarn mit geringerer Höhe verbindet:*

$$LS(p) := \max_{q \in N_*(p) \cup \{p\}} \left(\frac{I(p) - I(q)}{d(p,q)} \right), \tag{2.41}$$

wobei $d(p,q)$ den Abstand zwischen den Bildpunkten p und q bezeichnet. Für den Fall $p = q$ wird das Ergebnis der max*-Funktion als Null definiert.*

Wir definieren nun eine Kostenfunktion $c : \mathbb{R}^2 \times \mathbb{R}^2 \to \mathbb{R}$, welche die Kosten berechnet, die verursacht werden, wenn man vom Bildpunkt p zum Nachbarbildpunkt q wandert.

$$c(p,q) := \begin{cases} LS(p) \cdot d(p,q) & \text{wenn} \quad I(p) > I(q), \\ LS(q) \cdot d(p,q) & \text{wenn} \quad I(p) < I(q), \\ \frac{1}{2}(LS(p) + LS(q)) \cdot d(p,q) & \text{wenn} \quad I(p) = I(q). \end{cases} \tag{2.42}$$

Damit läßt sich nun der topographische Abstand entlang eines Pfades π angeben.

Definition 2.3.5 *Der* TOPOGRAPHISCHE ABSTAND *entlang eines Pfades $\pi := (p_0, \ldots p_l)$ zwischen den Punkten $p_0 = p$ und $p_l = q$ eines Bildes I ist definiert als:*

$$T_I^{\pi}(p,q) := \sum_{i=0}^{l-1} d(p_i, p_{i+1}) \cdot c(p_i, p_{i-1}). \tag{2.43}$$

Der topographische Abstand zwischen den Punkten p und q ist das Minimum der topographischen Abstände aller Pfade zwischen p und q:

$$T_I(p,q) := \min_{\pi \in [p \rightsquigarrow q]} T_I^{\pi}(p,q), \tag{2.44}$$

wobei die Menge aller Pfade π zwischen p und q mit $[p \rightsquigarrow q]$ bezeichnet wird.

Mit Hilfe von Definition (2.3.4) läßt sich nun eine neue Definition des Pfades des größten Gefälles geben.

Definition 2.3.6 *Ein Pfad $(p_0, \ldots, p_l)$ wird als* PFAD DES GRÖSSTEN GEFÄLLES *(*path of steepest descent*) von $p_0 = p$ zu $p_l = q$ bezeichnet, wenn für alle Punkte p_i gilt:*

$$p_{i+1} \quad \in \quad \Gamma(p_i), \tag{2.45}$$

wobei $\Gamma(p_i)$ die Menge der Nachbarbildpunkte bezeichnet, für die die Neigung $\frac{I(p)-I(q)}{d(p,q)} = LS(p)$ ist.

Weiters liegt ein Bildpunkt q „flußabwärts" (*downstream*) vom Bildpunkt p, wenn ein Pfad des gößten Gefälles von p nach q existiert. Ein Bildpunkt q liegt hingegen flußaufwärts (*upstream*) von p, wenn p flußabwärts von q liegt.

Theorem 2.3.1 *Sei $I(p) > I(q)$. Ein Pfad π von p nach q ist ein Pfad des größten Gefälles genau dann, wenn*

$$T_I^{\pi}(p,q) = I(p) - I(q). \tag{2.46}$$

Ist π kein Pfad des größten Gefälles, dann gilt: $T_I^{\pi}(p,q) > I(p) - I(q)$.

Aus Theorem 2.3.1 folgt, daß das Auffangbecken $CB(m_i)$ die Menge jener Punkte ist, die sich flußaufwärts vom Punkt m_i befinden. Die Definition der Auffangbecken (2.38) und der Wasserscheiden-Transformation (2.40) bleibt gleich.

Algorithmus 3 zeigt eine Wasserscheiden-Transformation mittels topographischen Abstand [3]. Der Algorithmus setzt voraus, daß die Bildpunkte p_i in einem *Array* der Länge n über einen Index i erreichbar sind, ohne daß das Wissen über die Nachbarschaftsbeziehung verloren geht. Im Schritt (1) des Algorithmus werden die Bildpunkte des Merkmalsbildes F auf den Index jenes Nachbarpunktes gesetzt, der Teil des Pfades des größten Gefälles ist. Hat ein Bildpunkt keinen niedriger liegenden Nachbarn, so ist dieser Teil eines Plateaus und wird entsprechend gekennzeichnet (der Wert von PLATEAU ist ein eindeutiger, ganzzahliger Wert, z. B. -1). Diese Plateaus entsprechen den lokalen Minima. Im zweiten Schritt wird für jedes Plateau ein eindeutiger Repräsentant bestimmt. Dazu wird die Funktion R benötigt.

$$R(p_i) := \begin{cases} i & \text{wenn} \quad F(p_i) = i, \\ R\left(p_{F(p_i)}\right) & \text{wenn} \quad F(p_i) \neq i. \end{cases} \tag{2.47}$$

Dabei werden nur Bildpunkte berücksichtigt, die bereits bearbeitet wurden (durch die Bedingung $j < i$). Im Schritt (3) werden alle Markierungen des Merkmalsbild durch ihre Repräsentanten ersetzt.

2.3.4.3 Das Plateau Problem

Probleme treten auf, wenn wir Algorithmus 3 direkt auf Bilder mit vorhandenen, nicht minimalen Plateaus anwenden — diese treten in wirklichen Bildern häufig auf. In diesem Fall ist der topographische Abstand innerhalb des Plateaus null. Aus diesem Grund muß eine zusätzliche Ordnungsrelation zwischen diesen Bildpunkten eingeführt werden [15].

Algorithm 3 Wasserscheiden-Transformationsalgorithmus für *lower complete* Bilder [3]. Die verwendete Funktion $R : I_D \rightarrow \{1,...n\}$ berechnet den Repräsentanten eines Bildpunktes p_i und ist folgendermaßen rekursiv definiert: $R(p_i) := i$, wenn $F(p_i) = i$ und $R\left(p_{F(p_i)}\right)$ sonst.

1. Für alle $p \in I_D$, $i \in \{1,...n\}$
   (a) Suche nach einem Bildpunkt $p_j \in N_*(p_i)$ mit $I\left(p_j\right) < I(p_i)$
   (b) Wenn $\exists p_j$
      i. $F(p_i) \leftarrow j$
   (c) Wenn $\neg\exists p_j$
      i. $F(p_i) \leftarrow$ PLATEAU
2. Für alle $p_i \in I_D$, $i \in \{1,...n\}$
   (a) Wenn $F(p_i) =$ PLATEAU
      i. $F(p_i) = i$
      ii. Für alle $p_j \in N_*(p_i)$ mit $\left(j < i \wedge I(p_i) = I\left(p_j\right)\right)$
         A. $r \leftarrow R(p_i)$
         B. $r' \leftarrow R\left(p_j\right)$
         C. $F(p_i) \leftarrow \min(r, r')$
         D. $F\left(p_j\right) \leftarrow \min(r, r')$
3. Für alle $p_i \in I_D$, $i \in \{1,...n\}$
   (a) $F(p_i) \leftarrow R(p_i)$

Definition 2.3.7 *Wir definieren die Funktion* $d : I_D \to \mathbb{N}$ *durch*

$$d(p) := \begin{cases} 0 & \text{wenn} \quad \prod_I^{\downarrow}(p) = \emptyset, \\ \min_{\pi \in \prod_I^{\downarrow}(p)} \{l(\pi)\} & \text{sonst}, \end{cases}$$

wobei $\prod_I^{\downarrow}(p)$ *die Menge aller absteigenden Pfade* π *ist, die im Punkt p beginnen und in irgendeinem Punkt q enden, mit* $I(p) > I(q)$*. Die Länge eines Pfades* π *wird mit* $l(\pi)$ *bezeichnet.*

Sei $L_c := \max_{p \in I_D} d(p)$*. Die* VERVOLLSTÄNDIGUNG *(*lower completion*)* I_{LC} *des Bildes I ist definiert durch*

$$I_{LC}(p) := \begin{cases} L_c \cdot I(p) & \text{wenn} \quad d(p) = 0, \\ L_c \cdot I(p) + d(p) - 1 & \text{sonst}. \end{cases}$$

Nach der Transformation des Bildes I in I_{LC} kann Algorithmus 3 darauf angewendet werden. Algorithmus 4 zeigt die *lower slope* Transformation mit Hilfe eines Stacks.

Algorithm 4 Algorithmus für die *lower slope* Transformation mit Hilfe eines Stacks S [15]. Mit N_* bezeichnen wir eine beliebige Nachbarschaft, z. B. N_4 oder N_8.

1. Für alle $p \in I_D$
   (a) $I_{LC}(p) \leftarrow 0$
   (b) Wenn $\exists q \in N_*(p)$ mit $I(q) < I(p)$
      i. $S.\text{push}(p)$
      ii. $I_{LC}(p) \leftarrow -1$
2. $d \leftarrow -1$
3. $S.\text{push}(p^*)$
4. Solange $S \neq \emptyset$
   (a) $p \leftarrow S.\text{pop}$
   (b) Wenn $p = p^*$
      i. Wenn $S \neq \emptyset$
         A. $S.\text{push}(p^*)$
         B. $d \leftarrow d + 1$
   (c) Wenn $p \neq p^*$
      i. $I_{LC}(p) \leftarrow d$
      ii. Für alle $q \in N_*(p)$ mit $(I(q) = I(p) \wedge I_{LC}(q) = 0)$
         A. $S.\text{push}(q)$
         B. $I_{LC}(q) \leftarrow -1$
5. Für alle $p \in I_D$
   (a) Wenn $I_{LC}(p) \neq 0$
      i. $I_{LC}(p) \leftarrow d \cdot I(p) + I_{LC}(p) - 1$
   (b) Wenn $I_{LC}(p) = 0$
      i. $I_{LC}(p) \leftarrow d \cdot I(p)$

Kapitel 3

Erweiterte Segmentierverfahren

3.1 Aktive Konturen

Eine wichtige Aufgabe der medizinischen Bildverarbeitung ist die Segmentierung von Grenzflächen zwischen Bildobjekten bzw. Gewebearten. Erschwert wird diese Segmentierung durch Unterbrechungen, Rauschen und andere Störeinflüsse an den Strukturgrenzen der zu segmentierenden Objekte. Aktive Konturen sind in der Lage, auch bei nicht optimalen Bedingungen, brauchbare Ergebnisse zu liefern.

3.1.1 *Snake*-Modell

Der am häufigsten eingesetzte Vertreter der aktiven Konturen ist das sogenannte *Snake*-Modell. Die *Snakes* zählen zu den kantenorientierten Segmentiermethoden, deren Aufgabe die Bestimmung von Begrenzungslinien von Bildobjekten ist.

Aktive Konturen können anschaulich als "Gummibänder" interpretiert werden, die sich im Lauf des Segmentiervorgangs immer mehr den Objektgrenzen anpassen. Geometrisch gesehen ist eine *Snake* eine parametrisierte Kurve im $\mathbb{R}^2$. Die Kontur wird durch die Funktion $\mathbf{v}(s) := \begin{pmatrix} x(s) \\ y(s) \end{pmatrix}$, mit $s \in [0,1]$, beschrieben. Für den diskreten Fall ist $\mathbf{v}$ durch $\mathbf{v} := \begin{pmatrix} x_i \\ y_i \end{pmatrix}$, mit $i \in \{0,...,N\}$, definiert. Da die Kontur geschlossen ist gilt weiters: $\begin{pmatrix} x(0) \\ y(0) \end{pmatrix} = \begin{pmatrix} x(1) \\ y(1) \end{pmatrix}$. Für das zu segmentierende Objekt wird nun eine Energiefunktion der Kontur definiert [12]:

$$E_S(\mathbf{v}) := \int_0^1 \left(\underbrace{E_i(\mathbf{v}(s))}_{\text{innere Energie}} + \underbrace{E_o(\mathbf{v}(s))}_{\text{äußere Energie}} \right) ds. \tag{3.1}$$

Ziel ist es nun eine Kontur $\mathbf{v}$ zu finden, sodaß die Energie E_S minimal wird. Die Energiefunktion setzt sich aus inneren (z. B. Krümmung) und äußeren Energien (Größe des Bildgradienten, Größe des Richtungsgradienten) zusammen — die Energien können auch als Kräfte interpretiert werden, die auf die Kontur wirken.

Die innere Energie E_i modelliert das Vorwissen über die Kontur und stellt somit ein implizites Konturmodell dar:

$$E_i(\mathbf{v}(s)) := \alpha(s)\left|\frac{d\mathbf{v}(s)}{ds}\right|^2 + \beta(s)\left|\frac{d^2\mathbf{v}(s)}{ds^2}\right|^2, \tag{3.2}$$

oder in diskreter Form:

$$E_i(\mathbf{v}_i) := \alpha_i\left|\frac{\mathbf{v}_i - \mathbf{v}_{i-1}}{h}\right|^2 + \beta_i\left|\frac{\mathbf{v}_{i-1} - 2\mathbf{v}_i + \mathbf{v}_{i+1}}{h^2}\right|^2, \tag{3.3}$$

mit $h := \frac{1}{N}$. Dabei werden die Konturpunkte mit $\mathbf{v}_i$ bezeichnet (siehe Abbildung 3.1). Der Term mit dem Gewicht α (erste Ableitung) definiert die Stetigkeit, Elastizität und die Geschlossenheit der Kontur. Der Term mit der Gewichtung β (zweite Ableitung) definiert die Glattheit, Biegsamkeit und die Krümmung; je höher der Wert für β, desto schlechter kann sich die *Snake* an stark gekrümmte Grenzen anpassen. Die Gewichtungsfaktoren α und β werden gewöhnlich über den gesamten Bereich von s konstant gehalten. Durch Veränderung der jeweiligen Werte von α und β während des Segmentiervorgangs bestimmt die innere Energie die Verformungseigenschaften der *Snake*. Je stärker die Kontur gekrümmt ist, desto größer ist die innere Energie der Kontur.

Bei der diskreten Approximation der inneren Energie wurde davon ausgegangen, daß der Abstand zwischen zwei Stützpunkten $\mathbf{v}_i$ und $\mathbf{v}_{i+1}$ immer gleich groß ist. Dies wird während des Optimierungsprozesses jedoch nicht gewährleistet. Aus diesem Grund wird der Stetigkeitsterm $\left|\frac{d\mathbf{v}(s)}{ds}\right|$ durch

$$\left|\frac{d\mathbf{v}(s)}{ds}\right| \approx \frac{d - |\mathbf{v}_i - \mathbf{v}_{i-1}|}{|\mathbf{v}_i - \mathbf{v}_{i-1}|} \tag{3.4}$$

(dabei ist d der mittlere Abstand zwischen den Stützpunkten der Kontur) und der Glattheitsterm $\frac{d^2\mathbf{v}(s)}{ds^2}$ durch

$$\frac{d^2\mathbf{v}(s)}{ds^2} \approx \frac{\mathbf{v}_{i+1} - \mathbf{v}}{|\mathbf{v}_{i+1} - \mathbf{v}|} - \frac{\mathbf{v}_i - \mathbf{v}_{i-1}}{|\mathbf{v}_i - \mathbf{v}_{i-1}|} \tag{3.5}$$

approximiert.

Die äußere Energie

$$E_o(\mathbf{v}(s)) \quad = \quad P(\mathbf{v}(s)) \tag{3.6}$$

wird durch eine Potentialfunktion $P(\mathbf{v}(s))$ beschrieben. Setzt man diese Funktion auf

$$P(\mathbf{v}(s)) := -\gamma|\nabla I(\mathbf{v}(s))|^2, \tag{3.7}$$

so wird dadurch die Kontur zu den Kanten des Objektes hingezogen. Die äußere Energie wird genau dann minimiert, wenn sich die Kontur an eine Kante anlegt. Kanten des Bildes I sind durch den Betrag des Gradienten $|\nabla I(\mathbf{v}(s))|$ gekennzeichnet. Der Gewichtungsfaktor γ legt dabei fest, wie stark die äußere Energie in die Berechnung der Gesamtenergie E_S einfließt.

Die Wechselwirkung der beiden Energien kommt zustande, indem die innere Energie eine Kraft in Richtung des Konturzentrums ausübt, wogegen die äußere Energie eine Kraft erzeugt, die vom Objektzentrum weg zeigt. Das Ziel der Optimierung ist es, eine aktive Kontur zu finden, bei der die Energie minimiert wird. Dies ist erreicht, wenn die beiden Energien im Gleichgewicht sind, d. h. der Betrag der inneren Kraft ist gleich dem Betrag der äußeren Kraft.

Zu Beginn des Optimierungsprozesses wird im digitalen Bild eine geschlossene Kontur durch manuelle Auswahl von Konturpunkten $\mathbf{v}_0, \ldots, \mathbf{v}_n$ definiert, wobei $\mathbf{v}_0 := \mathbf{v}_n$ (siehe Abbildung 3.1). Diese Punkte bilden, bei der iterativen Anpassung

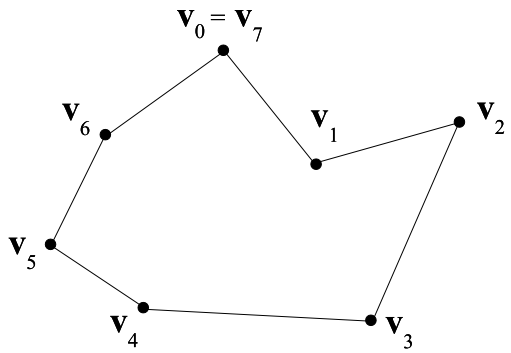


Abbildung 3.1: Geschlossene Kontur: In diesem Beispiel sind die ausgewählten Stützpunkte durch Polynome (ersten Grades) approximiert.

an die Objektgrenze, Stützstellen. Zwischen den Stützstellen wird, durch Polynome oder Splines, interpoliert. Die Komplexität der Bestimmung eines globalen Minimums ergibt sich dadurch, daß der Aufwand eines Suchalgorithmus mit der Anzahl der Konturpunkte exponentiell steigt; für eine aktive Kontur mit n Punkten in einem Bildbereich mit $N_x \cdot N_y$ Bildpunkten gibt es $(N_x \cdot N_y)^n$ mögliche Konturen.

Zur Reduktion der Komplexität (während eines Iterationsschrittes) wird die Suche nach besseren Konturpunkten auf die Nachbarschaft der aktuellen Konturpunkte eingeschränkt.

Diese Vorgehensweise wird durch den GREEDY-ALGORITHMUS umgesetzt. Der Algorithmus erzielt die Laufzeitreduktion dadurch, daß er in jedem Iterationsschritt die aktuelle Konturenergie mit der Konturenergie vergleicht, die entstehen würde, wenn man einen Konturpunkt durch einen seiner Nachbarpunkte austauscht (siehe Algorithmus 5). Ist die Energie, die sich bei der Wahl eines

Algorithm 5 *Greedy*-Algorithmus: Prüft für alle Bildpunkte der 8er--Nachbarschaft eines Konturpunktes, ob dadurch die Energie verringert wird.

1. Für alle $v \in \{v_1, \ldots, v_n\}$
   (a) $v \leftarrow \text{argmax}_{v' \in N_8(v)} \{E_S(v'), E_S(v)\}$
2. Wiederholen von Schritt 1 bis lokales Minimum erreicht ist; keiner der Konturpunkte hat sich geändert.

Nachbarbildpunktes ergibt, kleiner, als die des aktuellen Konturbildpunktes, so wird der Konturpunkt diesem Bildpunkt zugeordnet. Andernfalls bleibt die Position unverändert. Dies wird für alle Konturpunkte durchgeführt. Durch die auf die lokale Nachbarschaft eingeschränkte Betrachtung wird die Anzahl der Vergleiche in jedem Durchlauf auf $8 \cdot n$ reduziert (bei einer 8er-Nachbarschaft). Nachteil des *Greedy*-Algorithmus ist, daß er nicht immer eine optimale Lösung findet. In der Praxis sind die Ergebnisse jedoch fast immer ausreichend.

Konturmodelle sind sehr gut dazu geeignet, mit Artefakten behaftete, Bilder, wie sie in der Medizin im allgemeinen anzutreffen sind, zu segmentieren. Es ergeben sich aber auch einige Probleme. *Snakes* wurden für interaktive Anwendungen entwickelt; bei nicht interaktiver Anwendung ist die Vorgabe einer Startkontur in der Nähe der gesuchten Kontur notwendig. Ein weiteres Problem ist das Auffinden geeigneter Parameter und Gewichtsfaktoren, die in Abhängigkeit vom Bildmaterial und den Formeigenschaften der Bildobjekte stark variieren können. Die feste Parametrisierung der Kontur beeinflußt die Flexibilität, sodaß es für die *Snake* nicht möglich ist, z. B. röhrenartige Formen anzunehmen. Darüber hinaus muß die Topologie des Bildobjektes im voraus bekannt sein.

Um diese Unzulänglichkeiten zu beheben, wurden zahlreiche Verbesserungen des Verfahrens vorgeschlagen. Das *T-Snake*-Modell von McInerney und Terzopoulos [13] ist eine Weiterentwicklung des klassischen *Snake*-Modells, welches die genannten Probleme löst.

3.1.2 *T-Snake*-Modell

Das *T-Snake*-Modell (*topology adaptive snake model*) wird als geschlossene, zweidimensionale Kontur definiert, bestehend aus n Knoten (Modellknoten), verbunden durch n Kanten (auch Elemente genannt). Diese elastische Kontur ist eine diskrete Approximation des konventionellen *Snake*-Modells und behält ihre Eigenschaften bei, d. h. es agieren auch hier äußere und innere Kräfte.

Mit den Knoten der *T-Snake* werden zeitvariierende Positionen $\mathbf{x}_i(t) = \begin{pmatrix} x_i(t) \\ y_i(t) \end{pmatrix}$, mit $i \in \{1,...,n\}$, assoziiert.

Das Verhalten der *T-Snake* wird durch eine vereinfachte Form der Lagrange-Gleichung

$$\mu\frac{\partial^2\mathbf{v}}{\partial t^2} + \gamma\frac{\partial\mathbf{v}}{\partial t} - \frac{\partial}{\partial s}\left(\alpha\frac{\partial\mathbf{v}}{\partial s}\right) + \frac{\partial^2}{\partial s^2}\left(\beta\frac{\partial^2\mathbf{v}}{\partial s^2}\right) = -\nabla P(\mathbf{v}(s,t)) \tag{3.8}$$

dargestellt. Das Ergebnis ist eine Menge von gewöhnlichen Differentialgleichungen erster Ordnung:

$$\gamma_i\dot{\mathbf{x}}_i + a\alpha_i + b\beta_i = \rho_i + f_i, \tag{3.9}$$

wobei $\dot{\mathbf{x}}_i(t)$ die Geschwindigkeit von Knoten i und γ_i der Dämpfungsfaktor ist. Die Gleichung (3.9) wird auch als DISKRETE EULER-GLEICHUNG bezeichnet. Die internen Zugkräfte

$$\alpha_i(t) = 2x_i(t) - x_{i-1}(t) - x_{i-1}(t) \tag{3.10}$$

sind diskrete Approximationen der zweiten Ableitung der Lagrange-Gleichung nach s (dritter Term von Gleichung (3.8)). Sie dienen dazu, einen gleichmäßigen Abstand zwischen den Modellknoten zu gewährleisten. Der Parameter a kontrolliert dabei den Widerstand der Kontur gegen Streckungen. Die internen Biegekräfte

$$\beta_i(t) = 2\alpha_i(t) - \alpha_{i-1}(t) - \alpha_{i-1}(t) \tag{3.11}$$

sind diskrete Approximationen der vierten Ableitung der Lagrange-Gleichung nach s (vierter Term von Gleichung (3.8)). Der Parameter b kontrolliert dabei den Widerstand der Kontur gegen Verbiegung.

Die rechte Seite von Gleichung (3.9) bezeichnet die externen Kräfte ρ_i und f_i. Da das Modell trägheitslos ist, kommt es zur Ruhe, sobald die angreifenden, äußeren Kräfte gleich den internen Kräften sind. Um die Kontur in Richtung der Objektkanten zu drücken, wird die externe Kraft ρ_i (auch Inflationskraft genannt) auf

$$\rho_i(t) := qF_T(I(\mathbf{x}_i(t)))\,\mathbf{n}_i(t) \tag{3.12}$$

gesetzt, wobei $\mathbf{n}_i$ der normierte Normalvektor der Kontur am Knoten i ist und q die Amplitude der Kraft ρ_i bezeichnet. Die Funktion

$$F_T\left(I\left(x,y\right)\right) := \begin{cases} +1 & \text{wenn} \quad I\left(x,y\right) \geq T, \\ -1 & \text{sonst} \end{cases} \tag{3.13}$$

stellt eine Verbindung der Kraft mit dem Bild her; mit einem Schwellwert $T < 0$. Die Funktion F läßt die Kontur sich zusammenziehen, wenn $I\left(x,y\right) < T$. Sie verhindert auch, daß die Kontur in den Hintergrund verschwindet.

Damit die Kontur an signifikanten Kanten stoppt, wird die zweite, externe Kraft f_i auf

$$f_i\left(t\right) := p\nabla P\left(\mathbf{x}_i\left(t\right)\right) \tag{3.14}$$

gesetzt, mit

$$P\left(\mathbf{x}_i\left(t\right)\right) := -c\left\|I\left(\mathbf{x}_i\left(t\right)\right)\right\|, \tag{3.15}$$

wobei das Gewicht p (in Gleichung (3.14)) die Größe der Kraft bestimmt und c (in Gleichung (3.15)) ein zusätzlicher Gewichtungsfaktor der Bildintensität ist. Für gewöhnlich bewegen sich die beiden Faktoren p und q innerhalb der gleichen Größenordnung.

Wir integrieren nun die Gleichung (3.9) nach der Euler-Methode; dieses Verfahren approximiert die Ableitungen mit Hilfe finiter Differenzen. Das Verfahren aktualisiert die Positionen der Modellknoten vom Zeitpunkt t zum Zeitpunkt $t + \triangle t$ nach folgender Formel:

$$x_i^{(t+\triangle t)} = x^{(t)} - \frac{\triangle t}{\gamma}\left(a\alpha_i^{(t)} + b\beta_i^{(t)} - \rho_i^{(t)} - f_i^{(t)}\right). \tag{3.16}$$

Die Gleichung (3.16) ist leicht zu berechnen, sie verhält sich allerdings instabil bei kleinen Zeitschritten $\triangle t$.

Der Unterschied des *T-Snake* Modells zum klassischen *Snake*-Modell ist, daß die Menge der Knoten und Elemente (Kanten) während des Entwicklungsprozesses nicht konstant bleibt. Die Konstruktion der *T-Snakes* basiert auf der Simplexzerlegung des Bildraumes (siehe Abbildung 3.2). Diese bildet die zweite Komponente des *T-Snake* Modells — die Konturenergie bildet die erste Komponente.

Mit Hilfe des, durch die Simplexzerlegung entstandenen, Gitters läßt sich die innere Region von der äußeren Region der *T-Snake* eindeutig unterscheiden. Alle Simplexecken der inneren Region erhalten eine positive Markierung und alle Simplexecken der äußeren Region eine negative Markierung (Abbildung 3.3). Sind die Markierungen der Ecken eines Simplex alle gleich, kann es entweder nur ganz im Inneren oder ganz im Äußeren der anatomischen Struktur liegen. Simplizes mit verschiedenen Markierungen werden Grenzzellen genannt.

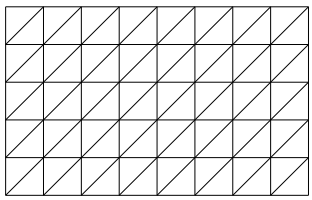


Abbildung 3.2: Darstellung der SIMPLEXZERLEGUNG des Bildraumes. Diese spezielle Zerlegung wird auch FREUDENTHAL-TRIANGULATION genannt.

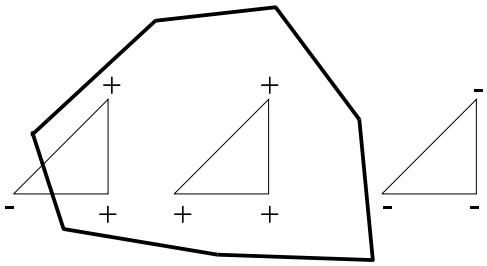


Abbildung 3.3: Diese Abbildung zeigt die SIMPLEXKLASSIFIKATION. Sind alle Ecken eines Simplex innerhalb des Bereichs, der durch die Kontur definiert wird, werden sie positiv gekennzeichnet, ansonsten negativ.

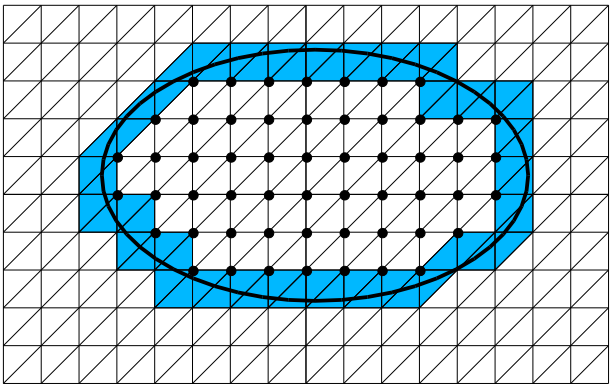


Abbildung 3.4: Diese Abbildung zeigt die Simplexapproximation einer Objektkontur (Ellipse) mittels einer Freudenthal-Triangulation. Die positiv gekennzeichneten Simplexecken sind durch Punkte gekennzeichnet. Die Modellknoten entsprechen den Schnittpunkten der Objektkontur mit den Simplexkanten. Die Grenzzellen wurden eingefärbt.

Nach $m \in \{5, ..., 10\}$ vom Benutzer definierten Zeitschritten[1], in denen sich die *T-Snake* aufgrund des Einflusses von internen und externen Kräften verformt hat (Gleichung (3.16)), werden die geometrischen Parameter der *T-Snake* neu bestimmt. Mit der Neudefinition der geometrischen Parameter — dies sind die Anzahl und Lage der Modellknoten — ist *ein* Iterationsschritt abgeschlossen. Zu Beginn des ersten Iterationsschrittes sind die *T-Snake* Knoten durch die Schnittpunkte mit den Kanten der Simplexzellen bestimmt. Nach dem Ende der m Verformungsschritte haben sich die Knoten relativ zu den Simplexkanten bewegt. Die geometrischen Parameter des *T-Snake* Modells werden nun unter Berücksichtigung der Simplexzerlegung neu definiert, indem eine neue Approximation für die verformte *T-Snake* berechnet wird. Diese Berechnung wird in zwei Phasen ausgeführt.

3.1.2.1 Phase I

Für jedes *T-Snake* Element wird getestet, ob es sich mit einer Simplexkante schneidet. Wird ein Schnittpunkt gefunden, wird dieser in einer Datenstruktur, zugehörig zu den Simplexkanten, gespeichert. Der gefundene Schnittpunkt ist ein potentiel-

[1] Dies ist ein empirischer Wert und wurde von [13] übernommen.

ler neuer Knoten der *T-Snake*. Wenn nach Phase II die beiden Ecken der geschnittenen Simplexkante positiv sind, wird der zugehörige Schnittpunkt verworfen, da beide Ecken im Inneren der *T-Snake* liegen und die dazwischen liegende Kante kein Grenzelement sein kann.

Weiterhin sollen in dieser Phase alle Simlexecken gefunden werden, die nach dem Verformungsschritt im Inneren der *T-Snake* liegen, obwohl sie davor in der äußeren Partition lagen. Da sie noch eine negative Markierung tragen, kann die korrekte Gestalt der *T-Snake* nicht mehr über die Markierung nachvollzogen werden. Daher wird für jeden Schnittpunkt das schneidende *T-Snake* Element betrachtet und sein Normalvektor benutzt, um festzustellen ob eine Ecke des zugehörigen Simplex in der inneren oder der äußeren Partition liegt (siehe Abbildung 3.5).

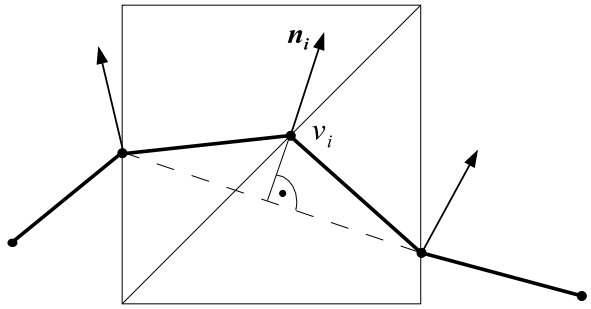


Abbildung 3.5: Dieses Beispiel zeigt, wie der Normalvektor genutzt wird, um das „Vorzeichen" einer Simplexecke festzustellen. Die Normalvektoren der Modellknoten zeigen nach außen.

Eine Ecke liegt in der äußeren Partition, wenn sie sich in positiver Richtung der Normalen befindet und in der inneren Partition, wenn sie sich in negativer Richtung der Normalen befindet. Alle so gefundenen potentiellen inneren Ecken werden in einer Liste gespeichert und in Phase II weiterbearbeitet.

Ein Spezialfall ist, wenn sich für eine Simplexkante mehrere Schnittpunkte finden lassen. Dies tritt auf, falls sich eine *T-Snake* mit sich selbst oder mit einer anderen *T-Snake* schneidet. In diesem Fall wird die niedriger numerierte Ecke der Simplexkante genommen. Dabei wird mit Hilfe der Normalen des *T-Snake* Elements bestimmt, auf welcher Seite des *T-Snake* Elements die Kante liegt. Somit trägt nun jeder Schnittpunkt ein Zeichen für innen oder außen. Nun wird jeder Schnittpunkt mit dem, unter Umständen noch existierenden, alten Schnittpunkt

aus der vorhergehenden Iteration verglichen. Haben sie gegensätzliche Markierungen, dann grenzen sie sich gegenseitig aus und die Simplexkante enthält in ihrer Datenstruktur keinen Schnittpunkt mehr. Besitzen sie dagegen dieselbe Markierung, dann ersetzt der neue Schnittpunkt den alten. Falls es jedoch keinen existierenden Schnittpunkt gab, dann wird der neue Schnittpunkt in der Datenstruktur, zugehörig zu der Simplexkante, gespeichert.

3.1.2.2 Phase II

Die Inflationskraft ρ_i, welche die Modellknoten in Richtung des Normalvektors drückt, kann Singularitäten und Selbstüberschneidungen zur Folge haben. In dieser Situation ist nicht klar, wie die *T-Snake* weiterentwickelt werden soll. Dieses Problem wird dadurch gelöst, indem eine sogenannte Entropiebedingung eingeführt wird: wenn eine Simplexecke einmal als positiv gekennzeichnet wurde, bleibt sie positiv.

In dieser Phase erhalten nun alle Simplexecken, die sich nach dem Verformungsschritt im Inneren der *T-Snake* befinden, eine positive Markierung. Dies geschieht dadurch, indem die Liste der potentiellen neuen, inneren Ecken aus Phase I durchlaufen werden. Die Datenstruktur, der zu ihnen gehörenden Simplexkanten, wird ausgewertet. Enthält sie einen Schnittpunkt, erhält dieser Knoten v_* eine positive Markierung. Die Nachbarn der Knoten v_* sind ebenfalls potentielle positive Knoten. Diese erhalten eine positive Markierung, wenn die Simplexkanten zu v_* keinen Schnittpunkt haben.

3.1.2.3 Algorithmus

Der *T-Snake* Algorithmus läuft folgendermaßen ab:

1. Verformungsschritt (Ausführung der m Zeitschritte):
   (a) Berechnen der externen und internen Kräfte, welche auf die *T-Snake* Knoten und Elemente wirken (Gleichung (3.10), (3.11), (3.12) und (3.14)).
   (b) Aktualisierung der Knotenpositionen durch die Euler-Methode (Gleichung (3.16)).
2. Neubestimmung der geometrischen Parameter (Knotenanzahl und Knotenposition).
   (a) Ausführen von Phase I.
   (b) Ausführen von Phase II.

3. Für jedes aktuelle *T-Snake* Element wird bestimmt, ob es noch gültig ist. Ein *T-Snake* Element heißt gültig, wenn sein zugehöriger Simplex noch eine Grenzzelle ist. Verwerfen ungültiger *T-Snake* Elemente und ungenutzter Schnittpunkte.

4. Approximation der Kontur:

   (a) Benutze die Simplexecke, welche in Phase II positive Markierungen erhalten haben, um die neuen Grenzzellen und somit auch die neuen *T-Snake* Knoten und Elemente zu bestimmen.

Eine *T-Snake* hat ihr Gleichgewicht erreicht, wenn alle Elemente für eine nutzerspezifische Anzahl von Verformungsschritten inaktiv waren. Die *T-Snake* Elementaktivität wird mittels Zuweisung einer Temperatur gemessen. Die Temperatur entspricht der Anzahl der Verformungsschritte, für welche das *T-Snake* Element und sein zugehöriger Simplex gültig geblieben sind. Fällt die Temperatur eines Elements unter einen bestimmten Wert, dann wird es inaktiv. Hat eine *T-Snake* ihr Gleichgewicht erreicht, d. h. ihre Topologie ändert sich nicht mehr, kann das Simplexgitter deaktiviert und das Modell als eine parametrisierte *Snake* weiter ausgeführt werden. Die internen Energiebedingungen werden dann gleichmäßiger verteilte Modellpunkte erzeugen.

3.2 *k-means* Algorithmus

Beim *k-means* Algorithmus handelt es sich um ein *Cluster*-Analyseverfahren. Während der Aufteilung der Datenpunkte $M := \{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n\} \subseteq \mathbb{R}^d$, in k disjunkte Teilmengen, werden Punkte mit ähnlichen Eigenschaften in gleichen Klassen zusammengefaßt. Datenpunkte mit unterschiedlichen Eigenschaften sind in unterschiedlichen Klassen zu finden. Als Ähnlichkeitsmaß zwischen zwei Punkten dient gewöhnlich der euklidische Abstand.

Für die Bewertung des Segmentierergebnisses führen wir nun eine Fehlerfunktion ein. Dazu betrachten wir die Datenpunkte einer Klasse. Als Klassenzentrum definieren wir

$$\mathbf{z}_j := \frac{1}{n_j} \sum_{i=1}^{n_j} \mathbf{m}_{j,i}, \tag{3.17}$$

wobei n_j die Anzahl der Punkte der Klasse ω_j ist. Mit $\mathbf{m}_{j,i}$ bezeichnen wir den i-ten Datenpunkt der Klasse ω_j. Die Fehlerfunktion ist nun die Summe der quadratischen Abstände der Datenpunkte zu ihrem jeweiligen Bereichszentrum:

$$\text{err} := \sum_{j=1}^{k} \sum_{i=1}^{n_j} \left\| \mathbf{m}_{j,i} - \mathbf{z}_j \right\|^2. \tag{3.18}$$

Diese Definition der Fehlerfunktion minimiert die Varianzen der Datenpunkte der einzelnen Klassen.

3.2.1 Paarweises Zusammenführen (*Pairwise Agglomerative Clustering*)

Die einfachsten Art der Partitionierung von Datenpunkten ist die Methode des PAARWEISEN ZUSAMMENFÜHRENS [6]. Am Beginn des Algorithmus bildet jeder Datenpunkt eine eigene Klasse, mit sich selbst als Klassenzentrum. In jedem darauffolgenden Schritt S_j werden jene beiden Klassen miteinander verschmolzen, deren jeweiliges Klassenzentrum den geringsten Abstand zueinander aufweist. Das Klassenzentrum des neu entstandenen Bereiches ist der Mittelwert der Klassenzentren der verschmolzenen Bereiche. Das Zusammenführen von Bereichen wird solange fortgesetzt, bis die gewünschte Anzahl von *Clustern* erreicht ist. Abbildung 3.6 zeigt die Anwendung dieser Methode auf einer kleine Menge von Punkten.

Diese naheliegende Vorgehensweise hat jedoch zwei große Nachteile:

1. Der Algorithmus ist zur Analyse von großen Datenmengen völlig ungeeignet. Die Methode ist rechnerisch sehr ineffizient. Jeder Schritt macht die Berechnung der Distanzen zwischen jedem einzelnen Datenpunkt und deren Vergleich notwendig.

2. Der Algorithmus führt zwar immer zu der gewünschten Anzahl von Bereichen, die berechneten Zentren sind jedoch nicht notwendigerweise *repräsentativ* für diese Bereiche, d. h. sie stimmen mit den Mittelwerten nicht überein.

Der erste Nachteil kann durch Angabe der benötigten Distanzberechnungen gezeigt werden. Es werden $n-k$ Schritte (wie in Abbildung 3.6 gezeigt) benötigt, bis die Datenpunkte auf die gewünschte Anzahl von Bereichen reduziert wurde. Die Anzahl der Datenpunkte wird mit n und die Anzahl der gewünschten Bereiche mit k bezeichnet. Im i-ten Schritt sind

$$w_i = \sum_{j=1}^{n-i+1} ((n-i+1)-j) = \frac{1}{2}\left(i^2 - i + n^2 + n - 2\cdot i\cdot n\right)$$

Abstandsberechnungen notwendig. Die Gesamtzahl der notwendigen Distanzberechnungen ist:

$$w = \sum_{i=1}^{(n-k)} \sum_{j=1}^{(n-i+1)} (n-i+1-j) = \frac{1}{6}\left(n^3 - n - k^3 + k\right). \tag{3.19}$$

Daraus folgt eine Komplexität von $O\left(n^3\right)$, für $n \gg k$.

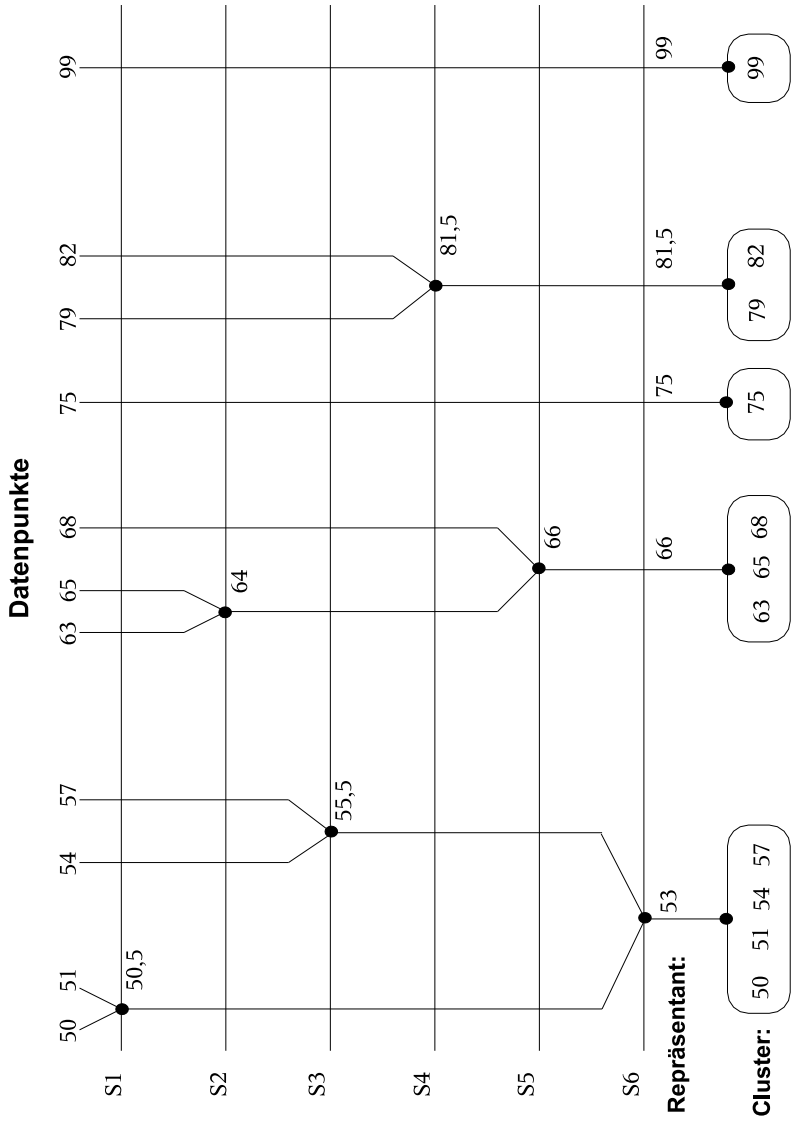


Abbildung 3.6: Paarweises Zusammenführen (*Pairwise Agglomerative Clustering*): Die Abbildung zeigt den Ablauf dieses Algorithmus. Zu Beginn bildet jeder Datenpunkt einen Bereich, mit sich selbst als Zentrum. In jedem weiteren Schritt S_j werden jene zwei Bereiche zusammengeführt, deren jeweiliges Zentrum den geringsten Abstand zueinander aufweist. Das Zentrum des neu entstandenen Bereichs wird auf den Mittelwert der beiden vorherigen Bereichszentren gesetzt. Das Verschmelzen von Bereichen wird dann beendet, wenn die gewünschte Anzahl von Bereichen erreicht ist.

3.2.2 Direkter *k-means* Algorithmus

Eine Verbesserung der Methode des Paarweisen Zusammenführens stellt der Direkte *k-means* Algorithmus dar [1]. Hier werden die Zentren der Bereiche, in die die Datenpunkte aufgeteilt werden sollen, zunächst geschätzt. Aufgrund dieser Schätzung wird eine erste Partitionierung vorgenommen, d. h. die Datenpunkte werden denjenigen Klassen zugeordnet, deren Abstand zum Klassenzentrum am geringsten ist. Aus dieser Partitionierung erfolgt eine neue Schätzung der Bereichszentren (durch die Berechnung der Bereichsmittelwerte). Die Neuschätzung der Zentren hat wiederum eine erneute Partitionierung zur Folge usw. Der Algorithmus bricht ab, wenn die Fehlerfunktion (Gleichung 3.18) einen bestimmten Wert unterschreitet oder sich keine (wesentliche) Änderungen in den Partitionierungen mehr ergeben. Algorithmus 6 faßt die wesentlichen Schritte noch einmal zusammen.

Algorithm 6 Vorgehensweise des Direkten *k-means* Algorithmus.

1. Initialisieren der k Repräsentanten $\{\mathbf{z}_1, \mathbf{z}_2, ..., \mathbf{z}_k\}$ mit zufälligen Punkten aus der zu partitionierenden Datenmenge. *Werden die Zentren so gewählt, daß sie* ***nicht*** *mit einem der Datenpunkte übereinstimmen, kann es vorkommen, daß ein Zentrum „iteriert" wird, dem keine Datenpunkte zugeordnet worden sind. Es entsteht dann eine Klasse ohne zugehörige Datenpunkte.*

2. Ein Datenpunkt $\mathbf{m}_i$ aus $\{\mathbf{m}_1, \mathbf{m}_2, ..., \mathbf{m}_n\}$ wird jenem Bereich ω_j zugeordnet, dessen Abstand zum Bereichszentrum $\mathbf{z}_j$ am geringsten ist. Jeder Punkt wird genau einem Bereich zugeordnet.

3. Neuberechnung der Bereichszentren $\mathbf{z}_j$ (Gleichung (3.17)).

4. Wiederholung der Schritte 2 und 3, bis sich die Bereiche nicht mehr ändern oder die Fehlerfunktion einen bestimmten Wert unterschreitet.

Der Direkte *k-means* Algorithmus stellt eine wesentliche Verbesserung gegenüber der Methode des Paarweisen Zusammenführens dar. Hier besteht jedoch die Gefahr, daß der Algorithmus auf ein lokales Minimum — bezüglich der Fehlerfunktion (3.18) — partitioniert.

Abbildung 3.7 zeigt den Algorithmus in Aktion. Getestet wurde der Algorithmus mit zufällig erzeugten Datenpunkten. Die Datenpunkte waren dabei multivariat normalverteilt:

$$p_{\vec{\mu},\Sigma}(\mathbf{m}) = \frac{1}{\sqrt{2\pi\,|\Sigma|}} \cdot e^{-\frac{1}{2}\cdot(\mathbf{m}_i-\vec{\mu})^T\Sigma^{-1}(\mathbf{m}_i-\vec{\mu})}, \tag{3.20}$$

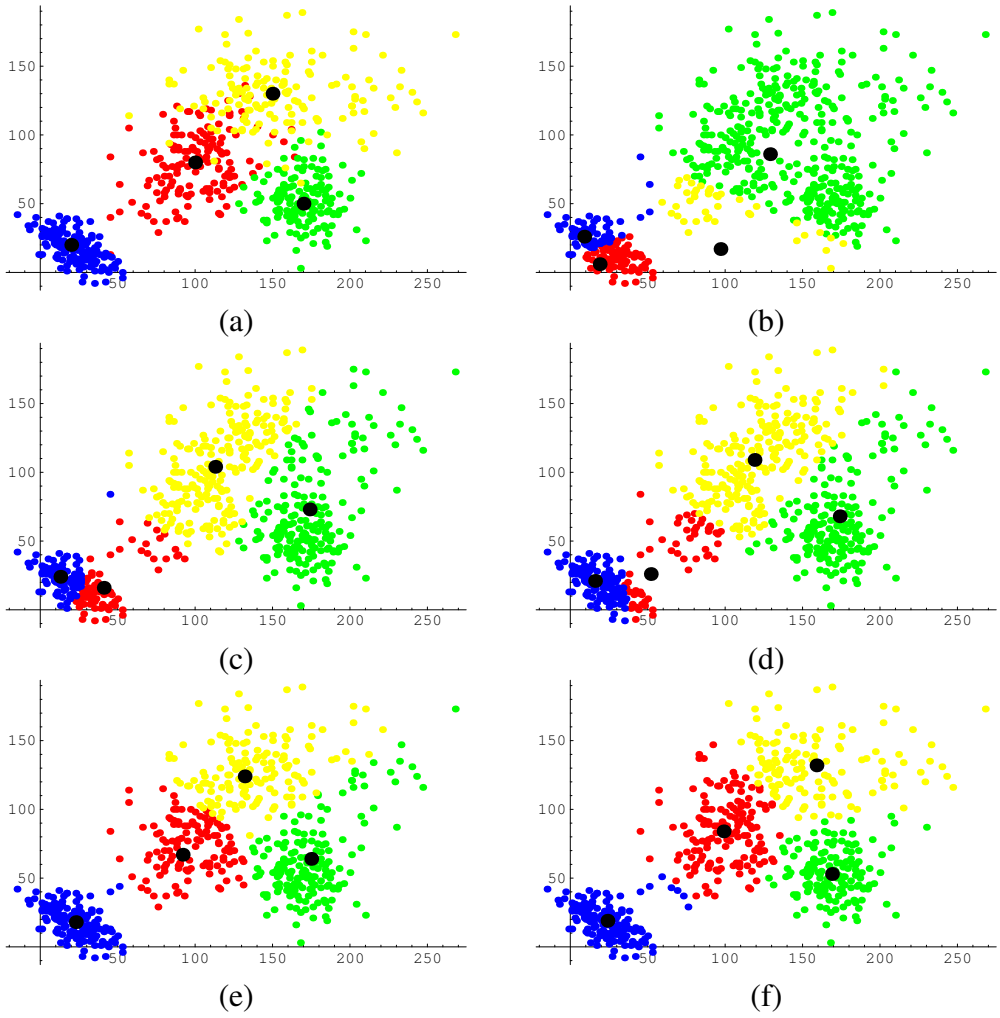


Abbildung 3.7: In dieser Abbildung sind einige der Iterationsschritte des Direkten *k-means* Algorithmus zu sehen. Die schwarzen (dicken) Punkte kennzeichnen die gefundenen Bereichszentren. In Bild (a) ist die „wirkliche" Aufteilung der zweidimensionalen Datenpunkte zu sehen. Innerhalb der Bereiche sind die Datenpunkte multivariat normalverteilt. Jede Klasse besteht aus 150 Punkten. Nach 24 Iterationen terminierte der Algorithmus.
Bild (b) zeigt die Punktaufteilung nach der Initialisierung; die Bereichsmittelpunkte wurden zufällig gewählt. Bild (c) zeigt die Aufteilung nach 9 Iterationen, Bild (d) nach 12 Iterationen, Bild (e) nach 15 Iterationen und Bild (f) nach der letzten Iteration.

wobei $\overrightarrow{\mu}$ dem Bereichszentrum z entspricht und Σ die Kovarianzmatrix bezeichnet. In Tabelle 3.1 sind die Werte der Parameter angegeben, mit denen die Datenpunkte erzeugt wurden. Weiters finden sich hier auch die, durch den Algorithmus geschätzten, Werte für Bereichsmittelpunkt und Kovarianzmatrix. Von ins-

Tabelle 3.1: Diese Tabelle zeigt die Parameter, mit denen die Datenpunkte in Abbildung 6 erzeugt wurden. Die Spalte z enthält die, durch den Algorithmus geschätzten, Bereichsmittelwerte und Spalte $\hat{\Sigma}$ die empirische Kovarianzmatrix.

Bereich	$\overrightarrow{\mu}$	Σ	z	$\hat{\Sigma}$
1	$(20,20)$	$\begin{pmatrix} 200 & -90 \\ -90 & 120 \end{pmatrix}$	$(24,19)$	$\begin{pmatrix} 240,81 & -45,44 \\ -45,44 & 140,06 \end{pmatrix}$
2	$(170,50)$	$\begin{pmatrix} 300 & 0 \\ 0 & 300 \end{pmatrix}$	$(169,53)$	$\begin{pmatrix} 283,41 & 12,56 \\ 12,56 & 296,82 \end{pmatrix}$
3	$(100,80)$	$\begin{pmatrix} 500 & 200 \\ 200 & 500 \end{pmatrix}$	$(99,84)$	$\begin{pmatrix} 309,37 & 26,74 \\ 26,74 & 563,11 \end{pmatrix}$
4	$(150,130)$	$\begin{pmatrix} 2000 & 0 \\ 0 & 500 \end{pmatrix}$	$(159,132)$	$\begin{pmatrix} 1107,84 & -31,01 \\ -31,01 & 422,13 \end{pmatrix}$

gesamt 600 Punkten wurden 550 Punkte richtig zugeordnet, d. h. jener Klasse, durch die sie erzeugt wurden. Daraus ergibt sich ein Klassifikationsfehler von $1 - \frac{550}{600} \approx 0,083$ oder rund 8 %.

Zur Verbesserung des Laufzeitverhalten des *k-means* Algorithmus gibt es prinzipiell zwei Möglichkeiten:

1. Es werden die Ergebnisse der vorhergehenden Iterationen benutzt, um die Anzahl der Distanzberechnungen zu minimieren.
   Der P-CLUSTER ALGORITHMUS [9] zieht Vorteil aus der Tatsache, daß sich nach den ersten Iterationsschritten die Zusammensetzung der Bereiche nur mehr wenig ändert. Es werden Heuristiken verwendet, die das nächstgelegenen Zentrum zu einem Datenpunkt bestimmen.

2. Die zweite Verbesserungsmöglichkeit liegt in der Verwendung von speziellen Datenstrukturen, die das Finden des nächstgelegenen Bereichszentrums beschleunigen. Ein Beispiel dafür ist die Speicherung der Datenpunkte in einem kd-Baum. Da die Position eines Datenpunktes im kd-Baum den Bereich einschränkt, in dem sich dieser Punkt im Raum befindet, kann die Suche nach dem Bereichsrepräsentanten mit dem geringsten Abstand beschleunigt werden.

Eine prinzipielle Einschränkung des *k-means* Algorithmus ergibt sich aus der Tatsache, daß es sich bei diesem um einen linearen Diskriminator handelt, d. h. im zweidimensionalen Merkmalsraum sind die Klassengrenzen Geraden, bzw. Geradensegmente. Klassengrenzen, die keiner Geraden entsprechen, können vom *k-means* Algorithmus also nicht gefunden werden. Abbildung 3.8 zeigt das Segmentierergebnis mit eingezeichneten Klassengrenzen. Dabei können die Diskriminatoren als Voronoikanten, mit den Bereichszentren als Voronoipunkte, aufgefaßt werden.

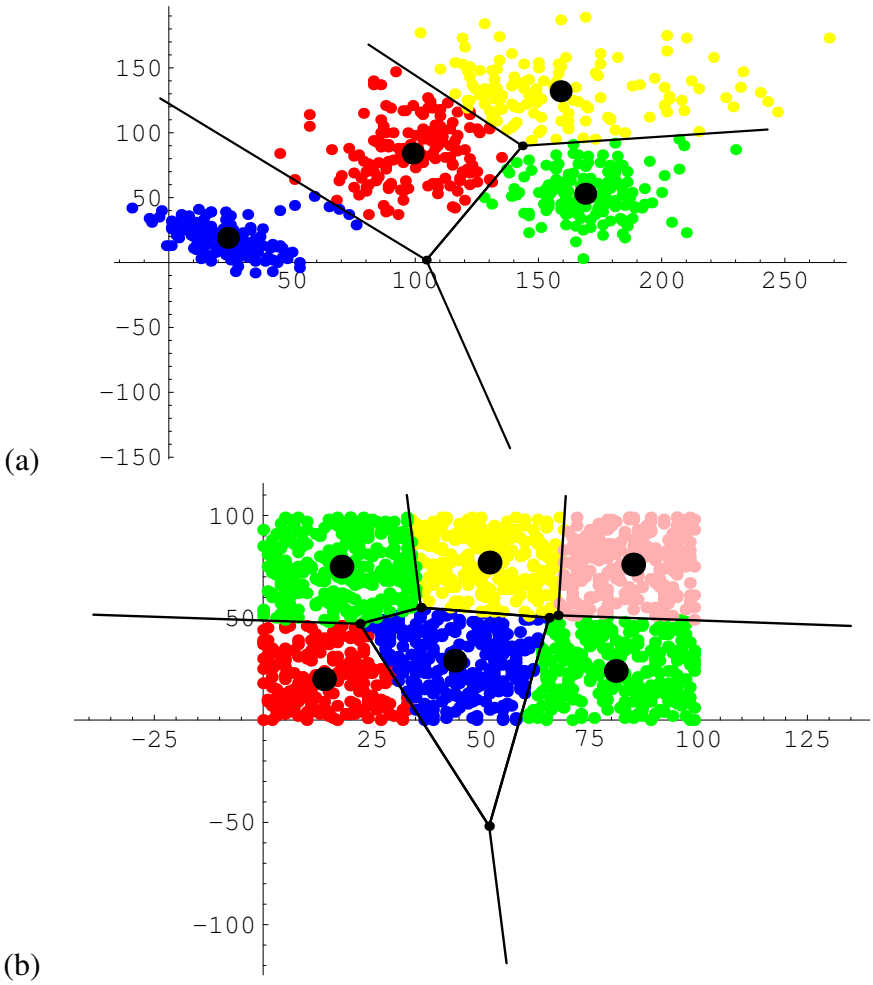


Abbildung 3.8: Diese Abbildung zeigt die Klassengrenzen, wie sie beim *k-means* Algorithmus entstehen. Die Begrenzungslinien können auch als Voronoikanten, mit den Bereichsmittelpunkten als Voronoipunkte, aufgefaßt werden. Bild (a) zeigt die Diskriminanten aus dem Beispiel in Abbildung 3.7. Bild (b) zeigt das Segmentierergebnis bei gleichverteilten Punkten. Bei Gleichverteilung der Datenpunkte kommt die Eigenschaft des *k-means* Algorithmus zu tragen, die Varianzen der einzelnen Bereiche zu minimieren.

Kapitel 4

Statistische Klassifikation

4.1 Allgemeines

Klassifikatoren stehen im Zentrum von Mustererkennungssystemen. Diese zeichnen sich vor allem durch ihre datengetriebene Lern- und Generalisierungsfähigkeit aus. In medizinischen Bildanalyse- und Erkennungssystemen werden sie sowohl zur Segmentierung als auch zur computergestützten Erkennung diagnostisch relevanter Bildstrukturen (Tumore, Läsionen usw.) eingesetzt.

Klassifikatoren unterscheiden sich von *Cluster*-Analyseverfahren durch die Art der Aufteilung der Daten in Äquivalenzklassen. Bei der *Cluster*-Analyse werden keine Annahmen über die Art der zu klassifizierenden Datenpunkte getroffen. Die einzigen Informationen die zur Verfügung stehen sind die Datenpunkte selbst. Jedes neue Datum wird, unabhängig von den übrigen, (genau) einer Klasse zugeordnet. Klassifikatoren hingegen werden auf eine bestimmte Art von Daten trainiert, d. h. die Parameter des darunter liegenden Modells werden durch das Training festgelegt. Ein Klassifikator arbeitet also nur korrekt, wenn die getroffene *Ähnlichkeitsannahme* der Datenpunkte erfüllt ist.

Die Fähigkeit zur Klassifikation erlangen Klassifikatoren durch Training an ausgewählten Datensätzen. Der Lernprozeß besteht dabei in der Schätzung der Modellparameter Θ. Die Wahl des Modells ist problemabhängig und muß *a priori* erfolgen.

Die in diesem Kapitel vorgestellten statistischen Klassifikationsverfahren werden auch als Diskriminanzanalyseverfahren oder parametrische Klassifikationsverfahren bezeichnet. Der prinzipielle Ablauf eines Segmentiervorgangs mittels (statistischer) Klassifikation ist in Abbildung 4.1 dargestellt. Ausgangspunkt der Segmentierung ist der medizinische Datensatz (MR-Bild). Nach einer optionalen Merkmalsextraktion werden die Merkmalsvektoren dem Klassifikationsalgorithmus übergeben. Dieser nimmt die Aufteilung in die entsprechenden Äquivalenz-

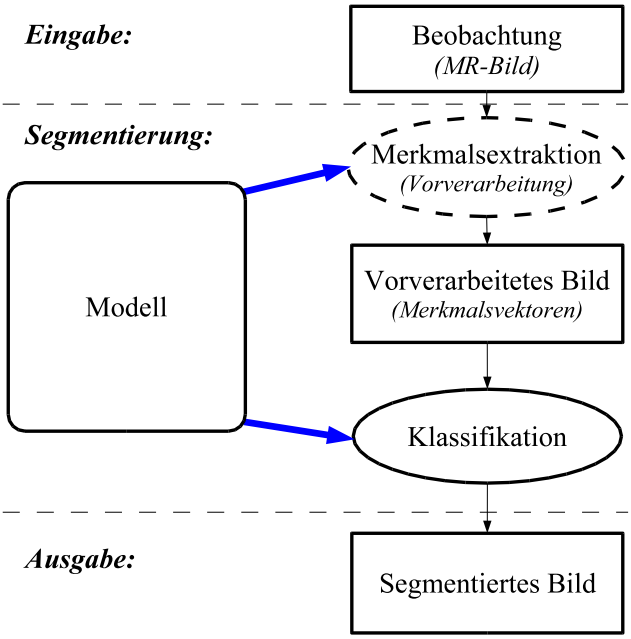


Abbildung 4.1: Diese Abbildung zeigt den prinzipiellen Ablauf des Segmentiervorgangs. Ausgehend von der Beobachtung (MR-Bild) findet eine Merkmalsextraktion (Vorverarbeitung) statt — dieser Schritt ist optional. Das so vorverarbeitete Bild besteht nun aus Merkmalsvektoren, auf denen die Klassifikation durchgeführt wird. Das Ergebnis der Klassifikation ist das segmentierte Bild.

klassen vor. Das Ergebnis dieser Klassifikation ist das segmentierte Bild. Auf welche Art und Weise die Merkmalsextraktion und die Klassifikation erfolgen, wird vom verwendeten Modell und dessen Parametern festgelegt.

4.1.1 Merkmalsextraktion

Das Ergebnis der Vorverarbeitung und Merkmalsgewinnung soll ein Merkmalsvektor $\mathbf{m}$ sein, aufgrund dessen eine möglichst einfache und zuverlässige Entscheidung über die Klassenzugehörigkeit getroffen werden kann. Die Vorverarbeitung dient dabei der Reduktion von Störeinflüssen.

Die Merkmale eines Bildes können dabei von ganz unterschiedlicher Art sein; sie hängen sehr stark von der gegebenen Anwendung ab. Eine übliche Einteilung der Merkmalsvektoren umfaßt folgende Gruppen:

- Geometrische Merkmale,
- Densiometrische Merkmale,
- Texturmerkmale,
- Merkmale der Kontextinformation.

Geometrische Merkmale (oder auch *Konturmerkmale*) enthalten Charakteristiken über die im Bild enthaltenen geometrischen Formen bzw. Gestalten. Dies sind z. B. Kanten und Linien — dabei werden die Ableitungen (Magnitude) des Bildes betrachtet.

Densiometrische Merkmale (oder auch *Merkmale der spektralen Signatur*) geben Informationen über das Reflexions- und Absorptionsvermögen einzelner Bildbereiche, in Abhängigkeit der Wellenlänge, wieder.

Texturmerkmale (oder *Strukturmerkmale*) beschreiben die Oberfläche des Bildes und sind in Verbindung mit Konturen und spektralen Signaturen von Bedeutung. Texturmerkmale sind z. B. Mittelwert und Standardabweichung einer Bildpunktumgebung (Nachbarschaft).

Merkmale der Kontextinformation (oder *Merkmale der Topologie*) eines Bildes geben Informationen über die Beziehung einer Klasse zu seiner Umgebung bzw. zu anderen Klassen wieder, ohne deren Größe, Form, Struktur usw. zu betrachten. Der Einsatz eines Gehirnatlases bei der Klassifikation unterschiedlicher Gehirnbereiche fällt in diesen Bereich.

Durch die Merkmalsextraktion wird der Merkmalsraum festgelegt, auf dem der Klassifikationsalgorithmus arbeitet. Findet keine Merkmalsextraktion statt, so sind die Merkmalsvektoren eindimensional — ausgenommen bei einem multimodalen Datensatz. Das einzige Merkmal ist hier der Grauwert der einzelnen Bildelemente[1].

4.1.2 Modellbeziehung

Wie bereits erwähnt, benötigt die statistische Klassifikation ein entsprechendes *(probabilistisches)* Modell. Der folgende Abschnitt liefert eine Motivation für diese Modellannahme.

[1] Die Dimensionalität der Merkmalsvektoren darf nicht mit der Dimensionalität des Bildes verwechselt werden, in das sie eingebettet sind. Die Dimension der Bilder, bei den hier betrachteten bildgebenden Verfahren, ist entweder zwei oder drei.

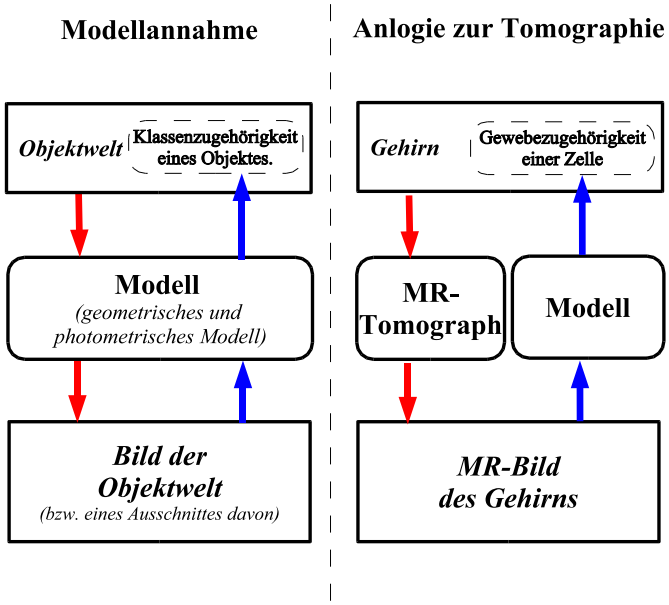


Abbildung 4.2: Diese Abbildung zeigt die Einbindung des Modells in den Prozeß der Bilderzeugung.

Abbildung 4.2 zeigt das Gesamtschema, in welches die Modellannahme eingebettet ist. Ausgangspunkt ist eine *abstrakte* Objektwelt. Diese Objektwelt enthält alle Informationen der darin enthaltenen Objekte. Alles *Wissen* über die Objektwelt erhalten wir über das *Modell*, welches Bilder von der Objektwelt, oder Ausschnitte davon, erzeugt. Realisierungen solcher Modelle sind z. B. Video- oder Photokameras, Röntgengeräte oder Computertomographen. Diese liefern uns zwei- bzw. dreidimensionale Bilder der Objektwelt.

Nun können wir zwar von einem Gehirn dreidimensionale Grauwertbilder erzeugen, unser Hauptinteresse gilt jedoch der Gewebe- bzw. Klassenzugehörigkeit der einzelnen Bildelemente. Die Idee ist also: wenn wir ein Modell finden können, welches die Eigenschaften eines Tomographen *möglichst genau* nachbildet, dann könnten wir durch Anwenden des „inversen" Modells auf die Objekte und deren Eigenschaften in der Objektwelt schließen. Eine vollständige Rekonstruktion der Objekte der Objektwelt wird jedoch in der Regel nicht möglich sein, da durch die *Modell-Transformation* eine Informationsreduktion stattgefunden hat. Beschränkt man sich nun auf bestimmte Aspekte der Objektwelt, so hoffen wir, ein Modell finden zu können, welches es uns erlaubt, vom gegebenen Bild auf

eben diese Aspekte der Objektwelt schließen zu können. Bei der Segmentierung medizinischer Bilddaten ist dieser Aspekt die Klassen- bzw. Gewebezugehörigkeit der einzelnen Bildpunkte (Zellen).

Das rechte Teilbild in Abbildung 4.2 zeigt, wie die allgemeine Modellannahme auf das Segmetierungsproblem von MR-Bildern des Gehirns übersetzt wurde. Die nach unten führenden Pfeile deuten an, wie die Objektwelt, in diesem Fall das Gehirn, durch einen Tomographen auf das *Bild des Gehirns* abgebildet wird. Die nach oben zeigenden Pfeile stellen das Segmentierungsproblem dar. Ausgehend vom Bild des Gehirns, soll mit Hilfe eines entsprechenden Modells auf die Gewebezugehörigkeit der Bildelemente geschlossen werden.

4.1.3 Modellparameter

Im vorigen Abschnitt wurde der Einsatz eines Modells für die Klassifikation motiviert. Welches konkrete Modell nun tatsächlich Anwendung findet ist damit noch nicht geklärt. Die Wahl des *richtigen* Modells ist ein hochgradig nicht triviales Problem und wird hier nicht behandelt (meist wird das *richtige* Modell durch Erfahrung und viele Versuche bestimmt). Wir wollen hier voraussetzen, ein passendes, (parametrisierbares) probabilistisches Modell gewählt zu haben. Das Problem, das sich jetzt stellt, ist die Schätzung der Parameter für dieses Modell. Die verschiedenen Methoden der Parameterschätzung werden von der sogenannten *Schätztheorie* behandelt. Die Schätzmethode welche am weitesten verbreitet ist, ist die MAXIMUM LIKELIHOOD METHODE (diese wird in Abschnitt 4.4.1 genauer vorgestellt).

4.2 Mathematische Grundlagen

In diesem Abschnitt sollen einige Begriffe aus der Statistik, die wir bei der statistischen Klassifizierung benötigen, wiederholt werden.

4.2.1 Grundbegriffe aus der Statistik und Wahrscheinlichkeitstheorie

Definition 4.2.1 *Sei $\Omega \neq \emptyset$ eine Menge (von Elementarereignissen). Eine Menge $\mathcal{A} \subseteq P(\Omega)$ heißt* σ-ALGEBRA *auf Ω, falls gilt:*

1. $\emptyset, \Omega \in \mathcal{A}$,
2. $a_1, \ldots, a_N \in \mathcal{A} \Rightarrow \bigcup_{i \in N} a_i \in \mathcal{A}$,
3. $a \in \mathcal{A} \Rightarrow \Omega \setminus a \in \mathcal{A}$.

Definition 4.2.2 *Sei $\mathcal{A}$ eine σ-Algebra auf $\Omega \neq \emptyset$ und $p : \mathcal{A} \to \mathbb{R}$ eine Abbildung mit folgenden Eigenschaften:*

1. *für alle $a \in \mathcal{A}$ gilt $0 \leq p(a) \leq 1$,*
2. *das sichere Ereignis besitzt die Wahrscheinlichkeit 1, d. h. $p(\Omega) = 1$ (*NORMIERUNGSAXIOM*),*
3. *sei a_n, $n \in \mathbb{N}$, eine Folge paarweise disjunkter Mengen in $\mathcal{A}$, so ist*

$$p\left(\bigcup_{n\in\mathbb{N}} a_n\right) = \sum_{n\in\mathbb{N}} p(a_n)$$

*(*ADDITIONSAXIOM*).*

Sind diese Eigenschaften erfüllt, so heißt p ein WAHRSCHEINLICHKEITSMASS *auf $(\Omega, \mathcal{A})$ und $(\Omega, \mathcal{A}, p)$ ein* WAHRSCHEINLICHKEITSRAUM.

Definition 4.2.3 *Eine (eindimensionale)* ZUFALLSVARIABLE *oder Zufallsgröße ist eine Abbildung $X : \Omega \to \mathbb{R}$, die jedem Element $\omega \in \Omega$ eine reelle Zahl $X(\omega) \in \mathbb{R}$ zuordnet.*

Die Werte, die eine Zufallsvariable annimmt, heißen Realisierungen oder Realisationen $x_1, x_2, \ldots$ Die Wahrscheinlichkeit, daß eine Zufallsgröße X den Wert x annimmt, wird mit $p(X = x)$ oder kurz mit $p(x)$ bezeichnet.

Definition 4.2.4 *Als* BEDINGTE VERTEILUNGSDICHTE *oder bedingte Wahrscheinlichkeit bezeichnet man die Größe*

$$p(x \mid y) = \frac{p(x \cap y)}{p(y)}, \tag{4.1}$$

wobei $p(x)$ die Wahrscheinlichkeit für das Eintreten des Ereignisses x bezeichnet.

Definition 4.2.5 SATZ DER TOTALEN WAHRSCHEINLICHKEIT: *Zerlegt man die Menge der Elementarereignisse Ω in disjunkte Teilmengen a_i mit $a_1 \cup \ldots \cup a_N = \Omega$ und $p(a_i) > 0$, dann gilt für beliebige Ereignisse x:*

$$p(x) = \sum_{i=1}^{N} p(a_i) \cdot p(x \mid a_i). \tag{4.2}$$

Definition 4.2.6 *Seien $p(x \mid a_i)$ und $p(a_i \mid x)$ bedingte Wahrscheinlichkeiten und a_i disjunkte Teilmengen von Ω mit $a_1 \cup \ldots \cup a_N = \Omega$ und $p(a_i) > 0$.*

$$p(a_i \mid x) = \frac{p(a_i) \cdot p(x \mid a_i)}{p(x)} = \frac{p(a_i) \cdot p(x \mid a_i)}{\sum_{i=1}^{N} p(a_i) \cdot p(x \mid a_i)}, \tag{4.3}$$

wird als BAYES'SCHES THEOREM *bezeichnet.*

Die Wahrscheinlichkeit $p(a_i \mid x)$ wird als *a posteriori* und $p(x \mid a_i)$ als *a priori* Wahrscheinlichkeit bezeichnet (dabei ist x das betrachtete Ereignis). Die zufälligen Ereignisse a_i heißen auch HYPOTHESEN (oder FALLUNTERSCHEIDUNGEN) in Ω.

Definition 4.2.7 GEMEINSAME WAHRSCHEINLICHKEIT*:*

$$p_Y(y) = \sum_{x \in X} p_{X,Y}(x,y). \tag{4.4}$$

Mit $p_Y(y)$ wird ausgedrückt, daß y nur Werte aus Y annehmen kann. Statt $p_{X,Y}(x,y)$ kann man auch $p_{X,Y}(x \cap y)$ schreiben, wobei auch hier gilt: $x \in X$ und $y \in Y$. Wenn klar ist aus welchen Mengen die Werte x und y stammen, schreibt man auch $p(x,y)$.

4.2.2 Zufallsprozesse

Prozesse, die neben einem deterministischen Verhalten auch einem zufälligen Einfluß unterliegen, werden als ZUFALLSPROZESSE bezeichnet. Das Verhalten des Prozesses wird in einer Zufallsvariablen $X(t,s)$ über die Zeit beobachtet, wobei der Parameter $t \in \{t_0, \ldots, t_m\}$ endlich viele Zeitpunkte markiert und s ein Element der Elementarereignisse ist. Eine Belegung der Zufallsvariablen X zu einem Zeitpunkt t, wenn s also ein bestimmtes Elementarereignis annimmt, heißt *Realisierung*. Die Zeit gibt die Entwicklungsrichtung des Zufallsprozesses an. Um auf n-dimensionale Zufallsprozesse zu verallgemeinern, wird von der Zeit abstrahiert und ein n-dimensionaler Vektor $\mathbf{t}$ eingeführt.

Viele Zufallsprozesse haben die Eigenschaft, daß ihre künftige Entwicklung nur vom gegenwärtigen Zustand, aber nicht vom Verlauf des Zufallsprozesses in der Vergangenheit abhängt. Man kann diese Eigenschaft eines Zufallsprozesses als „Gedächtnislosigkeit" des betreffenden Zufallsprozesses interpretieren, da dieser Zufallsprozess sozusagen keine Erinnerung an seine Vergangenheit hat.

Definition 4.2.8 *Die Eigenschaft der* Gedächtnislosigkeit *eines Zufallsprozesses wird als* MARKOV EIGENSCHAFT *bezeichnet. Einen Zufallsprozeß, der diese Eigenschaft besitzt, nennt man daher einen* MARKOV PROZESS.

4.2.3 Zufallsfelder

Sei $S = \{s_1, s_2, ..., s_n\}$ eine Menge von Feldern[2] (*sites*) und $\Lambda = \{\lambda_1, \lambda_2, ..., \lambda_m\}$ eine Menge von Zuständen[3] (*states*). Der Zustandsraum Λ ist im allgemeinen für jedes $s \in S$ verschieden; der dazugehörige Zustandsraum wird dann mit Λ_s bezeichnet. Die Definition des Zustandsraumes ist natürlich problemabhängig.

Definition 4.2.9 *Sei S eine endliche Menge von Feldern. Die Funktion*

$$x : S \to \Lambda_s : s \mapsto x(s) \tag{4.5}$$

wird als KONFIGURATION AUF *S mit dem Konfigurationsraum* $\Omega := \prod_{s \in S} \Lambda_s = \Lambda^S$ *bezeichnet.*

Mit den Mengen S und Λ konstruieren wir nun die Zufallsvariable X_s:

$$X_s : \Lambda^S \to \Lambda, \text{ mit } s \in S.$$

Mit $p(X_s = x(s))$ wird die Wahrscheinlichkeit bezeichnet, daß die Zufallsvariable X_s, bzw. das dazugehörige Feld s, den Wert $x(s)$ annimmt.

Definition 4.2.10 *Unter einem* ZUFALLSFELD *(*random field*) versteht man nun das geordnete Paar* $\left(\Lambda^S, p\right)$*, wobei p das Wahrscheinlichkeitsmaß von* Λ^S *ist.*

Bei der Untersuchung von Zufallsfeldern sind hauptsächlich die bedingten Wahrscheinlichkeiten

$$p(X_s \mid X_t = x(t)), \ s,t \in S, s \neq t \tag{4.6}$$

von Interesse. In vielen Fällen ist es nicht notwendig und/oder möglich die Abhängigkeiten über alle X_t zu modellieren. Man beschränkt sich für $t \in S$ deshalb auf eine kleine Teilmenge von S. Diese Teilmenge wird als Nachbarschaft von t bezeichnet.

Definition 4.2.11 *Mit einem* NACHBARSCHAFTSSYSTEM *bezeichnen wir eine Familie* $N = (N_s)_{s \in S}$*, sodaß gilt: (a) für* $s \in S$ *ist* N_s *eine Teilmenge von* $S \setminus s$ *und (b)* $t \in N_s \Rightarrow s \in N_t$ *für alle* s,t*.*

[2]Felder sind beliebige Objekte mit definierten Verbindungen zu Nachbarfeldern (Nachbarschaftsbeziehung). Man kann sich die Felder als Knoten eines Graphen vorstellen. Die Kanten entsprechen dabei den Nachbarschaftsbeziehungen. In unserem konkreten Anwendungsfall entspricht ein Feld einem Bildpunkt eines MR-Bildes. Die Nachbarschaftsbeziehung ist die normale Nachbarschaft von Bildpunkten in der Bildverarbeitung.

[3]Unter einem Zustand versteht man, in unserem Anwendungsfall, die konkrete Klassenzugehörigkeit eines Bildpunktes (Feldes).

Definition 4.2.12 *Ein Zufallsfeld* $\left(\Lambda^S, p\right)$ *wird als* MARKOVFELD *in Hinblick auf ein Nachbarschaftssystem N bezeichnet, wenn für alle* $s \in S$ *gilt:*

$$p\left(X_s = x(s) \mid X_t = x(t),\ t \in S,\ t \neq s\right) = p\left(X_s = x(s) \mid X_t = x(t),\ t \in N_s\right). \tag{4.7}$$

Die Bedingung in Gleichung 4.7 wird auch als MARKOV EIGENSCHAFT *bezeichnet [2].*

4.2.4 Gibbs Felder

Sei p ein strikt positives Wahrscheinlichkeitsmaß auf einer beliebigen Menge S, dann existiert eine reelle Funktion $\mathfrak{H}$, sodaß gilt:

$$p\left(\{x\}\right) = e^{-\mathfrak{H}(x)}. \tag{4.8}$$

Umgekehrt gilt:

$$p_{\mathfrak{H}}\left(\{x\}\right) := \frac{e^{-\mathfrak{H}(x)}}{Z}, \tag{4.9}$$

mit

$$Z := \sum_{y \in S} e^{-\mathfrak{H}(y)}. \tag{4.10}$$

Da der Ursprung von $\mathfrak{H}$ und Z in der statistischen Physik liegt, werden diese beiden Funktionen auch als ENERGIEFUNKTION und PARTITIONSFUNKTION bezeichnet [2].

Definition 4.2.13 *Sei S,* Λ *und N wie oben definiert:*

1. *Eine nicht leere Teilmenge C von S wird als* CLIQUE *bezeichnet, wenn für verschiedene* $s,t \in C$ *gilt:* $s \in N_t$ *und* $t \in N_s$*. Die Menge aller Cliquen wird mit* $\mathcal{C}$ *bezeichnet.*

2. *Als* GIBBS POTENTIAL *wird eine Familie* $V = (V_C)_{C \in \mathcal{C}}$ *bezeichnet, wobei* V_C *eine Funktion* $V_C : \Lambda^C \to \mathbb{R}$ *ist.*

3. *Sei* $V = (V_C)_{C \in \mathcal{C}}$ *ein Gibbs Potential. Die induzierte* ENERGIEFUNKTION $\mathfrak{H}_V$ *ist definiert durch:*

$$\mathfrak{H}_V(x) := \sum_{c \in C} V_C(x \mid c). \tag{4.11}$$

Theorem 4.2.1 *Das* GIBBS FELD*, welches durch V erzeugt wird, ist ein* MARKOV FELD*. Die Verteilung dieses Feldes ist gegeben durch:*

$$p(x) = \frac{1}{Z} e^{-\Sigma_{c \in C} V_C(x|c)}. \tag{4.12}$$

4.3 Modellierung

Die statistische Modellierung geht davon aus, daß die Merkmalsvektoren $\mathbf{m}$, nach einer Merkmalsextraktion, durch ein Zufallsexperiment erzeugt worden sind — die Merkmalsvektoren werden auch als Zufallsvariablen betrachtet. Auf Grund dieser Beobachtungen wird nun versucht, das Zufallsexperiment nachzubilden. Dabei wird vorausgesetzt, daß die Merkmalsvektoren jeder Klasse einer anderen Wahrscheinlichkeitsverteilung genügen (meist jedoch von der gleichen Familie). Die Wahl der *richtigen,* klassenbedingten Wahrscheinlichkeitsdichten $p(\mathbf{m} \mid \omega_i)$, mit $i \in \{1,...,k\}$, wird *a priori* getroffen. Dadurch wird das Problem der Bestimmung der klassenbedingten Wahrscheinlichkeitsdichten auf die Schätzung der Verteilungsparameter reduziert, die auf Basis der Stichprobe (Beobachtung) vorgenommen wird.

Definition 4.3.1 *Der Klassifikationsprozeß wird durch eine* ENTSCHEIDUNGSREGEL

$$e : M \quad \rightarrow \quad \Omega \tag{4.13}$$

beschrieben. Damit wird jedem Merkmalsvektor $\mathbf{m} \in M$ *eine Klasse* $\hat{\omega} \in \Omega$ *zugeordnet. Das* Dach *auf dem* ω *bedeutet, daß es sich bei der Entscheidungsregel nur um eine Schätzung der Klassenzugehörigkeit handelt und nicht notwendigerweise um die wirkliche Klasse. Weiters sei M die Menge aller Merkmalsvektoren und* Ω *die Menge der Äquvalenzklassen.*

Definition 4.3.2 *Sei* ω *die wahre Klasse des Merkmalsvektors* $\mathbf{m}$ *und* $\hat{\omega}$*, die durch die Entscheidungsregel* $e(\mathbf{m})$*, ermittelte Klasse. Die* KOSTENFUNKTION

$$v : \Omega \times \Omega \quad \rightarrow \quad \mathbb{R}$$

ordnet jedem Tupel $(\omega, e(\mathbf{m}))$ *Klassifikationskosten zu:*

$$v(\omega, e(\mathbf{m})) := \begin{cases} 0 & \textit{falls } e(\mathbf{m}) = \omega, \\ v_{\omega, e(\mathbf{m})} & \textit{sonst}, \end{cases} \tag{4.14}$$

wobei $v_{\omega,e(\mathbf{m})} > 0$ *die Kosten für eine Fehlklassifikation sind. Wir betrachten hier nur den Fall ohne Zurückweisung, d. h.* jeder *Merkmalsvektor wird einer Klasse zugeordnet (es gibt keine "ich-weiß-nicht-Klasse").*

Die Entscheidungsfunktion des Modells wird nun so gewählt, daß der Erwartungswert der Klassifikationskosten, dieser wird auch als *Risiko R* bezeichnet, minimiert wird.

$$R(e(\mathbf{m})) = E[v(\omega, e(\mathbf{m}))], \tag{4.15}$$

$$= \int_M \left(\sum_{\omega \in \Omega} v(\omega, e(\mathbf{m})) \cdot p(\mathbf{m}, \omega) \right) d\mathbf{m} = \text{minimal}. \tag{4.16}$$

Hierbei beschreibt die (unbekannte) gemeinsame Verteilungsdichte $p(\mathbf{m}, \omega)$, die durch die beiden Zufallsvariablen $\mathbf{m}$ und ω induziert wird, den mustererzeugenden Prozeß [8].

4.3.1 Bayes-Klassifikator

Für einen gegebenen Merkmalsvektor $\mathbf{m}$ ist die *optimale Entscheidung* für eine Klasse $\hat{\omega}$, mit der die Fehlerwahrscheinlichkeit minimiert wird, gegeben durch:

$$\begin{aligned} \hat{\omega} &= \text{argmax}_{\omega\in\Omega}\{p(\omega \mid \mathbf{m})\}, \\ &= \text{argmax}_{\omega\in\Omega}\left\{\frac{p(\mathbf{m} \mid \omega)\cdot p(\omega)}{p(\mathbf{m})}\right\}, \\ &= \text{argmax}_{\omega\in\Omega}\{p(\mathbf{m} \mid \omega)\cdot p(\omega)\}. \end{aligned} \tag{4.17}$$

Man berechnet also, unter der Bedingung der gegebenen Beobachtungen, die Wahrscheinlichkeit jeder Alternative und entscheidet sich dann für diejenige mit größter *a posteriori* Wahrscheinlichkeit.[4] Diesen Ansatz nennt man den (optimalen) BAYES-KLASSIFIKATOR. Jeder gute Klassifikator approximiert den Bayes-Klassifikator, da dieser optimal ist.

Die Entscheidungsregel für den Bayes-Klassifikator lautet somit:

$$e(\mathbf{m}) := \text{argmax}_{\omega\in\Omega}\{p(\mathbf{m} \mid \omega)\cdot p(\omega)\}. \tag{4.18}$$

4.3.2 Maximum-Likelihood-Klassifikator

Bei der Maximum-Likelihood-Segmentierung (ML-Segmentierung) wird eine Zuordnung der Merkmalsvektoren unabhängig von den merkmalsspezifischen *a priori* Wahrscheinlichkeiten $p(\omega)$ durchgeführt, d. h. die Wahrscheinlichkeit, daß ein bestimmtes Merkmal im Bild vorhanden ist, ist für alle Merkmale gleich. Sie ist daher insbesonders in Anwendungen von Bedeutung, bei denen die *a priori* Merkmalswahrscheinlichkeiten unbekannt sind bzw. nicht sinnvoll geschätzt werden können [8].

Da $p(\omega_1) = ... = p(\omega_k)$ — dies kann als Spezialfall des Bayes-Klassifikators aufgefaßt werden — ergibt sich als Entscheidungsregel für den ML-Klassifikator:

$$e(\mathbf{m}) := \text{argmax}_{\omega\in\Omega}\{p(\mathbf{m} \mid \omega)\}. \tag{4.19}$$

Die Klassifikation folgt somit dem Maximum Likelihood Prinzip, nach dem ein Merkmalsvektor $\mathbf{m}$ jener Klasse ω_i zugeordnet wird, bei der die Likelihood Funktion (4.24) maximal ist.

[4] Die verwendete Funktion $\text{argmax}_{\omega\in\Omega}\{...\}$ bedeutet, daß jenes *Argument* ω zurückgegeben wird, bei dem der Ausdruck in den geschwungenen Klammern maximal wird. (Nicht zu verwechseln mit der $\max_{\omega\in\Omega}\{...\}$ Funktion, bei dem der maximale Wert des *Klammerausdrucks* zurückgegeben wird.)

4.3.3 Mischverteilungs-Klassifikator

Bei der Mischverteilung (*mixture model*) wird angenommen, daß das Bild durch eine Linearkombination klassenspezifischer Verteilungsdichten von der Form

$$p(\mathbf{m};\Theta) = \sum_{i=1}^{k} \alpha_i \cdot p_i(\mathbf{m};\Theta_i) \tag{4.20}$$

entstanden ist, wobei $\mathbf{m}$ ein d-dimensionaler Merkmalsvektor, k die Anzahl der einzelnen Modelle und Θ_i die Parametermenge der Dichtefunktion p_i ist. Mit $\Theta := \{\alpha_1,...,\alpha_k,\Theta_1,...,\Theta_k\}$ werde alle (freien) Parameter des Modells bezeichnet; weiters wird $\sum_{i=1}^{k} \alpha_i = 1$ und $\alpha_i \geq 0$ für die Gewichte α_i vorausgesetzt.

Die klassenspezifischen Verteilungsfunktionen werden oft als multivariate Normalverteilungen angenommen mit der Parametermenge $\Theta := \{\alpha_i, \overrightarrow{\mu_i}, \Sigma_i\}_{i=1}^{k}$, wobei $\overrightarrow{\mu_i}$ dem Erwartungswert und Σ_i der Kovarianzmatrix entspricht. Den Erwartungswert $\overrightarrow{\mu_i}$ kann man sich als *Ort* der i-ten Klasse ("Klassenzentrum") im Merkmalsraum und die Kovarianzmatrix als die *Ausdehnung* der Klasse vorstellen; da es sich bei der Kovarianzmatrix um eine $d \times d$ Matrix handelt, kann man die Spaltenvektoren auch als die Hauptachsen des Superellipsoid betrachten, das die Klasse *umschließt*.

Der Klassifizierungsvorgang bei der Mischverteilung läuft nun folgendermaßen ab: Das Bild, bzw. die Merkmalsvektoren, wurden durch eine Mischverteilung erzeugt. Jede Komponente des Modells, d. h jede Verteilungsfunktion p_i, wird als Dichtefunktion der Klasse ω_i festgelegt. Weiters soll jeder Merkmalsvektor von genau einer Klasse erzeugt worden sein. Wenn wir die Verteilungsfunktionen, deren Parameter, die Gewichte und die Anzahl der Klassen kennen oder auch nur Schätzungen davon, dann können wir die Wahrscheinlichkeit, daß ein Merkmalsvektor $\mathbf{m}$ zur Klasse ω_i gehört, mit Hilfe des Bayes'schen Gesetztes (4.3) berechnen:

$$p(\omega_i \mid \mathbf{m}) = \frac{p_i(\mathbf{m};\Theta_i) \cdot \alpha_i}{\sum_{j=1}^{k} p_j(\mathbf{m};\Theta_j) \cdot \alpha_j}, \text{ mit } 1 \leq i \leq k. \tag{4.21}$$

Die Gewichte α_i entsprechen dabei den Klassenwahrscheinlichkeiten $p(\omega_i)$ und $p_i(\mathbf{m};\Theta_i)$ entspricht der klassenspezifischen Verteilungsdichte $p(\mathbf{m} \mid \omega_i)$ mit den Parametern Θ. Ein Merkmalsvektor wird dann jener Klasse zugeordnet, bei der die Wahrscheinlichkeit $p(\omega_i \mid \mathbf{m})$ maximal wird, d. h. für die Entscheidungsfunktion ergibt sich:

$$e(\mathbf{m}) := \text{argmax}_{\omega \in \Omega} p(\omega \mid \mathbf{m}). \tag{4.22}$$

Sollen für eine gegebene Stichprobe $M = \{\mathbf{m}_1,...,\mathbf{m}_N\}$ die Verteilungsparameter geschätzt werden, so geschieht dies gewöhnlich dadurch, indem man die

LIKELIHOOD FUNKTION der Mischverteilung (*mixture likelihood function*) maximiert:

$$\mathcal{L}(X;\Theta) = \prod_{j=1}^{N} \sum_{i=1}^{k} \alpha_i \cdot p_i\left(\mathbf{m}_j \mid \Theta_i\right). \tag{4.23}$$

Die Likelihood Funktion der Mischverteilung (4.23) hat in der Regel sehr viele lokale Maxima und es existiert keine geschlossene Form für die zu schätzenden Parameter. Das Finden eines globalen Maximums ist somit ein nicht triviales Problem. Nichtsdestoweniger ist dieses Modell sehr beliebt; ein Grund dafür ist, daß es effiziente iterative Methoden für die Parameterschätzung gibt. Erwähnt sei hier der EM-Algorithmus (*expectation maximization*; Abschnitt 4.4.2).

4.4 Parameterschätzung

4.4.1 Maximum-Likelihood-Schätzer

Die Aufgabe von statistischen Schätzmethoden — wie der Maximum-Likelihood-Methode — ist es, **Aussagen** über eine unbekannte Verteilungsfunktion f, einer gegebenen aber unbekannten Grundgesamtheit M *oder* deren Parameter Θ, an Hand von Stichproben aus M, zu machen. Diese Aussagen können, bei gegebener (oder vermuteter) Verteilung, Aussagen über (a) die Parameter Θ oder, (b) aufgrund der Stichprobe, über die Verteilung f sein.

Mit anderen Worten: Sei X eine *reelle* Zufallsvariable mit unbekannter Verteilungsfunktion, die von einem Parameter Θ abhängt (beim Parameter Θ kann es sich auch um einen Vektor handeln); der Wert für Θ ist dabei unbekannt. Gesucht wird eine Schätzung $\widehat{\Theta}$ für Θ, aufgrund von n Beobachtungen $\{x_1, x_2, ..., x_n\}$ einer Stichprobe, wobei auch diese Beobachtungen Vektoren sein können. Dies entspricht der Aussage (a).

Bei einer (vermuteten) Normalverteilung der Zufallsvariable X ist der Parameter $\Theta := \begin{pmatrix} \mu \\ \sigma^2 \end{pmatrix}$ ein zweidimensionaler Vektor. Eine Methode für die Schätzung des Parameters Θ, ist die Maximum-Likelihood-Methode (ML-Methode).

Bei der ML-Schätzmethode handelt es sich um eine Punktschätzung. Hier wird von einer gegebenen, bzw. vermuteten Verteilung und deren Parameter ausgegangen. Ausgangspunkt dieser Schätzung ist eine konkrete Stichprobe $\{x_1, ..., x_n\}$ vom Umfang n aus einer, nach f verteilten, Grundgesamtheit. Zur Schätzung von Θ definiert man die Likelihood Funktion (4.24) der konkreten Stichprobe.

Definition 4.4.1 *Sei X eine diskret verteilte Zufallsvariable mit den Wahrschein-*

lichkeiten $p(X = x_i)$, mit $i \in \{1,2,...,n\}$, so heißt die durch

$$\mathcal{L}(x_1,x_2,...,x_n;\Theta) \quad := \quad \prod_{i=1}^{n} p(X = x_i) \tag{4.24}$$

definierte Funktion die LIKELIHOOD FUNKTION *der konkreten Stichprobe.*

Definition 4.4.2 *Wird die Likelihood Funktion logarithmiert, so erhält man:*

$$\log \mathcal{L} = \log \left(\prod_{i=1}^{n} p(X = x_i) \right) = \sum_{i=1}^{n} \log p(X = x_i). \tag{4.25}$$

Diese Funktion wird auch als LOGARITHMIERTE LIKELIHOOD FUNKTION *bezeichnet.*

Das Prinzip der ML-Schätzmethode ist es, die Likelihood Funktion als Funktion des unbekannten Parameters zu betrachten und so eine Schätzung für Θ zu ermitteln, für die $\mathcal{L}$ maximal wird. Für den diskreten Fall bedeutet dies, aus allen möglichen Schätzungen für den unbekannten Parameter Θ denjenigen auszuwählen, für den die konkrete Stichprobe die größte Wahrscheinlichkeit besitzt:

$$\hat{\Theta} = \text{argmax}_{\Theta} \mathcal{L}(X;\Theta).$$

Zur Bestimmung dieses Maximums der Likelihood Funktion bildet man die erste Ableitung und setzt diese gleich Null. Anstelle der Gleichung $\frac{\partial \mathcal{L}}{\partial \Theta} = 0$ wird auch

$$\frac{\partial \log \mathcal{L}}{\partial \Theta} = 0 \tag{4.26}$$

verwendet. Diese Änderung beeinflußt die Lage des Maximums nicht, vereinfacht im allgemeinen jedoch die Rechnung. Die Ableitung von $\log \mathcal{L}$ bezeichnet man als MAXIMUM LIKELIHOOD GLEICHUNG. In der Regel müssen die partiellen Ableitungen von $\mathcal{L}$ gebildet werden, da Θ aus mehr als einem Parameter zusammengesetzt sein kann. Ist f gleich der Normalverteilung, so muß für die Schätzung der Parameter μ und σ^2 das Gleichungssystem

$$\begin{pmatrix} \frac{\partial \log \mathcal{L}}{\partial \mu} \\ \frac{\partial \log \mathcal{L}}{\partial \sigma^2} \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix}$$

gelöst werden.

Als Beispiel wollen wir nun für eine diskrete Stichprobe $X = \{x_1,x_2,...,x_n\}$, welche wir als normalverteilt annehmen, Maximum Likelihood Schätzer für den

Erwartungswert μ und der Standardabweichung σ berechnen. Für die Likelihood Funktion dieser Stichprobe erhalten wir:

$$\begin{aligned}\mathcal{L}\left(X;\mu,\sigma^2\right) &= \prod_{i=1}^{n}\frac{1}{\sqrt{2\pi\sigma^2}}\cdot e^{-\frac{(x_i-\mu)^2}{2\sigma^2}},\\ &= \left(\frac{1}{\sqrt{2\cdot\pi}}\right)^n\cdot\left(\frac{1}{\sigma}\right)^n\cdot e^{-h},\end{aligned}$$

wobei

$$h := \frac{1}{2\cdot\sigma^2}\sum_{i=1}^{n}(x_i-\mu)^2.$$

Die Logarithmierte Likelihood Funktion ist somit:

$$\log\mathcal{L}\left(X;\mu,\sigma^2\right) = -n\cdot\log\sqrt{2\cdot\pi} - n\cdot\log\sigma - h.$$

Die partielle Ableitung nach μ ergibt:

$$\begin{aligned}\frac{\partial\log\mathcal{L}}{\partial\mu} &= -\frac{\partial h}{\partial\mu},\\ &= \frac{1}{\sigma^2}\sum_{i=1}^{n}(x_i-\mu).\end{aligned}$$

Setzt man diese gleich Null, so folgt daraus:

$$\sum_{i=1}^{n}(x_i-\mu) = 0.$$

Als Schätzung für den Erwartungswert μ erhalten wir somit:

$$\hat{\mu} = \frac{1}{n}\sum_{i=1}^{n}x_i = \overline{x}.$$

Für die partielle Ableitung nach σ erhalten wir:

$$\begin{aligned}\frac{\partial\log\mathcal{L}}{\partial\sigma} &= -\frac{n}{\sigma} - \frac{\partial h}{\partial\sigma},\\ &= -\frac{n}{\sigma} + \frac{1}{\sigma^3}\cdot\sum_{i=1}^{n}(x_i-\mu)^2.\end{aligned}$$

Setzt man die Ableitung gleich Null und ersetzt man weiters μ durch $\hat{\mu}$, so erhält man, wenn man nach σ auflöst:

$$\hat{\sigma} = \sqrt{\frac{1}{n}\cdot\sum_{i=1}^{n}(x_i-\bar{x})^2}.$$

Es läßt sich jedoch zeigen, daß dieser Schätzer für die Standardabweichung nicht erwartungstreu ist. (Ein erwartungstreuer Schätzer ist: $\hat{\sigma} = \sqrt{\frac{1}{n-1} \cdot \sum_{i=1}^{n} (x_i - \bar{x})^2}$.)

Im allgemeinen ist es nicht möglich, eine geschlossene Form für die zu schätzenden Parameter anzugeben. In diesem Fall muß die Parameterbestimmung mit Hilfe eines anderen (numerischen) Verfahrens durchgeführt werden. Dabei wird die Parameterschätzung als Optimierungsaufgabe betrachtet. Bei dieser Sichtweise handelt es sich bei der Likelihood Funktion um eine *Kostenfunktion*, die angibt wie „gut" oder „schlecht" der gewählte Parameter Θ ist.

Das obige Beispiel wurde mit einer univariaten Normalverteilung durchgerechnet. Dies entspricht der Segmentierung ohne vorheriger Merkmalsextraktion — wir haben es nur mit Grauwerten zu tun. Nach einer Merkmalsextraktion stehen uns jedoch Merkmalsvektoren zur Verfügung, die dann einer *multivariaten* Klassenverteilung gehorchen. Die Likelihood Funktion hat nun folgende Form:

$$\mathcal{L}(X;\mu,\Sigma) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi|\Sigma|}} \cdot e^{-\frac{1}{2}\cdot(\mathbf{m}_i-\vec{\mu})^T\Sigma^{-1}(\mathbf{m}_i-\vec{\mu})}. \tag{4.27}$$

4.4.2 EM-Algorithmus

Der EM-Algorithmus (*expectation maximization algorithm*) [4] ist ein iterativer Algorithmus zur Berechnung der Parameter der ML-Parameterschätzmethode. Der Algorithmus wird eingesetzt, wenn (a) die beobachtete Datenmenge unvollständig ist oder (b) wenn die Likelihood Funktion durch die Annahme von zusätzlichen (versteckten) Parametern vereinfacht werden kann. Der zweite Fall ist vor allem in der Mustererkennung gegeben.

Wir nehmen an, daß die beobachtete Datenmenge $X := \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}$ durch eine Verteilungsfunktion erzeugt worden ist, wobei diese Datenmenge X nicht vollständig ist. Wir wollen weiters annehmen, daß ein vollständiger Datensatz $Z = \{X, Y\}$ existiert; Y bezeichnet die Menge der fehlenden Datenpunkte. Nun definieren wir die *gemeinsame Verteilung*

$$p_Z(\mathbf{z} \mid \Theta) := p_{X,Y}(\mathbf{x}, \mathbf{y} \mid \Theta) = p_{Y|X}(\mathbf{y} \mid \mathbf{x}, \Theta) \cdot p_X(\mathbf{x} \mid \Theta). \tag{4.28}$$

Diese *gemeinsame Verteilung* der Zufallsvariablen X und Y — wir können die Punkte aus X bzw. Y als Zufallsvariable auffassen, da bereits vorausgesetzt wurde, daß diese durch eine Verteilungsfunktion erzeugt wurden — stellt einen Zusammenhang zwischen fehlenden und beobachteten Daten her. Mit dieser neuen Dichtefunktion definieren wir eine neue Likelihood Funktion

$$\mathcal{L}(Z;\Theta) := \mathcal{L}(X,Y;\Theta) = p(X,Y \mid \Theta) \tag{4.29}$$

und nennen sie LIKELIHOOD FUNKTION DER VOLLSTÄNDIGEN DATEN. Diese neue Likelihood Funktion kann wiederum als eine Zufallsvariable betrachtet werden. Wir können also $\mathcal{L}(X,Y;\Theta) := h_{X,\Theta}(Y)$ für eine beliebige Funktion $h_{X,\Theta}$ definieren, wobei die Parameter X und Θ jeweils konstant sind und Y eine Zufallsvariable ist. Die ursprüngliche Likelihood Funktion $L(X;\Theta)$ wird als LIKELIHOOD FUNKTION DER UNVOLLSTÄNDIGEN DATEN bezeichnet.

Der EM-Algorithmus ermittelt zuerst den Erwartungswert der logarithmierten *Likelihood Funktion der vollständigen Daten* $\log p(X,Y \mid \Theta)$, hinsichtlich der unbekannten Menge Y, bei gegebenen Beobachtungsdaten X und vorläufiger Parameterschätzung. Dieser Teil wird durch Gleichung (4.30) beschrieben:

$$Q\left(\Theta,\Theta^{(i-1)}\right) := E\left[\log p(X,Y \mid \Theta) \mid X,\Theta^{(i-1)}\right], \tag{4.30}$$

wobei $\Theta^{(i-1)}$ die vorläufige Parameterschätzung bezeichnet, welche wir für die Berechnung des Erwartungswertes verwenden. Die neuen Parameter Θ werden dabei so optimiert, daß der Wert von Q erhöht wird.

Wichtig dabei ist, daß X und $\Theta^{(i-1)}$ konstant sind, Θ ist eine normale Variable welche wir anpassen und Y ist eine Zufallsvariable welche der Verteilung $f\left(\mathbf{y} \mid X,\Theta^{(i-1)}\right)$ gehorcht. Damit und mit Hilfe von Gleichung (4.28) kann die rechte Seite der Gleichung folgendermaßen umgeschrieben werden:

$$E\left[\log p(X,Y \mid \Theta) \mid X,\Theta^{(i-1)}\right] = \int_{\mathbf{y}\in\mathbb{Y}} \log p(X,\mathbf{y} \mid \Theta)\, f\left(\mathbf{y} \mid X,\Theta^{(i-1)}\right) d\mathbf{y}. \tag{4.31}$$

Dabei ist zu beachten, daß $f\left(\mathbf{y} \mid X,\Theta^{(i-1)}\right)$ die Grenzverteilung der nicht beobachteten Daten Y ist, die von X und der aktuellen Parameterschätzung abhängt. Die Werte, welche $\mathbf{y}$ annehmen kann, ist durch die Menge $\mathbb{Y}$ gegeben. Im günstigen Fall ist die Grenzverteilung eine einfache, analytische Funktion des Parameters $\Theta^{(i-1)}$ und den Daten X.

Betrachten wir z. B. die Funktion $h(\theta,Y)$, wobei θ konstant und Y eine Zufallsvariable ist, mit der Verteilungsfunktion $f_Y(y)$. In diesem Fall ist

$$E_Y[h(\theta,Y)] \;=\; \int_y h(\theta,y)\, f_Y(y)\, dy$$

eindeutig bestimmt und kann maximiert werden. Bei der "Festlegung" der Verteilung der fehlenden Daten werden wir darauf achten, daß der Ausdruck für den Erwartungwert möglichst einfach wird.

Der EM-Algorithmus setzt sich aus zwei Schritten zusammen:

1. Die Berechnung des Erwartungswertes wird als E-STEP bezeichnet. Wichtig dabei ist die Bedeutung der beiden Argumente von $Q\left(\Theta,\Theta^{(i-1)}\right)$. Das

erste Argument Θ bezieht sich auf den Parameter den wir optimieren wollen um die Likelihood Funktion zu maximieren. Das zweite Argument $\Theta^{(i-1)}$ bezieht sich auf den Parameter den wir verwenden um den Erwartungswert zu berechnen.

2. Der zweite Schritt (M-STEP) des EM Algorithmus maximiert den Erwartungswert, welchen wir im ersten Schritt berechnet haben. Somit erhalten wir:
$$\Theta^{(i)} = \text{argmax}_{\Theta} Q\left(\Theta, \Theta^{(i-1)}\right). \tag{4.32}$$

Diese beiden Schritte werden so oft wiederholt, bis eine vorgegebene Fehlerschranke erreicht bzw. unterschritten wird. Der EM-Algorithmus garantiert, daß bei jeder Iteration der Wert der Likelihood Funktion erhöht wird und der Algorithmus zu einem lokalen Maximum der Likelihood Funktion konvergiert.

4.4.3 Parameterschätzung für Mischverteilungen mittels EM

Die Parameterschätzung für Mischverteilungen ist eine der Hauptanwendungen der EM-Methode. Eine Definition der Mischverteilung wurde bereits in Abschnitt 4.3.3 gegeben. Unter der Voraussetzung, daß die einzelnen Verteilungen p_i der Mischverteilung normal verteilt sind, lassen sich mittels der EM-Methode folgende Schätzungen für die Parameter des Modells ableiten [14]:

$$\hat{\alpha}_i = \frac{1}{n} \cdot \sum_{j=1}^{n} \hat{\tau}_{i,j}, \tag{4.33}$$

$$\hat{\mu}_i = \frac{1}{n\hat{\alpha}_i} \cdot \sum_{j=1}^{n} \hat{\tau}_{i,j} \cdot \mathbf{m}_j \quad \text{und} \tag{4.34}$$

$$\hat{\Sigma}_i = \frac{1}{n\hat{\alpha}_i} \cdot \sum_{j=1}^{n} \hat{\tau}_{i,j} \cdot \left(\mathbf{m}_j - \hat{\mu}_i\right) \cdot \left(\mathbf{m}_j - \hat{\mu}_i\right)^T, \tag{4.35}$$

wobei

$$\tau_{i,j} = \frac{\alpha_i \left|\Sigma_i\right|^{-\frac{1}{2}} \cdot \exp\left[-\frac{1}{2}\left(\mathbf{m}_j - \mu_i\right)^T \Sigma_i^{-1} \left(\mathbf{m}_j - \mu_i\right)\right]}{\sum_{t=1}^{n} \alpha_t \left|\Sigma_t\right|^{-\frac{1}{2}} \cdot \exp\left[-\frac{1}{2}\left(\mathbf{m}_j - \mu_t\right)^T \Sigma_t^{-1} \left(\mathbf{m}_j - \mu_t\right)\right]}. \tag{4.36}$$

Dabei sei $\mathbf{m} \in M := \{\mathbf{m}_1, ..., \mathbf{m}_n\}$ die Menge der beobachteten Merkmalsvektoren und $\exp[x] := e^x$.

Die Gleichungen (4.33) bis (4.35) beinhalten bereits beide Schritte, den E-STEP und den M-STEP. Algorithmus 7 zeigt, wie sich die Parameter iterativ berechnen lassen.

Algorithm 7 Iterative Vorgehensweise bei der Parameterschätzung mittels des EM-Algorithmus.

1. Durchführen einer ersten initialen Schätzung der Parameter α_i, μ_i und Σ_i.
2. Berechnen von $\tau_{i,j}$ mittels der geschätzten Parameter.
3. Erneutes Berechnen von α_i, μ_i und Σ_i mit neuem $\tau_{i,j}$.
4. Wiederholen von Punkt (2) und (3) bis das Abbruchkriterium erfüllt ist.

4.5 ML-Klassifikation mittels *k-means* Algorithmus

In diesem Abschnitt soll nun gezeigt werden, wie die Parameterschätzung des ML-Klassifikators (Abschnitt 4.3.2) mit einer konkreten Verteilung durchgeführt wird [17]. Wir fassen die getroffenen Annahmen nochmals zusammen:

1. Die Klassenzugehörigkeit eines Bildpunktes ist unabhängig von der Klassenzugehörigkeit seiner Nachbarn und unabhängig von der räumlichen Lage des Bildpunktes.
2. Alle Klassen sind gleich wahrscheinlich.
3. Jeder Bildpunkt c ist *genau* einer Klasse zugeordnet.
4. Die Grauwerte der Bildpunkte der Klasse ω_i sind normalverteilt mit Mittelwert μ_i und Varianz σ_i^2, wobei die Varianzen als konstant und für alle Klassen gleich festgelegt werden.

Die Parameterschätzung erfolgt, indem die logarithmierte Likelihood Funktion

$$\log \mathcal{L}(c_1, ..., c_n; \Theta) = n \cdot \log \left[\frac{1}{\sqrt{2\pi\sigma_i^2}} \right] - \frac{1}{2\sigma_i^2} \sum_{j=1}^{n} \left(c_j - \mu_i \right)^2 \tag{4.37}$$

maximiert wird. Der Parameter Θ besteht aus dem Klassenbild F und den Klassenmittelwerten $\mu = \{\mu_1, ..., \mu_k\}$. Das Klassenbild F ordnet jedem Bildpunkt eine Klasse zu (das Klassenbild entspricht dem segmentierten Bild).

Die Anwendung des *k-means* Algorithmus auf das Segmentierproblem ist äquivalent dem Maximierungsproblem der Likelihood Funktion. Der *k-means* Algorithmus minimiert, bei der Aufteilung der Bildpunkte in k Bereiche, automatisch

die Bereichsvarianzen σ_i^2. Das Maximierungsproblem $\log \mathcal{L}(c_1,...,c_n;F,\mu)$ wird nun dadurch gelöst, indem abwechselnd für F und für μ maximiert wird:

$$F^{(m+1)} = \text{argmax}_F \left\{ \log \mathcal{L}\left(c_1,...,c_n;F,\mu^{(m)}\right)\right\}, \tag{4.38}$$

$$\mu^{(m+1)} = \text{argmax}_\mu \left\{ \log \mathcal{L}\left(c_1,...,c_n;F^{(m+1)},\mu\right)\right\}. \tag{4.39}$$

$F^{(m)}$ bzw. $\mu^{(m)}$ bezeichnen den Wert von F bzw. μ nach der m-ten Iteration. Gleichung (4.38) bedeutet also, daß jenes segmentierte Bild — aus der Menge aller segmentierten Bilder — ausgewählt wird, bei dem die Likelihood Funktion maximal wird. Analog ist Gleichung (4.39) zu lesen.

Für den Segmentiervorgang werden die Varianzen σ_i^2 als konstant vorausgesetzt. Die zu maximierende Likelihood Funktion läßt sich somit vereinfachen:

$$\log \mathcal{L}(c_1,...,c_n;F,\mu) \propto -\sum_{j=1}^{n} \left(c_j-\mu_i\right)^2. \tag{4.40}$$

Algorithmus 8 zeigt den Maximierungsvorgang der Likelihood Funktion. Vor dem ersten Iterationsschritt werden die Mittelwerte $\mu = \{\mu_1,...,\mu_k\}$ mit einem Schätzwert $\mu^{(*)}$ vorbelegt. Im M1-STEP werden die Bildpunkte $c_1,...,c_n$ jener Klasse ω_i zugeordnet, deren Grauwerte zum Klassenmittelwert μ_i den geringsten Abstand aufweisen (dies ist die Vorgehensweise des *k-means* Algorithmus). Der M2-STEP nimmt danach, mit dem im M1-STEP entstanden Klassenbild F, eine neue Schätzung der Mittelwerte vor. Der Minimierungsschritt wird so lange wiederholt, bis sich das Klassenbild nicht mehr ändert oder die Änderungen der Klassenmittelwerte einen gewissen Wert unterschreitet. Die Schätzung der Varianz, diese wird vom ML-Klassifikator benötigt, läßt sich nach der Minimierung der Likelihood Funktion mit Hilfe des Klassenbildes F durchführen:

$$\hat{\sigma}_\omega^2 = \frac{\sum_{i=1}^{n} \left(\hat{\mu}_\omega - c_i\right)^2 \cdot \delta(F_i=\omega)}{\sum_{i=1}^{n} \delta(F_i=\omega) - 1}. \tag{4.41}$$

Die Schätzung der Klassenmittelwerte $\hat{\mu}_\omega$ erfolgte bereits durch den *k-means* Algorithmus.

Bei genauerer Betrachtung handelt es sich bei diesem Algorithmus nicht mehr um einen reinen Klassifikationsalgorithmus, da nicht nur die Parameter geschätzt werden, sondern auch gleichzeitig die Segmentierung vorgenommen wird. Ist man primär nicht an der Schätzung der Modellparameter, sondern nur an der Segmentierung interessiert, so kann die Berechnung der Varianz entfallen, da das Bild durch den *k-means* Algorithmus bereits segmentiert wurde. In diesem Fall handelt es sich um einen *Cluster*-Algorithmus. Will man jedoch nur die Modellparametern schätzen, so ist die zusätzliche Datenstruktur des Klassenbildes überflüssig.

Algorithm 8 ML-Algorithmus [17]: Im S-STEP werden die Klassenmittelwerte mit einem Schätzwert $\mu^{(*)}$ vorbelegt. Im M1-STEP wird das Klassenbild F erzeugt. Der M2-STEP nimmt danach eine neue Schätzung der Merkmalsmittelwerte vor. Der Klassenmittelwert der Klasse ω wurde hier mit μ_ω bezeichnet. Weiters ist die Funktion δ (Kronecker-Symbol) definiert durch: $\delta(x=y) := 1$ wenn $x = y$ und 0 sonst. F_i bezeichnet die Klassenzugehörigkeit des i-ten Bildpunktes.

1. *Initialisieren der Mittelwerte (*S-STEP*):*
   (a) $\mu^{(0)} \leftarrow \mu^{(*)}$
2. *Maximierungsschritt (*M-STEP*):*
   (a) M1-STEP:
      i. Für alle $i \in \{1,...,n\}$
         A. $F_i^{(m+1)} \leftarrow \text{argmax}_{\omega\in\Omega}\left\{\left(c_i - \mu_\omega^{(m)}\right)^2\right\}$
   (b) M2-STEP:
      i. Für alle $\omega \in \Omega$
         A. $\mu_\omega^{(m+1)} \leftarrow \frac{\sum_{i=1}^n c_i \cdot \delta\left(F_i^{(m+1)}=\omega\right)}{\sum_{i=1}^n \delta\left(F_i^{(m+1)}=\omega\right)}$
         B. $m \leftarrow m+1$

Algorithmus 9 zeigt die Parameterschätzung, ohne Verwendung eines Klassenbildes.

Der in Algorithmus 9 verwendete Ausdruck für $\hat{\sigma}^2$, für die Schätzung der Varianz, läßt sich leicht herleiten:

$$\begin{aligned}\hat{\sigma}^2 &= \frac{1}{n-1}\sum_{i=1}^{n}(c_i-\hat{\mu})^2, \\ &= \frac{1}{n-1}\sum_{i=1}^{n}\left(c_i^2 - 2c_i\hat{\mu}+\hat{\mu}\right).\end{aligned}$$

Durch Umschreiben der Summen erhält man schließlich:

$$\hat{\sigma}^2 = \frac{1}{n-1}\left(\underbrace{\sum_{i=1}^{n} c_i^2}_{v} - 2\hat{\mu}\underbrace{\sum_{i=1}^{n} c_i}_{s} + \underbrace{\sum_{i=1}^{n}\hat{\mu}^2}_{n\cdot\hat{\mu}^2}\right). \tag{4.42}$$

Algorithm 9 Diese Form des ML-Algorithmus kommt ohne Klassenbild aus. Hier wird keine implizite Segmentierung vorgenommen; es werden *nur* die Modellparameter geschätzt. Dies ist vor allem dann sinnvoll, wenn das Modell anhand von ausgewählten Datensätzen *nur* trainiert werden soll.

1. *Initialisieren der temporären Variablen* **s**, **v** *und* **u**:
   (a) Für alle $i \in \{1,...,k\}$
      i. $s_i \leftarrow 0,\ v_i \leftarrow 0,\ u_i \leftarrow 0$
2. Für alle $j \in \{1,...n\}$
   (a) $i \leftarrow \text{argmax}_{l\in\{1,...,k\}} \left\{ \left(c_j - \mu_l^{(m)} \right)^2 \right\}$
   (b) $s_i \leftarrow s_i + c_j$
   (c) $v_i \leftarrow v_i + \left(c_j\right)^2$
   (d) $u_i \leftarrow u_i + 1$
3. Für alle $l \in \{1,...,k\}$
   (a) $\mu_l^{(m+1)} \leftarrow \frac{s_l}{u_l}$
4. Wiederholen von (1) bis (3) bis Fehlergrenze erreicht wurde.
5. *Berechnen der Varianz mit den Werten aus s und v:*
   (a) Für alle $l \in \{1,...,k\}$
      i. $\hat{\sigma}_l^2 \leftarrow \frac{1}{u_l - 1} \left(v_l - 2 \cdot \hat{\mu}_l \cdot s_l + n \cdot \hat{\mu}_l^2 \right)$

4.6 Maximum *a posteriori* Segmentierung

Bei diesem Verfahren (auch als *Markov Random Field* - Maximum *a posteriori* Segmentierung (MRF-MAP) bezeichnet) handelt es sich jetzt um ein *Cluster*-Analyseverfahren. Es stellt eine Erweiterung der ML-Klassifikation dar (in der ersten dargestellten Form, mit Hilfe eines Klassenbildes F). Diese Erweiterung setzt somit die gleichen Annahmen über das Bild voraus wie die ML-Segmentierung. Die Erweiterung ist von der Art, daß hier die Klassenzugehörigkeit eines Bildpunktes nicht mehr unabhängig von seinen Nachbarbildpunkten betrachtet wird. Diese räumliche Abhängigkeit wird durch Einführen eines Markov Feldes bzw. eines Gibbs Feldes erreicht (siehe Abschnitt 4.2.3).

Die Dichteverteilung des Klassenbildes F ist durch eine Gibbs Verteilung gegeben:

$$p(F_i) = \frac{e^{-\sum_{c\in C} V_C(F_i|c)}}{Z}, \tag{4.43}$$

wobei

$$Z = \sum_{i=1}^{n} e^{-\sum_{c\in C} V_C(F_i|c)} \tag{4.44}$$

ein konstanter Normierungsfaktor ist. Das Cliquenpotential V_C geht von der 18er-Nachbarschaft N_{18} eines Bildpunktes aus.

$$V_C(F_i \mid c) : \quad = \quad \begin{cases} 0 & \text{wenn} \quad F_i = c, \\ \beta & \text{wenn} \quad F_i \neq c \quad \text{und} \quad c \in N_6, \\ \frac{\beta}{\sqrt{2}} & \text{wenn} \quad F_i \neq c \quad \text{und} \quad c \in N_{18} \setminus N_6, \end{cases} \tag{4.45}$$

wobei $\beta > 0$. Das auf diese Weise definierte Cliquenpotential drückt die Vermutung aus, daß zwei benachbarte Bildpunkte mit größerer Wahrscheinlichkeit zur gleichen, als zu verschiedenen Klassen gehören.

Die Likelihood Funktion der Dichteverteilung des Klassenbildes ist somit:

$$\begin{aligned} \log \mathcal{L}(F) &= \log\left(\prod_{i=1}^{n} \frac{e^{-\sum_{c\in C} V_C(F_i|c)}}{Z}\right), \\ &= -\sum_{i=1}^{n}\left(\sum_{c\in C} V_C(F_i \mid c)\right) - n\log Z. \end{aligned} \tag{4.46}$$

Mit Hilfe der Bayes'schen Formel

$$p(F \mid c_1,...,c_n,\mu) = \frac{p(c_1,...,c_n \mid F,\mu)\cdot p(F)}{p(c_1,...c_n)} \tag{4.47}$$

und der Likelihood Funktion der ML-Klassifikation (siehe Gleichung (4.37))

$$\log p\left(c_1,...,c_n \mid F,\mu\right) = n \cdot \log\left[\frac{1}{\sqrt{2\pi\sigma_i^2}}\right] - \frac{1}{2\sigma_i^2}\sum_{j=1}^{n}\left(c_j-\mu_i\right)^2, \text{ mit } 1 \leq i \leq k, \tag{4.48}$$

(mit k wird die Anzahl der Merkmale bezeichnet) erhalten wird die *a posteriori* Likelihood Funktion

$$\begin{aligned}\log \mathcal{L}\left(F;c_1,...c_n,\mu\right) &= n \cdot \log\left[\frac{1}{\sqrt{2\pi\sigma_i^2}}\right] - \frac{1}{2\sigma_i^2}\sum_{j=1}^{n}\left(c_j-\mu_i\right)^2 - \\ &\quad \sum_{i=1}^{n}\left(\sum_{c\in C} V_C\left(F_i \mid c\right)\right) - n\log Z. \end{aligned} \tag{4.49}$$

Unter den gleichen Annahmen wie bei der ML-Segmentierung, d. h. unter Annahme einer konstanten Varianz, ergibt sich für die zu maximierende Funktion:

$$\log \mathcal{L}\left(F;c_1,...c_n,\mu\right) \propto -\sum_{j=1}^{n}\left(\left(c_j-\mu_i\right)^2 + \sum_{c\in C} V_C\left(F_j \mid c\right)\right) - n\log Z. \tag{4.50}$$

Da der letzte Term ($n\log Z$) aus Gleichung (4.50) das Maximum der Likelihood Funktion ebenfalls nicht verändert, wird auch dieser weggelassen:

$$\log \mathcal{L}\left(F;c_1,...c_n,\mu\right) \propto -\sum_{j=1}^{n}\left(\left(c_j-\mu_i\right)^2 + \sum_{c\in C} V_C\left(F_j \mid c\right)\right). \tag{4.51}$$

Für die praktische Anwendung von Gleichung (4.51) empfiehlt es sich, die Summe der Cliquenpotentiale mit dem Grauwertbereich (des zu segmentierenden Bildes) zu multiplizieren. Dadurch wird die Wahl von β unabhängig von der Farbtiefe des Bildes. Dies führt zur erweiterten Likelihood Funktion:

$$\log \mathcal{L}\left(F;c_1,...c_n,\mu\right) \propto -\sum_{j=1}^{n}\left(\left(c_j-\mu_i\right)^2 + R\cdot\sum_{c\in C} V_C\left(F_j \mid c\right)\right), \tag{4.52}$$

dabei ist R die Anzahl der im Bild verwendeten Farben.

Wie bei der ML-Klassifikation wird die Segmentierung durch einen verallgemeinerten *k-means* Algorithmus (Algorithmus 10) durchgeführt. Der M2-STEP ist der gleiche wie bei der ML-Klassifikation. Das Ergebnis ist somit eine *maximum a posteriori* (MAP) Lösung. Mit einem *passend* gewählten β (siehe dazu Abschnitt 6.3) erzielt diese Methode (bei verrauschten Bildern) im allgemeinen bessere Ergebnisse als die ML-Segmentierung. Nachteil ist jedoch eine mögliche „Überglättung" des Bildes und die dadurch verloren gehenden Details.

Algorithm 10 MRF-MAP Algorithmus [17]. Der M2-STEP ist gleich wie beim ML Algorithmus (8).

1. *Initialisieren der Mittelwerte (*S-STEP*):*

   (a) $\mu^{(0)} \leftarrow \mu^{(*)}$

2. *Maximierungsschritt (*M-STEP*):*

   (a) M1-STEP:

      i. Für alle $i \in \{1,...,n\}$

         A. $F_i^{(m+1)} \leftarrow \text{argmax}_{\omega\in\Omega} \left\{ \left(c_i - \mu_\omega^{(m)} \right)^2 + R \cdot \sum_{c\in C} V_C \left(F_i^{(m)} \mid c \right) \right\}$

   (b) M2-STEP*:*

      i. Für alle $\omega \in \Omega$

         A. $\mu_\omega^{(m+1)} \leftarrow \frac{\sum_{i=1}^n c_i \cdot \delta\left(F_i^{(m+1)}=\omega\right)}{\sum_{i=1}^n \delta\left(F_i^{(m+1)}=\omega\right)}$

         B. $m \leftarrow m+1$

4.7 Bewertung von Klassifikatoren

Die Bewertung von Klassifikatoren kann nach unterschiedlichen Gesichtspunkten erfolgen. Neben der (a) ERKENNUNGSLEISTUNG eines Klassifikators (Klassifikationsgenauigkeit) spielt auch (b) die KOMPAKTHEIT des verwendeten Modells, (c) die EFFIZIENZ (Klassifikationsgeschwindigkeit) und (d) die SKALIERBARKEIT (für große Datenmengen) eine wesentliche Rolle.

Das primäre Qualitätskriterium von Klassifikatoren soll im weiteren jedoch die Erkennungsleistung sein.

Definition 4.7.1 *Sei $e_\mathcal{K}$ die Entscheidungsfunktion des Klassifikators $\mathcal{K}$, $M_{TR} \subseteq M$ die Trainingsmenge und $M_{TE} \subseteq M$ die Testmenge; mit $S(\mathbf{m}) \in \Omega$ bezeichnen wir die tatsächliche Klassenzugehörigkeit des Merkmalvektors* $\mathbf{m}$.

- KLASSIFIKATIONSGENAUIGKEIT *(*classification accuracy*) von $\mathcal{K}$ auf M_{TE}:*

$$G_{\mathcal{K},TE} := \frac{|\{\mathbf{m} \in M_{TE} \mid e_\mathcal{K}(\mathbf{m}) = S(\mathbf{m})\}|}{|M_{TE}|}. \tag{4.53}$$

- TATSÄCHLICHER KLASSIFIKATIONSFEHLER *(*true classification error*):*

$$F_{\mathcal{K},TE} := \frac{|\{\mathbf{m} \in M_{TE} \mid e_\mathcal{K}(\mathbf{m}) \neq S(\mathbf{m})\}|}{|M_{TE}|}. \tag{4.54}$$

- BEOBACHTETER KLASSIFIKATIONSFEHLER *(*apparent classification error*):*

$$F_{\mathcal{K},TR} := \frac{|\{\mathbf{m} \in M_{TR} \mid e_{\mathcal{K}}(\mathbf{m}) \neq S(\mathbf{m})\}|}{|M_{TR}|}. \tag{4.55}$$

- GESAMTER KLASSIFIKATIONSFEHLER *bzw.* GESAMTFEHLERRATE *(*error rate*):*

$$F_{\mathcal{K},M} := \frac{|\{\mathbf{m} \in M \mid e_{\mathcal{K}}(\mathbf{m}) \neq S(\mathbf{m})\}|}{|M|}. \tag{4.56}$$

- GESAMTE KLASSIFIKATIONSGENAUIGKEIT *(*classification performance*):*

$$G_{\mathcal{K},M} := \frac{|\{\mathbf{m} \in M \mid e_{\mathcal{K}}(\mathbf{m}) = S(\mathbf{m})\}|}{|M|}. \tag{4.57}$$

Die Aufteilung der Merkmalsvektoren in eine Trainings- und Testmenge soll verhindern, daß das trainierte Modell unbemerkt auf die Trainingsdaten *überangepaßt* wird. Dieses Problem wird auch als *overfitting* bezeichnet (mehr dazu in Abschnitt 4.7.1).

Die Erkennungsleistung eines Klassifikators wird über die Gesamtfehlklassifikationswahrscheinlichkeit oder *kurz* Fehlklassifikationswahrscheinlichkeit (FKW) charakterisiert. Die FKW ist abhängig vom verwendeten Klassifikator und definiert durch:

$$P_F := 1 - \sum_{i=1}^{k} \int_{\mathfrak{R}_i} p(\mathbf{m} \mid \omega_i)\, p(\omega_i)\, d\mathbf{m}, \tag{4.58}$$

wobei k die Anzahl der Klassen und $p(\omega_i)$ die *a priori* Wahrscheinlichkeit der Klasse ω_i ist. Weiters bezeichnen wir mit $p(\mathbf{m} \mid \omega_i)$ die klassenbedingte Wahrscheinlichkeitsdichte und mit $\mathfrak{R}_i$ den Klassenbereich der Klasse ω_i. Das komplementäre Maß für den Klassifikationsfehler ist die Klassifikationsgenauigkeit:

$$P_G := \sum_{i=1}^{k} \int_{\mathfrak{R}_i} p(\mathbf{m} \mid \omega_i)\, p(\omega_i)\, d\mathbf{m}. \tag{4.59}$$

Eine Möglichkeit zur Schätzung der FKW besteht darin, die unbekannten *a priori* Wahrscheinlichkeiten $p(\omega_i)$ und Verteilungsdichten $p(\mathbf{m} \mid \omega_i)$ zu schätzen und in Gleichung (4.58) einzusetzen. Für den ML-Klassifikator ist diese Methode sehr einfach durchführbar, da durch die Parameterschätzung die klassenbedingten Verteilungsdichten bereits ermittelt wurden und die *a priori* Klassenwahrscheinlichkeiten auf $p(\omega_i) := \frac{1}{k}$ gesetzt wurden. In Abbildung 4.3 ist die FKW graphisch dargestellt; sie entspricht dem Schnitt der Flächen unter den beiden Verteilungsdichten. In dem Beispiel wurden die beiden Verteilungsdichten als normalverteilt

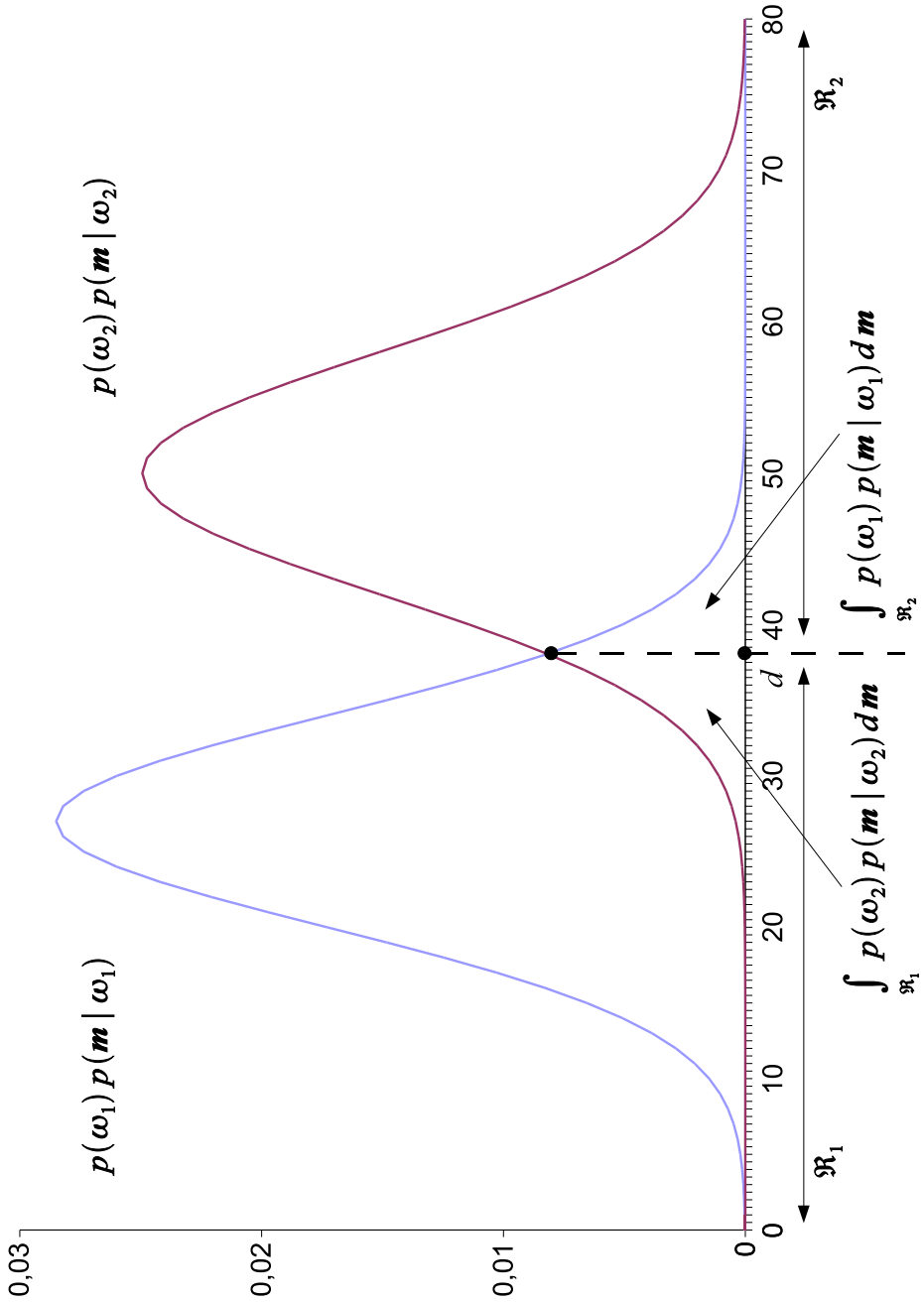


Abbildung 4.3: Diese Abbildung zeigt die klassenspezifischen Verteilungsdichten mit zwei Klassen im 1-dimensionalen Merkmalsraum. Die durch die beiden Pfeile gekennzeichneten Flächen entsprechen der FKW (Fehlklassifikationswahrscheinlichkeit). Die durch den optimalen Bayes-Klassifikator ermittelte Schwelle $d \approx 38,058$ ist durch die gestrichelte Linie gekennzeichnet.

angenommen, mit Erwartungswerten von $\mu_1 = 27$, $\mu_2 = 50$ und Varianzen von $\sigma_1^2 = 49$, $\sigma_2^2 = 64$. Die *a priori* Klassenwahrscheinlichkeiten wurden auf $\frac{1}{2}$ gesetzt. Setzt man diese Werte in Gleichung (4.58) ein — oder besser gesagt: wertet man diese Gleichung numerisch aus — so erhält man $p_F \approx 0,0618$.

Die soeben beschriebene Vorgehensweise führt jedoch zu einer zu optimistischen Schätzung der FKW; der Grund dafür liegt in den postulierten Verteilungsannahmen [8]. Eine bessere Möglichkeit ist, den gesamten Klassifikationsfehler als Schätzer für die FKW (Gleichung (4.56)) zu verwenden. Dies setzt jedoch die Kenntnis der *wahren* Klassenzugehörigkeit eines Merkmalvektors voraus. Solche *vorklassifizierten* Datensätze stehen jedoch beim unüberwachten Lernen meist nicht zur Verfügung.

4.7.1 Kreuzvalidierung (*cross validation*)

Die Kreuzvalidierung erlaubt die Bewertung der Generalisierungsfähigkeit eines Klassifikators. Hierbei wird geprüft, ob der Klassifikator in der Lage ist, die Zusammenhänge zwischen Merkmalsvektoren und Klassenzugehörigkeit zu verallgemeinern und dies auf unbekannte Daten anzuwenden. Dadurch wird vermieden, daß der Klassifikator in Hinblick auf eine bestimmte Datenmenge (Trainingsbilder) überangepaßt wird.

Hold-Out-Kreuzvalidierung

Die *Hold-Out*-Kreuzvalidierung (HO-CV) ist die intuitivste Form der Kreuzvalidierung. Dabei werden die verfügbaren Merkmalsvektoren in zwei disjunkte Mengen aufgeteilt, in eine *Trainingsmenge* und eine *Testmenge*. Die Parameterschätzung des Klassifikators wird anhand der Trainingsmenge vorgenommen, die Schätzung der FKW anhand der Testmenge.

Für die *Hold-Out*-Methode werden relativ umfangreiche Stichproben benötigt. Werden für die Trainings- und Testmenge gleich große Stichproben gewählt, so ist der daraus gewonnene Schätzwert für die FKW pessimistisch. Das bedeutet, der geschätzte Wert ist im Mittel zu groß. Bei kleinem Stichprobenumfang kann das Verfahren auch mehrmals, mit jeweils verschiedenen, zufälligen Trainings- und Teststichproben durchgeführt werden. (Dies führt dann zur m-fachen Kreuzvalidierung.)

m-fache Kreuzvalidierung

Hier werden die zur Verfügung stehenden Merkmalsvektoren in m disjunkte Teilmengen zerlegt. Für das Training werden die Merkmalsvektoren aus $m-1$ dieser

Teilmengen verwendet. Die verbleibende Menge dient zur Bewertung des Klassifikators.

Aus den so gefundenen m Modellen und Klassifikationsfehlern wird nun jenes Modell ausgewählt, bei dem die Schätzung für die FKW am geringsten war.

Leave-One-Out-Kreuzvalidierung

Diese Kreuzvalidierungsart wird auch als totale Kreuzvalidierung bezeichnet. Hier besteht die Teststichprobe aus einem und die Lernstichprobe aus $n-1$ Elementen, wobei n die gesamte Anzahl der verfügbaren Merkmalsvektoren ist. Die Anzahl der zur Fehlerratenschätzung zu trainierenden Klassifikatoren ist daher gleich der Anzahl der Merkmalsvektoren n. Die Fehlerrate ergibt sich dann als der relative Anteil der in den n Durchläufen falsch klassifizierten Merkmalsvektoren.

Kreuzvalidierung der Likelihood Funktion

Die bisher beschriebenen Kreuzvalidierungsmethoden benötigen, für die Bewertung der Generalisierungsfähigkeit eines Klassifikators, die Kenntnis der *wahren* Klassenzugehörigkeit eines Merkmalsvektors. Der gesamte Klassifikationsfehler $F_{\mathcal{K},M}$ diente dabei als Maß für die Generalisierungsfähigkeit. Ersetzt man nun dieses Maß durch die Likelihood Funktion, so ist auch ohne Kenntnis der wahren Klassenzugehörigkeit eine Kreuzvalidierung möglich [16].

Kapitel 5

Implementierung

In diesem Kapitel wird die Implementierung[1] zweier ausgewählter Segmentieralgorithmen beschrieben (ML-Klassifikator Abschnitt 4.5 und MAP-Klassifikator Abschnitt 4.6). Die Implementierung der Algorithmen erfolgte in der objektorientierten Programmiersprache JAVA. Im weiteren Text werden Klassen- und Schnittstellennamen in einem „`Schreibmaschinen`"-Zeichensatz gesetzt.

Das Ziel der Implementierung war, eine Klassenstruktur zu finden, welche in möglichst natürlicher und intuitiver Weise die beschriebenen theoretischen Grundlagen der (statistischen) Klassifikation (Abschnitt 4) abbildet. Abbildung 5.1 zeigt das UML-Diagramm der gewählten Klassenstruktur. Die Wurzel der Klassenhierarchie, der `Segmenter`, bildet die Schnittstelle für jegliche Art von Segmentieralgorithmen. Diese wird durch die Schnittstellen `Clusterer` und `Classifier` erweitert, gemäß der Unterscheidung zwischen *Cluster*-Analyseverfahren und Klassifikatoren. Bei diesen Schnittstellen handelt es sich um sogenannte *Markerinterfaces*[2] und sie dienen *nur* der besseren Lesbarkeit des Programms. Es ist in diesem Entwurf nicht vorgesehen, die Schnittstelle `Segmenter` direkt zu implementieren. Eine Implementierung erfolgt entweder über die `Clusterer` oder `Classifier` Schnittstelle.

Die Schnittstelle `Estimator` bildet die Basis für die Implementierung der verschiedenen Parameterschätzverfahren. Dabei besteht zwischen dem `Estimator` und dem `Classifier` ein enge Bindung; das Ergebnis des Schätzers (`Estimator`) ist ein Klassifikator (`Classifier`). Die Klasse `MLClassifier` enthält die Implementierung des ML-Klassifikators. Den dazugehörigen Code für die Parameterschätzung (mittels *k-means* Algorithmus) findet man in der `MLKMeansEstimator`-

[1] Ein nicht unerheblicher Teil der Implementierungsarbeit entfiel auf die Klassen, welche für das Einlesen und Anzeigen der zu segmentierenden Bilder zuständig sind. Die Bedienung der Segmentieralgorithmen erfolgt über eine graphische Oberfläche. Der Aufbau dieses Programmteils wird hier nicht beschrieben.

[2] *Markerinterfaces* definieren in der Regel keine Methoden.

Klasse. Weiters wurden noch zwei, miteinander eng verwandte, *Cluster*verfahren implementiert: das ML-*Cluster*verfahren (`MLKMeansClusterer`) – ebenfalls mittels *k-means* Algorithmus – und das Maximum *a posteriori Cluster*verfahren (`MAP-KMeansClusterer`).

Die Reihenfolge der Algorithmen ist die gleiche wie in Abschnitt 4, d. h. wir sehen uns zuerst die Implementierung des ML-Klassifikators an — der eigentlich ein *Cluster*-Analyseverfahren ist — und danach betrachten wir die beiden Klassen des „wirklichen" ML-Klassifikators: den Parameterschätzer `MLKMeansEstimator` und `MLClassifier`. Zum Schluß beschreiben wir noch die Implementierung des MAP-Klassifikators in der Klasse `MAPKMeansClusterer`.

Die Beschreibung der Implementierung erfolgt durch den Code des Programmes, unterbrochen durch Kommentare. Der Kommentar bezieht sich dabei immer auf das unmittelbar folgende Codefragment. Die angegebenen Klassen enthalten nur, den für das Funktionieren des Algorithmus, notwendigen Code. Alle Programmzeilen, die nicht zum Verständnis des Algorithmus beitragen, wie z. B. *Logging* oder Ausnahmebehandlung, wurden weggelassen.

5.1 ML-Klassifikation mittels *k-means* Algorithmus

5.1.1 `MLKMeansClusterer`

Als einziges Argument benötigt der Konstruktor dieser Klasse die Anzahl der Merkmale, in die das Bild segmentiert werden soll.

```
public class MLKMeansClusterer implements Clusterer {
    protected final int k;
    protected double[] mean;
    ... //weitere Variablendeklarationen
    public MLKMeansClusterer(int k) {
        this.k = k;
        ...    //Initialisierung der Variablen
    }
```

Die Methode `segment` ist in der Schnittstelle `Segmenter` definiert. In dieser Methode wird über die M1- und M2-STEPS iteriert. Zuvor werden die Merkmalsmittelwerte mit zufälligen Werten (im Farbbereich des Bildes `mrt`) initialisiert (`initMeans`).

```
    public void segment(Image mrt, Image segimg) {
        createSegimgOld();
        initMeans();
        do {
            m1Step(mrt, segimg);
```

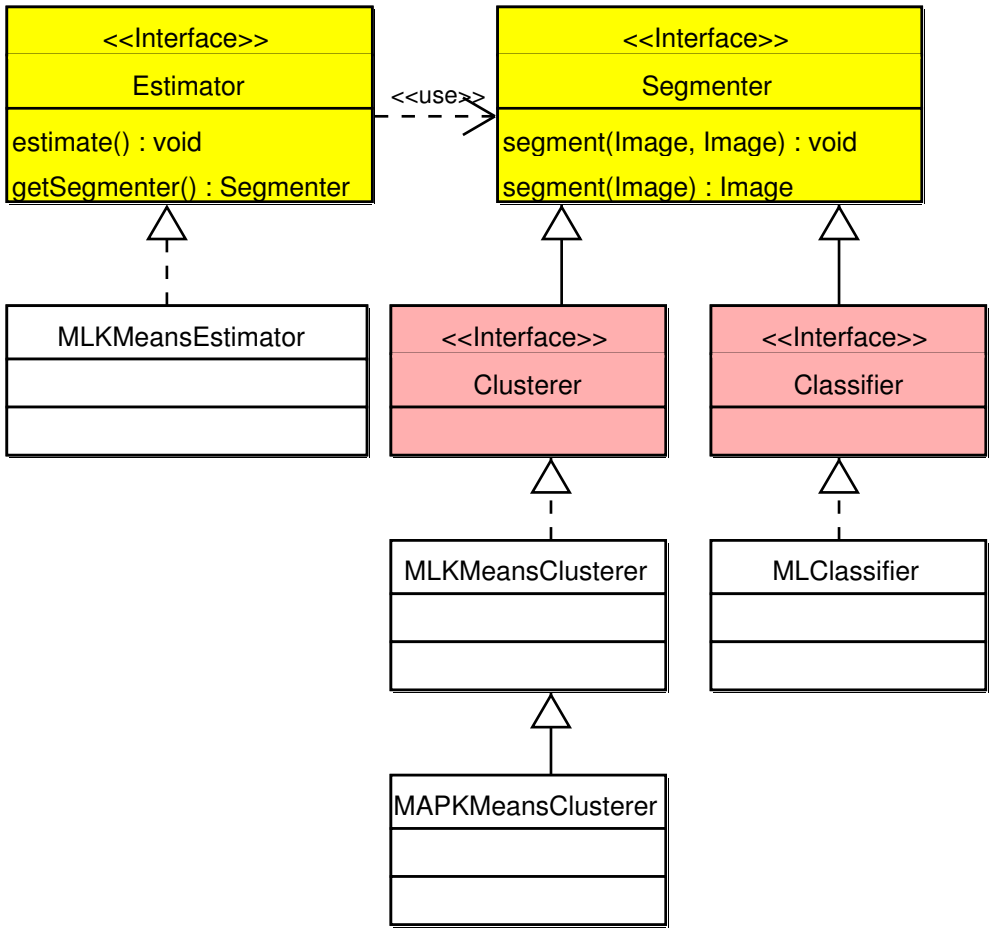


Abbildung 5.1: Diese Abbildung zeigt das UML-Diagramm der wichtigsten Klassen und Schnittstellen. Die getroffene Unterscheidung zwischen Klassifikatoren und *Cluster*-Analyseverfahren findet auch im Klassendiagramm Eingang. Es wird hier zwischen `Classifier` und `Clusterer` unterschieden. Es handelt sich dabei um sogenannte *Markerinterfaces*. Sie dienen der besseren Lesbarkeit. Diese beiden Schnittstellen wiederum erweitern die Schnittstelle `Segmenter` — da sie ja das gleiche Ziel verfolgen: die Segmentierung eines Bildes. Die zwei bzw. drei implementierten Algorithmen sind in den Klassen `MLKMeansClusterer` und `MAPKMeansClusterer` umgesetzt. Der ML-Klassifikator benötigt zwei Klassen, den Schätzer (`MLKMeansEstimator`) und den eigentlichen Klassifikator (`MLClassifier`).

```
            m2Step(mrt, segimg);
        } while (ERROR_LIMIT < error());
    }
```

In der `m1Step`-Methode werden alle Bildpunkte durchlaufen und jenem Merkmal zugeordnet, bei dem der Abstand zum Merkmalsmittelwert am geringsten ist.

```
    private void m1Step(Image mrt, Image si, Image sit) {
        ... //Einige Variablendeklarationen
        for (int i = 0, n = mrt.getNVoxels(); i < n; i++) {
            color = mrt.getColor(i);
            minDistance = Integer.MAX_VALUE;
            minDistanceFeature = 0;
```

In dieser Schleife wird für einen Bildpunkt jenes Merkmal gesucht, bei dem der Abstand des Farbwertes zum Merkmalsmittelwert am geringsten ist. Im Hinblick auf den MAP-Algorithmus wird hier zum berechneten Abstand das Cliquenpotential (bzw. Gibbs Potential) hinzuaddiert. Die Methode `getCliquenPotential` muß dafür in der abgeleiteten Klasse `MAPKMeansClusterer` nur mehr überschrieben werden.

```
            for (int f  = 0; f < k; f++) {
                distance = mean[f] - (double)color;
                distance *= distance;
                cp = getCliquesPotential(i, f);
                distance += cp;
                if (distance < minDistance) {
                    minDistanceFeature = f;
                    minDistance = distance;
                }
            }
```

Das Sichern des vorherigen Merkmals im Merkmalsbild ist in der Klasse `MAP-KMeansClusterer` notwendig, da sonst bei der Bestimmung des Cliquenpotentials auf einem sich ständig ändernden Merkmalsbild gearbeitet werden würde. Dies hätte zur Folge, daß das Cliquenpotential von der Reihenfolge abhängt, in der man die Bildpunkte durchläuft.

```
            saveOldFeatureColor(i, segimg.getColor(i));
            segimg.setColor(i, mini);
        }
    }
```

Die Methode `m2Step` berechnet aus dem zu segmentierenden Bild `mrt` und aus dem vorläufig segmentierten Bild `segimg` die neuen Merkmalsmittelwerte. Dabei werden die alten Mittelwerte gemerkt; dies ist für die Berechnung des Fehlers und in weiterer Folge für die Abbruchbedingung notwendig.

```
    private void m2Step(Image mrt, Image segimg) {
        ... //Variablendeklarationen
        for (int i = 0; i < size; i++) {
            f = segimg.getColor(i);
            meanSum[f] += mrt.getColor(i);
            meanNo[f]++;
        }
        System.arraycopy(mean, 0, meanOld, 0, mean.length);
        for (int i = 0; i < k; i++) {
            mean[i] = meanSum[i] / meanNo[i];
        }
        Arrays.sort(mean);
    }
```

Eine einfache Definition des aktuellen Fehlers. Man kann sich hier auch „ausgefeiltere" Fehlerberechnungen vorstellen. Diese einfache Methode hat sich in den Testläufen jedoch bewährt.

```
    protected double error() {
        double err = Integer.MIN_VALUE;
        for (int i = 0; i < k; i++) {
            err = Math.max(Math.abs(mean[i] - meanOld[i]), err);
        }
        return err;
    }
```

Diese Methode wird in der Klasse `MAPKMeansClusterer` überschrieben und berechnet dort das Cliquenpotential V_C. Hier liefert diese Methode für das Cliquenpotential natürlich null.

```
    protected double getCliquesPotential(int pos, int f) {
        return 0;
    }
}
```

5.1.2 MLClassifier

Diese Klasse segmentiert ein Bild mit Hilfe des Maximum-Likelihood-Klassifikators (Gleichung (4.19)). Daher erfordert der Konstruktor die Angabe der Klassenmittelwerte `mean` und die dazugehörigen Varianzen `var`.

```
public class MLClassifier implements Classifier {
    private double[] mean;
    private double[] var;
    private double[] dev;
```

```
private int k;
public MLClassifier(double[] mean, double[] var) {
    k = mean.length;
    mean = new double[k];
    var = new double[k];
    dev = new double[k];
    System.arraycopy(mean, 0, this.mean, 0, k);
    System.arraycopy(var, 0, this.var, 0, k);
     for (int i = 0; i < k; i++) {
        dev[i] = Math.sqrt(var[i]);
    }
}
```

Die Segmentiermethode `segment` ordnet jedem Bildpunkt, mit Hilfe der Entscheidungsregel `e`, eine Klasse zu.

```
public void segment(Image mrt, Image segimg) {
   for (int i = 0, n = mrt.getNVoxels(); i < n; i++) {
        segimg.setColor(i, e(mrt.getColor(i)));
    }
}
```

Die Entscheidungsregel `e` ist beim ML-Klassifikator so implementiert, daß einem Bildpunkt jene Klasse zugeordnet wird, bei der die klassenbedingte Wahrscheinlichkeitsdichte $p(c \mid \omega)$ maximal ist.

```
private int e(int color) {
    double prop;
    double maxProp = -Double.MAX_VALUE;
    for (int i = 0; i < k; i++) {
        prop = p(color, i);
        if(prop > propMax) {
            maxFeatureIndex = i;
            maxProp = prop;
        }
    }
    return maxFeatureIndex;
}
```

Die Methode `p` ist die Umsetzung der Wahrscheinlichkeitsdichte

$$p(c \mid \omega) = \frac{1}{\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{5.1}$$

unter Normalverteilungsannahme. Dabei wurde der konstante Term $\frac{1}{\sqrt{2\pi}}$ weggelassen, da dieser die Lage des Maximums nicht verändert.

```
    private double p(double color, int f) {
        return Math.exp(-((color-mean[f])*color-mean[f])/
                        (2*var[f])) / dev[f];
    }

}
```

5.1.3 MLKMeansEstimator

Für die Segmentierung benötigt die `MLClassifier`-Klasse den Mittelwert und die Varianz jeder einzelnen Klasse. Diese müssen aber erst geschätzt werden. Die Parameterschätzung übernimmt der `MLKMeansEstimator`; wie der Name vermuten läßt passiert dies mit Hilfe des *k-means* Algorithmus.

Der `MLKMeansEstimator` erwartet als Parameter das zu segmentierende Bild `mrt` und die Anzahl der Äquvalenzklassen `k`.

```
public class MLKMeansEstimator implements Estimator {
    private Image mrt;
    private int k;

    private double[] colorSum;
    private double[] colorSqrSum;
    private double[] colorNo;

    ... //Weitere Variablendeklarationen

    public MLKMeansEstimator(Image mrt, int k) {
        this.mrt = mrt;
        this.k = k;
    }
```

Die Methode `estimate`, diese ist in der Schnittstelle `Estimator` definiert, startet den Schätzvorgang.

```
    public void estimate() {
         initMeans();
        do {
            m1Step(mrt);
            m2Step();
        } while (ERROR_LIMIT < error());
        calculateVariance();
    }
```

Die `m1Step`-Methode unterscheidet sich von der in der Klasse `MLKMeansClusterer` nur in den letzten paar Zeilen. (Der gemeinsame Code aus den Klassen `MLKMeansEstimator` und `MLKMeansClusterer` ist in einer kleineren Schriftgröße gesetzt.)

```
    private void m1Step(Image mrt) {
        ...
        for (int i = 0, n = mrt.getNVoxels(); i < n; i++) {

             color = mrt.getColor(i);
             minDistance = Double.MAX_VALUE;
             minDistanceFeature = 0;

             for (int f  = 0; f < k; f++) {
                 distance = mean[f] - (double)color;
                 distance *= distance;
                 cp = getCliquesPotential(i, f);
                 distance += cp;
                 if (distance < minDistance) {
                     minDistanceFeature = f;
                     minDistance = distance;
                 }
             }

            colorSum[minDistanceFeature] += color;
            colrSqrSum[minDistanceFeature] += (color*color);
            colorNo[minDistanceFeature]++;
        }
    }
```

Die Methode `m2Step` ist der in der Klasse `MLKMeansClusterer` ebenfalls sehr ähnlich.

```
    private void m2Step(Image mrt) {
        System.arraycopy(mean, 0, meanOld, 0, k);
        for (int i = 0; i < k; i++) {
            mean[i] = colorSum[i] / colorNo[i];
        }
        Arrays.sort(mean);
    }
```

Die Varianz wird gemäß Gleichung (4.42) berechnet.

```
    private void calculateVariance() {
        for (int i = 0; i < k; i++) {
            var[i] = (colorSum[i] -
                      2*mean[i]*colorSum[i] +
                      colorNo[i]*mean[i]*mean[i]) /
                          (colorNo[i]-1.0);
         }
    }
```

Das Ergebnis der Schätzung ist eine `Segmenter`-Klasse (nämlich `MLClassifiers`).

```
    public Segmenter getSegmenter() {
        return new MLClassifier(mean, var);
    }
}
```

5.2 Maximum *a posteriori* Klassifikator

5.2.1 `MAPKMeansClusterer`

Durch die, in Abschnitt 4.6 beschriebene, Ähnlichkeit des `MLKMeansClusterer` mit dem `MAPKMeansClusterer`, ist diese Klasse auch konsequenterweise davon abgeleitet. Das bringt den Vorteil mit sich, daß bei der Implementierung nur einige wenige Methoden überschrieben werden müssen.

```
public class MAPKMeansClusterer extends MLKMeansClusterer {
    private double beta = 0.04;
    private double betaSqrt = Math.sqrt(beta);
    ...
    private int[] n6 = new int[6];
    private int[] n12 = new int[12];

    public MAPKMeansClusterer(int k) {
        super(k);
    }

    protected void createSegimgOld(Image segimg) {
        segimgOld = (Image)segimg.clone();
        size = segimgOld.getNVoxels();
    }

    protected void saveOldFeatureColor(int pos, int color) {
        segimgOld.setColor(pos, color);
    }
```

Dabei ist die Methode, welche das Cliquenpotential V_C (im Programm mit `Vc` bezeichnet) berechnet, die wichtigste. Man sieht, daß die Berechnung der Nachbarschaft eines Bildpunktes vom `Image`-Objekt übernommen wird; dafür ist die Kenntnis der internen Speicherstruktur notwendig. Die Parameter `beta` bzw. `betaSqrt` entsprechen dem β und $\frac{\beta}{\sqrt{2}}$ in Gleichung (4.45) des Cliquenpotentials. Für die Bestimmung des passenden Wertes für `BETA` siehe Abschnitt 6.3 auf Seite 109.

```
    protected double getCliquesPotential(int pos, int f) {
        double Vc = 0.0;

        segimgOld.getNeighbor3D6Positions(pos, n6);
        segimgOld.getNeighbor3D12Positions(pos, n12);

         for (int i = 0; i < 6; i++) {
            if (n6[i] >= 0 && n6[i] < size) {
                if (segimgOld.getColor(n6[i]) != f) {
                    Vc += beta;
```

```
                    }
                }
            }
            for (int i = 0; i < 12; i++) {
                if (n12[i] >= 0 && n12[i] < size) {
                    if (segimgOld.getColor(n12[i]) != f) {
                        Vc += betaSqrt;
                    }
                }
            }
            return Vc*colorRange.getNColors():
        }
    }
```

Die soeben beschriebenen Erweiterungen reichen für die Implementierung des MAP-Algorithmus aus.

Kapitel 6

Ergebnisse

6.1 MR-Simulator

Für die Bewertung der Segmentierergebnisse wurden künstlich erzeugte MR-Datensätze aus *„BrainWeb*: Simulated Brain Database"[1] verwendet [10], [11]. Momentan beinhaltet diese Datenbank zwei anatomische Modelle:

1. Modell eines normalen, gesunden Gehirns und

2. Modell eines *Multiple Sklerose* geschädigten Gehirns.

Für beide Modelle sind 3D-Datensätze in T_1-, T_2- und PD-Modalität (*Proton Density*) verfügbar (mit jeweils unterschiedlichen Rauschanteilen von 3 %, 5 %, 7 % und 9 %). Bei der Validierung der implementierten Segmentiermethoden liegt unser Hauptaugenmerk auf der Erkennungsleistung, d. h. der Gesamtfehlerrate (siehe Definition 4.7.1). Eine geringere Rolle soll die Ausführungsgeschwindigkeit spielen.

Der MR-Simulator des *Brain Web* erzeugt die 3D-Datensätze mit Hilfe eines diskreten anatomischen Modells. In diesem Modell ist jedem Bildpunkt eine Gewebeart zugeordnet. In Tabelle 6.1 sind die Gewebetypen und die dazugehörige Anzahl der Bildpunkte aufgelistet, aus denen sich das anatomische Modell zusammensetzt. Für die Validierung der Segmentierergebnisse wurde das von *Brain Web* verwendete anatomische Modell als Referenz benutzt.

[1] BrainWeb: `http://www.bic.mni.mcgill.ca/brainweb/`. Die Generierung der Datensätze erfolgt mit Hilfe eines sogenannten MR-SIMULATORS.

Tabelle 6.1: Das für die Simulation verwendete anatomische Modell (gesundes Gehirn) setzt sich aus 9 Gewebetypen plus Hintergrund zusammen. Die Tabelle zeigt die Gewebetypen und die Anzahl der dazugehörigen Bildpunkte.

Nr.	Gewebetyp	#Bildpunkte	Nr.	Gewebetyp	#Bildpunkte
0	Hintergrund	3.001.960	5	Muskel/Haut	617.482
1	CSF	371.945	6	Haut	726.649
2	Graue Masse	902.912	7	Schädelknochen	362.561
3	Weiße Masse	674.777	8	Glia	5.987
4	Fett	146.514	9	Bindegewebe	298.350

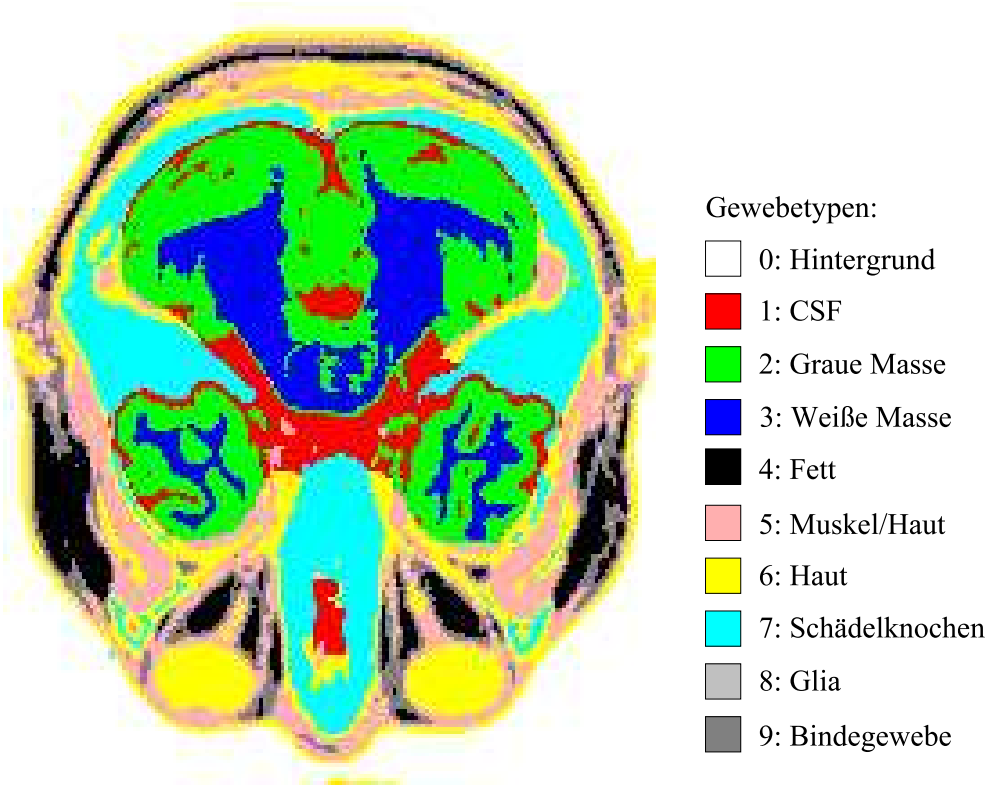


Abbildung 6.1: Diese Abbildung zeigt das für die Validierung verwendete (anatomische) Modell. Dies ist auch jenes Modell, welches der MR-Simulator für die Erzeugung der MR-Bilder verwendet. Es setzt sich aus zehn Merkmalen zusammen. Dargestellt ist hier die Schicht 35.

6.2 Validierung des ML-Algorithmus

6.2.1 *A posteriori* Validierung

Aus der fest vorgegebenen Anzahl der Gewebetypen ergibt sich ein Problem für die Validierung: die Anzahl der Merkmale k muß bei den verwendeten Segmentieralgorithmen *a priori* bekannt sein. Da jedoch der Parameter k in der Regel nicht mit der Anzahl der Gewebetypen K übereinstimmt, ist für die Berechnung der Fehlerrate eine Abbildung $T_3 : \{0,\ldots,k-1\} \rightarrow \{0,\ldots,K-1\}$ notwendig, welche die *abstrakten* Merkmale des Segmentierergebnisses auf die anatomischen Merkmale (Gewebetypen) des Modells abbildet. (Von der Art der Berechnung der Transformation T_3 – Gleichung (6.1) – leitet sich der Name dieser Validierungsart ab.) In Abbildung 6.2 ist der gesamte Ablauf, von der Erzeugung des MR-Bildes

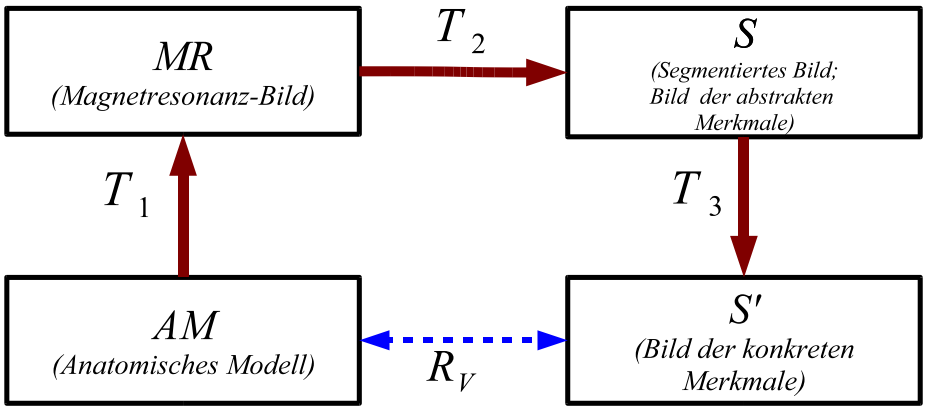


Abbildung 6.2: Ausgehend vom anatomischen Modell erzeugt der MR-Simulator ein MR-Bild (Transformation T_1). Dieses Bild wird vom Segmentieralgorithmus in abstrakte Merkmale zerlegt (Transformation T_2). Für eine Validierung ist es jedoch notwendig die abstrakten Merkmale auf die konkreten Merkmale (Gewebetypen) des anatomischen Modells abzubilden (Transformation T_3). Erst jetzt kann die Validierung des Ergebnisses vorgenommen werden (Validierungsrelation R_V).

über die Segmentierung und die Validierung, zusammengefaßt.

Transformation T_3 Die Transformation T_3 muß so gewählt werden, daß bei der darauf folgenden Validierung der gesamte Klassifikationsfehler (4.56) minimiert bzw. die gesamte Klassifikationsgenauigkeit (4.57) maximiert wird.

Betrachtet man T_3 als Entscheidungsregel des Maximum-Likelihood-Klassifikators (4.19), so können wir T_3 folgendermaßen schreiben:

$$T_3(j) := \text{argmax}_{i\in\{0,\ldots,K-1\}}\{p(j \mid i)\}, \quad j \in \{0,\ldots,k-1\}. \tag{6.1}$$

Mit

$$p(j \mid i) = \frac{p(j \cap i)}{p(i)}$$

ergibt sich

$$T_3(j) := \text{argmax}_{i\in\{0,\ldots,K-1\}}\left\{\frac{p(j \cap i)}{p(i)}\right\}, \tag{6.2}$$

wobei $p(j \cap i)$ die Wahrscheinlichkeit bezeichnet, daß Gewebe i als das abstrakte Merkmal j segmentiert wurde. Für die Transformation T_3 müssen die Wahrscheinlichkeiten $p(j \cap i)$ und $p(i)$ nicht explizit berechnet werden. Es genügt, wenn wir die absoluten Häufigkeiten $h_{i,j}$, welche wir in einer Matrix H mit K Zeilen und k Spalten speichern, ermitteln. Die Werte $h_{i,j}$ geben an, wie oft dem Gewebetyp i des anatomischen Modells das abstrakte Merkmal j des segmentierten Bildes (für gleiche Bildkoordinaten) zugeordnet wurde. (Algorithmus 11 zeigt die Berechnung von H.) Nun sind wir auch schon fertig. Da

$$h_{i,j} \propto p(j \cap i) \tag{6.3}$$

und

$$\sum_{j=0}^{k-1} h_{i,j} \propto p(i), \tag{6.4}$$

folgt

$$T_3(j) := \text{argmax}_{i\in\{0,\ldots,K-1\}}\left\{\frac{h_{i,j}}{\sum_{j=0}^{k-1} h_{i,j}}\right\}. \tag{6.5}$$

Mit der Transformation T_3 im Gepäck, können wir uns nun daran machen eine Bewertungsart aus Definition 4.7.1 auszuwählen und auf unsere Segmentierergebnisse anzuwenden. Da wir uns bei der Validierung auf kein bestimmtes Merkmal festlegen wollen, ist es am besten den gesamten Klassifikationsfehler bzw. die Gesamtfehlerrate $F_{\mathcal{K},M}$ (Gleichung (4.56)) der Segmentierung zu bestimmen. Algorithmus 12 zeigt die Berechnung der Gesamtfehlerrate mit Hilfe des anatomischen Modells AM, des segmentierten Bildes S und der Transformation T_3.

6.2.2 *A priori* Validierung

Die *a posteriori* Validierung ist nicht die einzige Möglichkeit das anatomische Modell und das segmentierte Bild miteinander vergleichbar zu machen. Bei der *a*

Algorithm 11 Berechnung der Matrix H. Dabei bezeichnet $I_1(p)$ den Gewebetyp des anatomischen Modells und $I_2(p)$ das abstrakte Merkmal des segmentierten Bildes, für einen Bildpunkte p.

1. Initialisieren der Matrixelemente $h_{i,j}$ mit null.
2. Für alle $p \in I_D$
   (a) $i \leftarrow I_1(p)$
   (b) $j \leftarrow I_2(p)$
   (c) $h_{i,j} \leftarrow h_{i,j} + 1$

Algorithm 12 Berechnung der Gesamtfehlerrate mit Hilfe des anatomischen Modells AM, des segmentierten Bildes S und der Transformation T_3. Der Bildbereich I_D ist für beide Bilder gleich.

1. Für alle $p \in I_D$
   (a) $a \leftarrow AM(p)$
   (b) $s \leftarrow S(p)$
   (c) Wenn $a \neq T_3(s)$
      i. $n \leftarrow n + 1$
2. $F_{\mathcal{K},M} \leftarrow \frac{n}{|I_D|}$

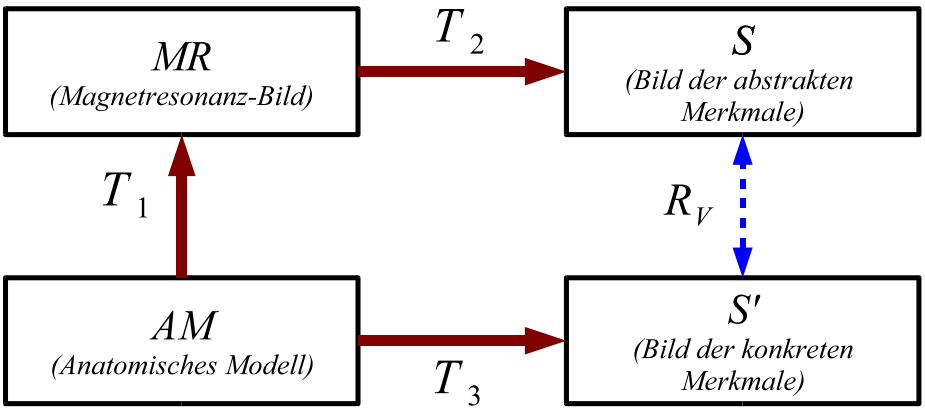


Abbildung 6.3: Übersicht der durchgeführten Transformationen bei der *a priori* Validierung.

posteriori Validierung wird das segmentierte Bild dem anatomischen Modell angepaßt, bei der *a priori* Validierung passiert die Anpassung in umgekehrter Richtung. Der Unterschied zwischen den beiden Validierungsarten liegt somit in der Berechnung (bzw. „Richtung") der Transformation T_3 (siehe Abbildung 6.3). Für $T_3 : \{0,...,K-1\} \rightarrow \{0,...,k-1\}$ folgt daher:

$$T_3(j) := \text{argmax}_{i\in\{0,...,k-1\}} \{p(i \mid j)\}, \tag{6.6}$$

$$:= \text{argmax}_{i\in\{0,...,k-1\}} \left\{ \frac{p(i \cap j)}{p(j)} \right\}, \tag{6.7}$$

mit $j \in \{0,...,K-1\}$. Die konkrete Berechnung von T_3 erfolgt auf ähnliche Weise wie bei der *a posteriori* Validierung. Wiederum ist

$$h_{i,j} \propto p(i \cap j)$$

und

$$\sum_{i=0}^{k-1} h_{i,j} \propto p(j). \tag{6.8}$$

Daraus ergibt sich für T_3:

$$T_3(j) := \text{argmax}_{i\in\{0,...,k-1\}} \left\{ \frac{h_{i,j}}{\sum_{i=0}^{k-1} h_{i,j}} \right\}. \tag{6.9}$$

Auch die Berechnung der Gesamtfehlerrate ist sehr ähnlich. In Algorithmus 12 muß lediglich der Vergleich $a \neq T_3(s)$ (in Zeile 1. (c)) gegen $s \neq T_3(a)$ ausgetauscht werden.

6.2.3 Ergebnisse

Der einzige Parameter der sich beim ML-Algorithmus variieren läßt, ist die Anzahl der zu segmentierenden (abstrakten) Merkmale k. Durch die „zufällige" Initialisierung des Algorithmus erhalten wir für identische Startparameter in der Regel *keine* identischen Segmentierergebnisse. Für eine statistische Auswertung der Ergebnisse wurde eine Segmentierung (mit dem gleichen Parameter k) 50 mal durchgeführt. Der Parameter k wurde im Bereich von 2 bis 20 variiert. (Die Berechnung der Gesamtfehlerrate erfolgte mittels der *a posteriori* Validierung.) Bei den verwendeten Bildern handelt es sich um T_1 gewichtete MR-Aufnahmen eines gesunden Gehirns. Die Abbildungen 6.4 bis 6.7 zeigen die Gesamtfehlerrate in Abhängigkeit der abstrakten Merkmale k.

Je höher der Rauschanteil der Bilder, desto schlechter wird auch das Segmentierergebnis, d. h. desto höher fällt die Gesamtfehlerrate aus. Dieses Ergebnis bestätigt die „Anfälligkeit" des ML-Algorithmus gegenüber verrauschten Eingabebildern. Der MAP-Algorithmus löst dieses Problem, jedoch zum Preis eines erhöhten Rechenaufwandes. Die Versuche zeigten außerdem sehr eindeutig, daß die optimale Anzahl von Merkmalen bei sechs liegt.[2]

Die Abbildungen 6.8 bis 6.11 zeigen typische Segmentierergebnisse der MR-Bilder mit den unterschiedlichen Rauschanteilen. Die Anzahl der abstrakten Merkmale wurde dabei auf sechs gesetzt. Bei diesem Wert ist die Gesamtfehlerrate für die untersuchten Bilder minimal.

Neben der Gesamtfehlerrate ist auch noch interessant, wie sich die Anzahl der benötigten Iterationen in Abhängigkeit von k verändert. Abbildung 6.12 zeigt die mittlere Anzahl der benötigten Iterationen in Abhängigkeit der abstrakten Merkmale k. Der Mittelwert der Iterationen und die Standardabweichung wurde aus jeweils 50 Durchläufen ermittelt.

6.3 Validierung des MAP-Algorithmus

Bei den bisher behandelten Validierungsarten (*a posteriori* Validierung und *a priori* Validierung) wurden die segmentierten Merkmale der einzelnen Bildpunkte als unabhängig von den Nachbarmerkmalen betrachtet. Für die Validierung des

[2] Dieses Ergebnis kann jedoch **nicht** verallgemeinert werden! Es bezieht sich explizit auf die hier verwendeten Datensätze.

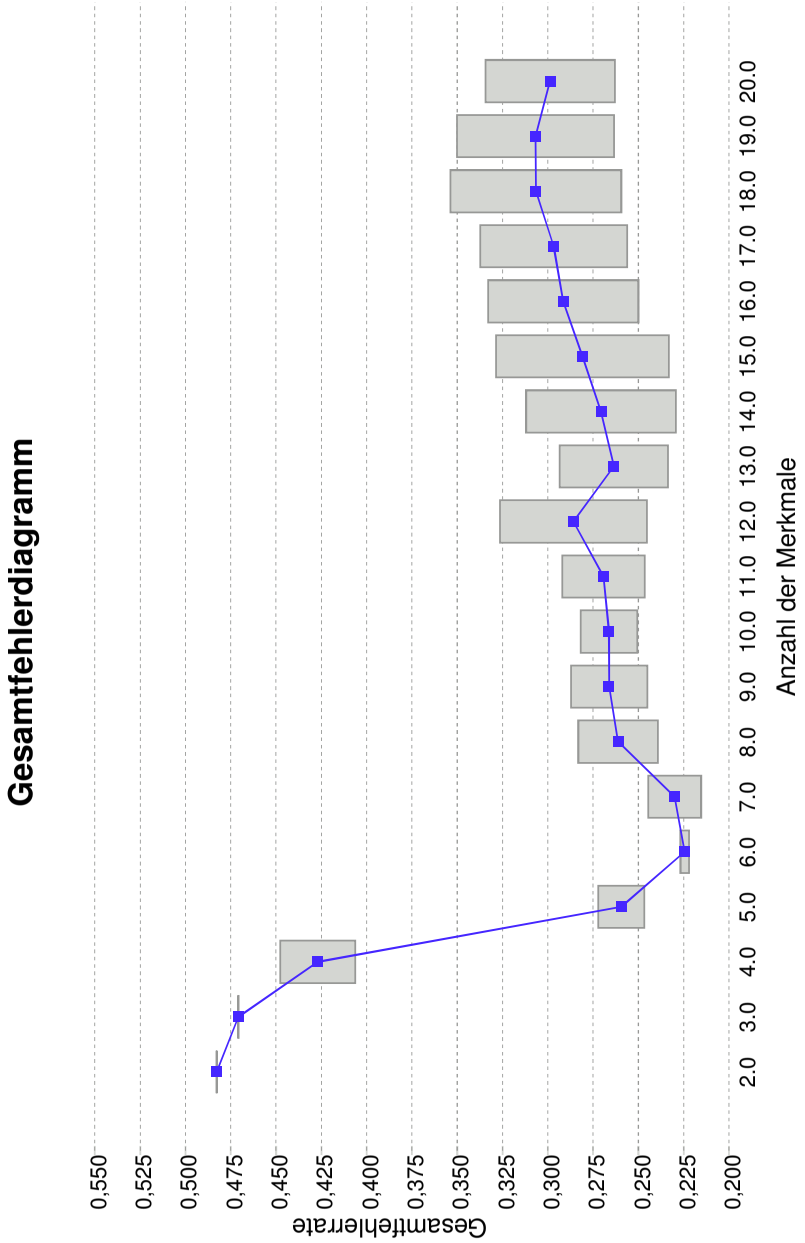


Abbildung 6.4: Mittelwert und Standardabweichung der Gesamtfehlerrate (50 Durchläufe) für ein T_1 gewichtetes MR-Bild mit einem Rauschanteil von 3 %.

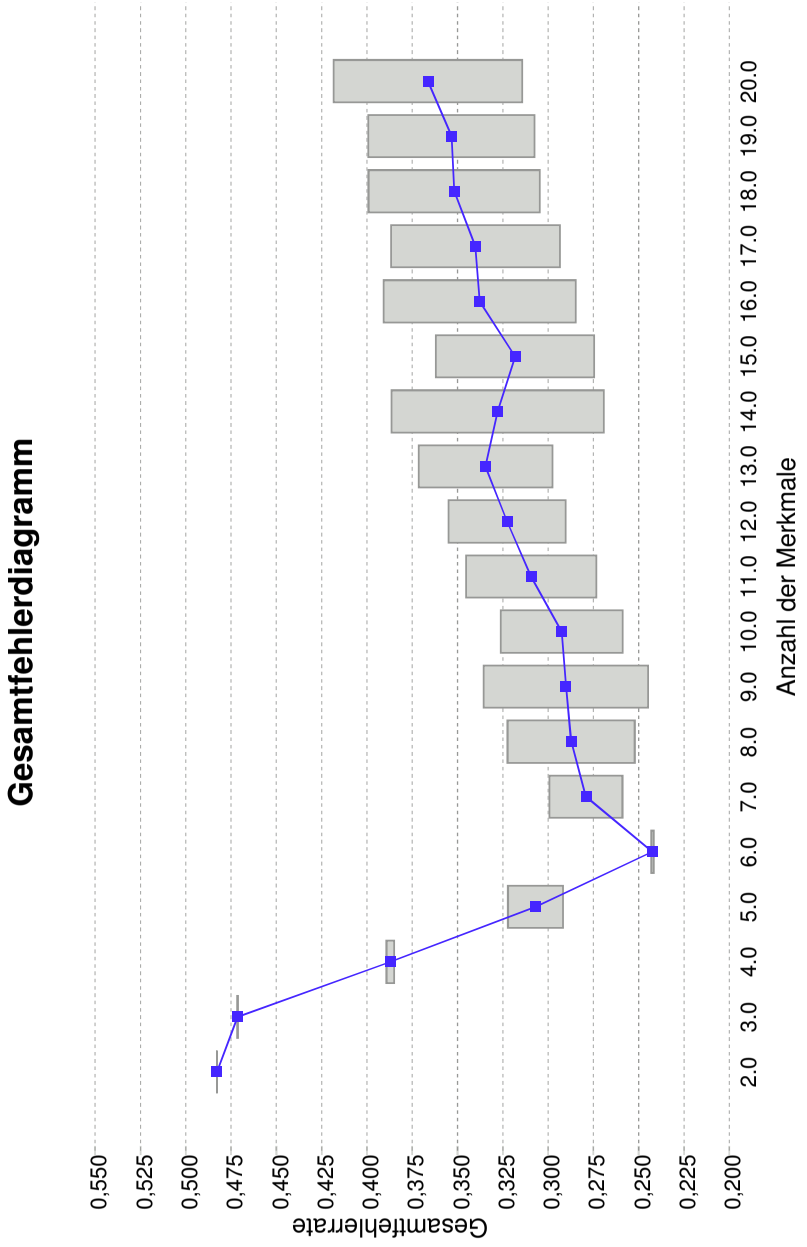


Abbildung 6.5: Mittelwert und Standardabweichung der Gesamtfehlerrate (50 Durchläufe) für ein T_1 gewichtetes MR-Bild mit einem Rauschanteil von 5 %.

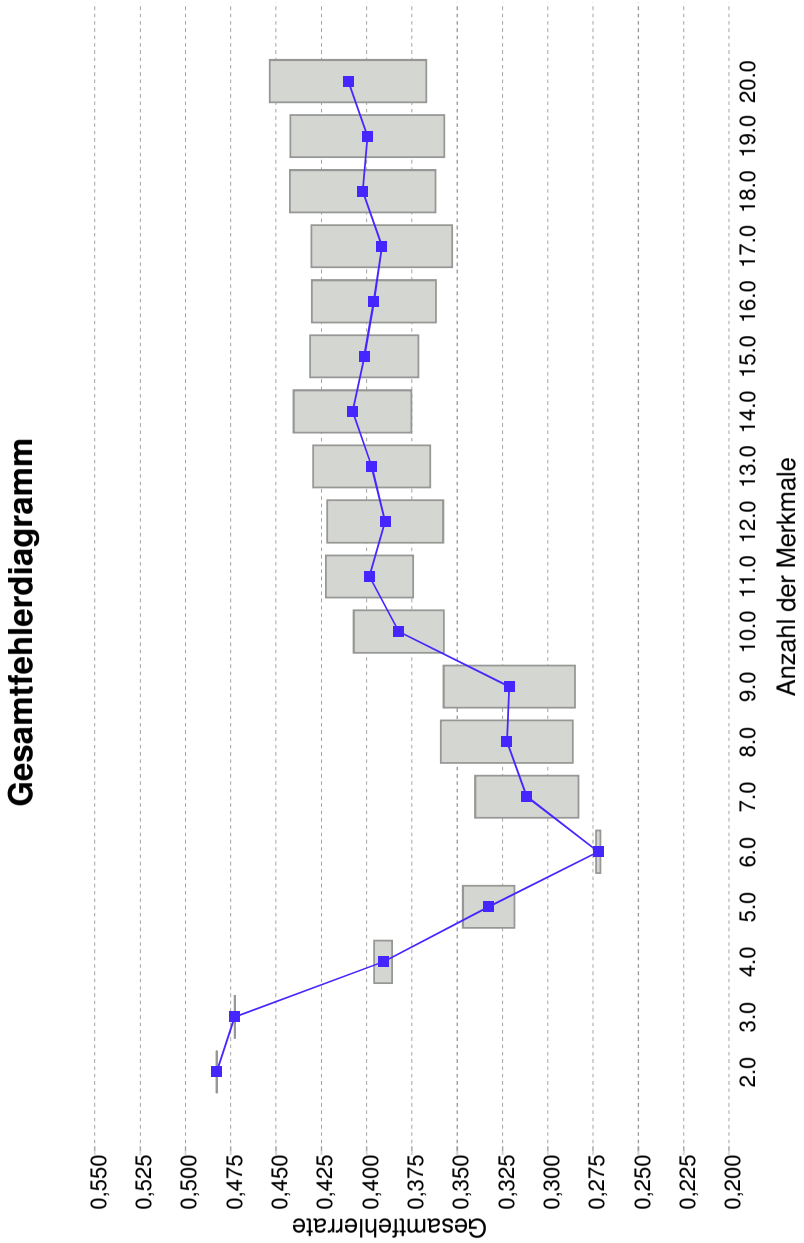


Abbildung 6.6: Mittelwert und Standardabweichung der Gesamtfehlerrate (50 Durchläufe) für ein T_1 gewichtetes MR-Bild mit einem Rauschanteil von 7 %.

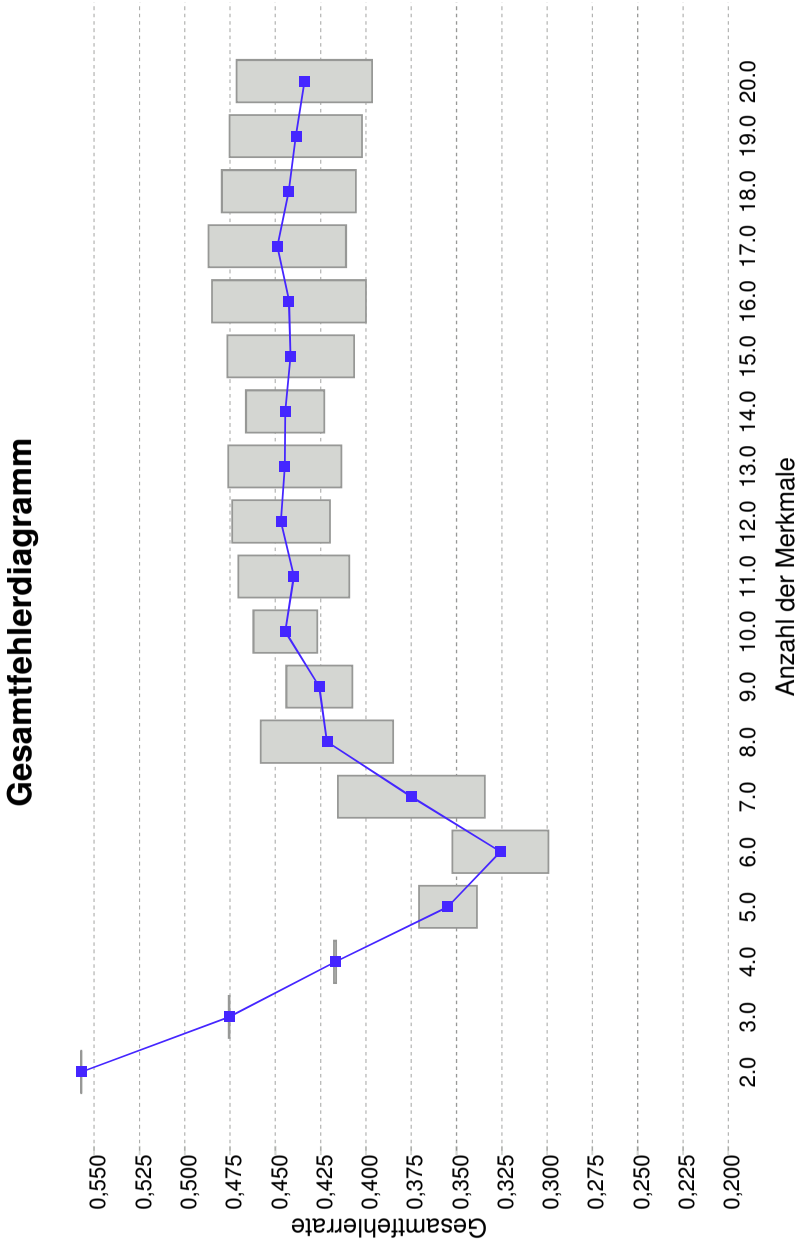


Abbildung 6.7: Mittelwert und Standardabweichung der Gesamtfehlerrate (50 Durchläufe) für ein T_1 gewichtetes MR-Bild mit einem Rauschanteil von 9 %.

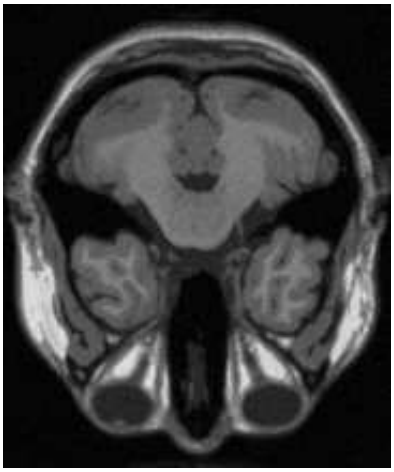
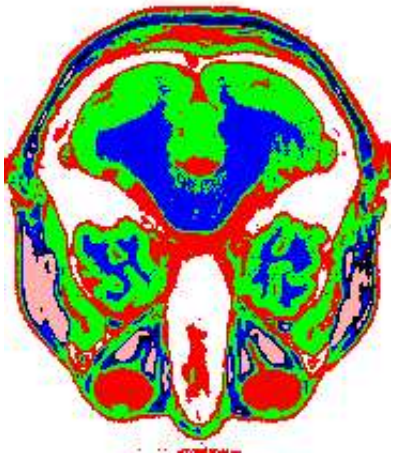
(a)  (b) 

Abbildung 6.8: Bild (a) zeigt Schicht 35 des segmentierten Datensatzes mit einem Rauschanteil von 3 %. Bild (b) zeigt das Segmentierergebnis des ML-Algorithmus mit $k = 6$ abstrakten Merkmalen.

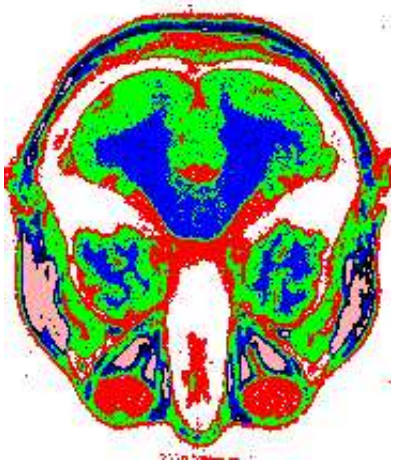
(a)  (b) 

Abbildung 6.9: Bild (a) zeigt Schicht 35 des segmentierten Datensatzes mit einem Rauschanteil von 5 %. Bild (b) zeigt das Segmentierergebnis des ML-Algorithmus mit $k = 6$ abstrakten Merkmalen.

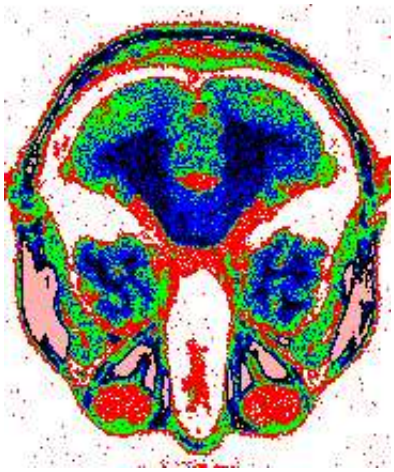
(a)  (b) 

Abbildung 6.10: Bild (a) zeigt Schicht 35 des segmentierten Datensatzes mit einem Rauschanteil von 7 %. Bild (b) zeigt das Segmentierergebnis des ML-Algorithmus mit $k = 6$ abstrakten Merkmalen.

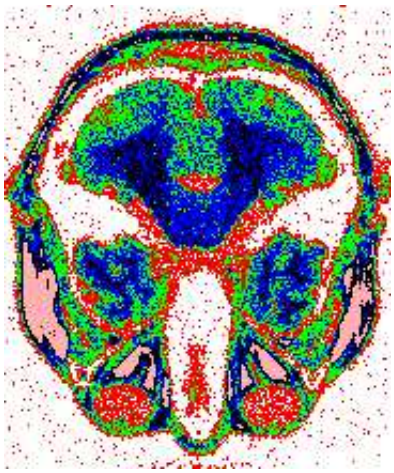
(a)  (b) 

Abbildung 6.11: Bild (a) zeigt Schicht 35 des segmentierten Datensatzes mit einem Rauschanteil von 9 %. Bild (b) zeigt das Segmentierergebnis des ML-Algorithmus mit $k = 6$ abstrakten Merkmalen.

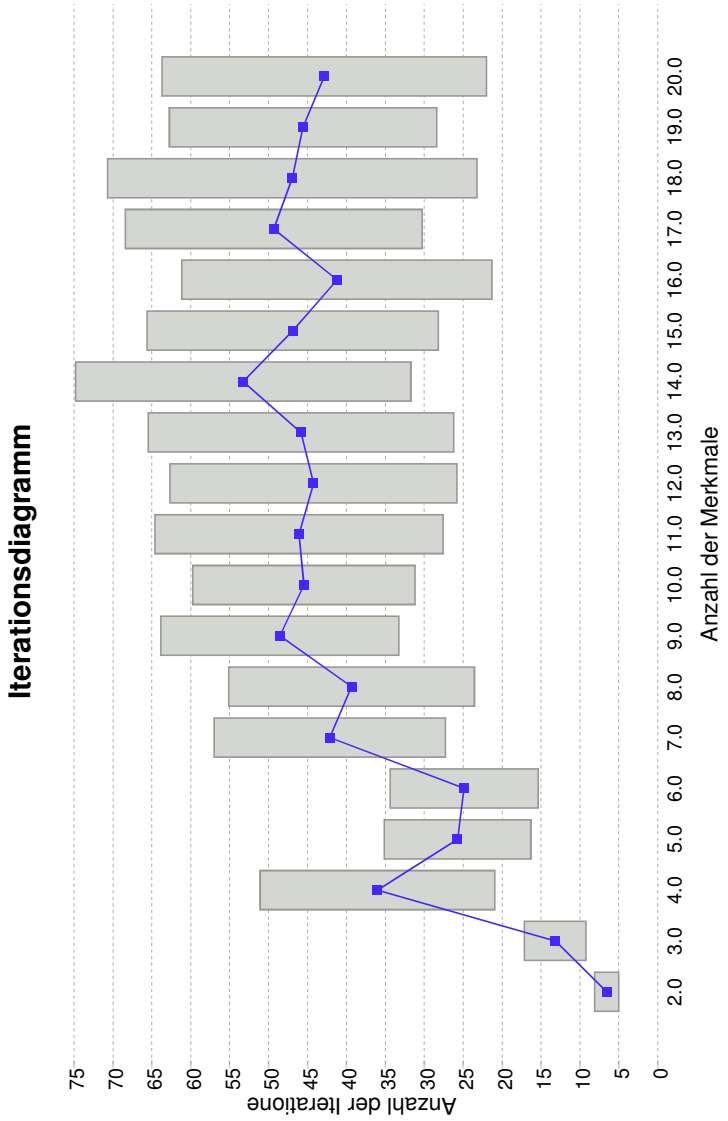


Abbildung 6.12: Anzahl der benötigten Iterationen in Abhängigkeit von k. Auf der Ordinate ist die mittlere Anzahl der Iterationen und die Standardabweichung (50 Durchläufe) aufgetragen. Segmentiert wurde das MR-Bild mit einem Rauschanteil von 3 %. Die Bilder mit den Rauschanteilen von 5 %, 7 % und 9 % zeigen einen ähnlichen Anstieg der benötigten Iterationen bei wachsendem k.

MAP-Algorithmus ist dies jedoch nicht ausreichend. Um die Qualität des MAP-Algorithmus bestimmen zu können, fließt in die Bewertung des Segmentierergebnisses die Nachbarschaft eines Bildpunktes mit ein. Die Berücksichtigung der Nachbarschaft wird durch die Berechnung des Varianzbildes (Abschnitt 6.3.1) realisiert. Hat man das Varianzbild des anatomischen Modells und des segmentierten, abstrakten Merkmalsbildes berechnet, so können diese mittels *Mutual Information* (Abschnitt 6.3.2) miteinander verglichen werden. Dabei bedeutet ein hoher *Mutual Information* Wert eine gute Übereinstimmung der beiden Bilder. (Bei einem Wert von eins sind die beiden Bilder miteinander identisch.)

6.3.1 Varianzbild

Für die Berechnung des Varianzbildes betrachten wir die 26er-Nachbarschaft N_{26} eines Bildpunktes $p \in F_D$. (Im weiteren wird mit F das segmentierte Bild und mit F_D der dazugehörige Bildbereich bezeichnet. Mit $F(p)$ bezeichnen wir außerdem das Merkmal des Bildpunktes p.) Um das Varianzbild nicht von der „Numerierung" der Merkmale abhängig zu machen, definieren wir die Funktion $f_p : F_D \rightarrow \{0,1\}$:

$$f_p(q) := \begin{cases} 1 & \text{wenn} \quad F(p) = F(q), \\ 0 & \text{sonst.} \end{cases} \tag{6.10}$$

Mit Hilfe dieser Funktion berechnen wir nun den Mittelwert

$$\mu_p = \frac{1}{|N_{26}(p) \cup \{p\}|} \cdot \sum_{q \in N_{26} \cup \{p\}} f_p(q) \tag{6.11}$$

und die Varianz

$$\sigma_p^2 = \frac{1}{|N_{26}(p) \cup \{p\}| - 1} \cdot \sum_{q \in N_{26} \cup \{p\}} (f_p(q) - \mu_p)^2 \tag{6.12}$$

eines Bildpunktes p.

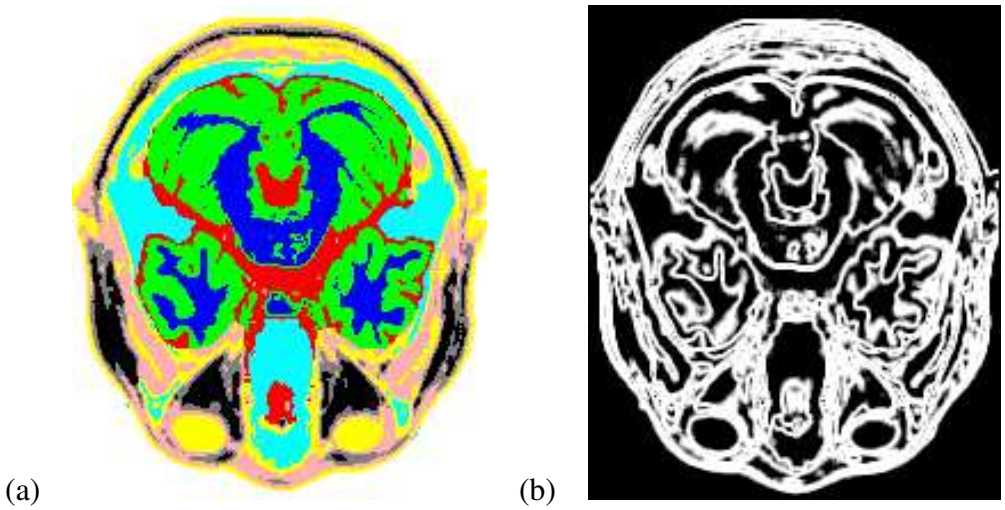
Für die konkrete Berechnung des Varianzbildes reicht es aus, die Summe

$$S_p := \sum_{q \in N_{26} \cup \{p\}} f_p(q) \tag{6.13}$$

für einen Punkt p zu bestimmen und die dazugehörige Varianz aus einer, zuvor berechneten, Tabelle zu entnehmen (Tabelle 6.2). Für die Darstellung des Bildes — und dessen Weiterverarbeitung — wird nun nicht die Varianz gespeichert, sondern eine eindeutige Kennzahl N_{σ_p}, welche proportional zur Varianz ist. Abbildung 6.13 zeigt das verwendete Modell und das daraus berechnete Varianzbild. (Dabei wurde für die Darstellung der Bereich von N_{σ_p} auf einen Grauwertbereich von 0 bis 255 gestreckt.) Abbildung 6.14 zeigt die Segmentierergebnisse von MR-Bilder mit unterschiedlichen Rauschanteilen und die dazugehörigen Varianzbilder.

Tabelle 6.2: Tabelle der vorberechneten Varianzen und die dazugehörigen Kennzahlen N_{σ_p}.

S_p	σ_p^2	μ_p	N_{σ_p}	S_p	σ_p^2	μ_p	N_{σ_p}	S_p	σ_p^2	μ_p	N_{σ_p}
1	0,963	$\frac{1}{27}$	1	10	6,296	$\frac{10}{27}$	10	19	5,630	$\frac{19}{27}$	8
2	1,852	$\frac{2}{27}$	2	11	6,519	$\frac{10}{27}$	11	20	5,185	$\frac{20}{27}$	7
3	2,667	$\frac{3}{27}$	3	12	6,667	$\frac{12}{27}$	12	21	4,667	$\frac{21}{27}$	6
4	3,407	$\frac{4}{27}$	4	13	6,740	$\frac{13}{27}$	13	22	4,074	$\frac{22}{27}$	5
5	4,074	$\frac{5}{27}$	5	14	6,740	$\frac{14}{27}$	13	23	3,407	$\frac{23}{27}$	4
6	4,667	$\frac{6}{27}$	6	15	6,667	$\frac{15}{27}$	12	24	2,667	$\frac{24}{27}$	3
7	5,185	$\frac{7}{27}$	7	16	6,519	$\frac{16}{27}$	11	25	1,852	$\frac{25}{27}$	2
8	5,630	$\frac{8}{27}$	8	17	6,296	$\frac{17}{27}$	10	26	0,963	$\frac{26}{27}$	1
9	6	$\frac{9}{27}$	9	18	6	$\frac{18}{27}$	9	27	0	$\frac{27}{27}$	0



(a) (b)

Abbildung 6.13: Bild (b) zeigt das von Bild (a) — dabei handelt es sich um das anatomische Modell *AM* — erzeugte Varianzbild. (Der Grauwertbereich wurde für diese Darstellung gestreckt.)

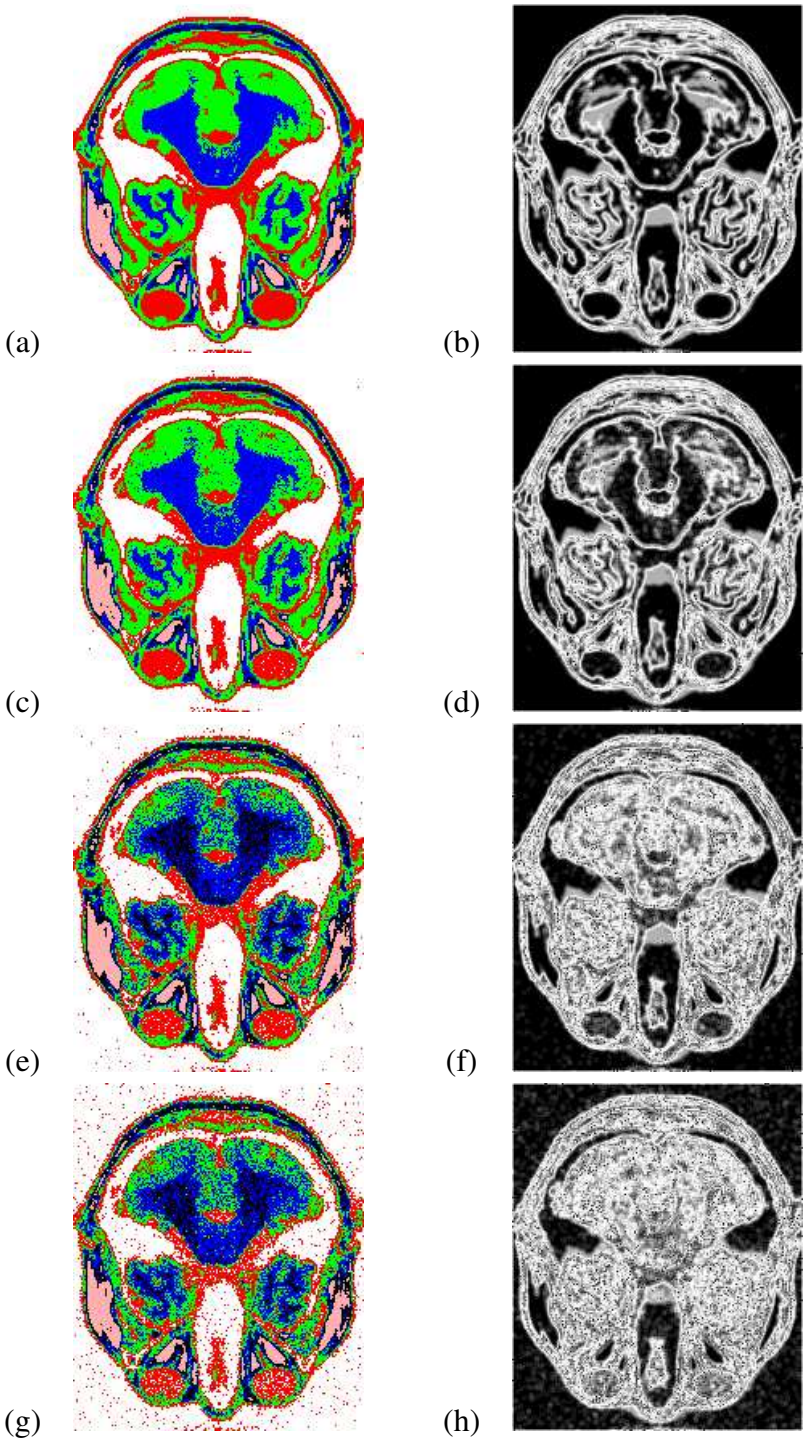


Abbildung 6.14: Diese Abbildung zeigt die Varianzbilder der Segmentierergebnisse des ML-Algorithmus, mit einem Rauschanteil von 3 % (erste Zeile), 5 % (zweite Zeile), 7 % (dritte Zeile) und 9 % (vierte Zeile).

6.3.2 *Mutual Information*

Definition 6.3.1 *Die* MUTUAL INFORMATION *zwischen zwei diskreten Zufallsvariablen X und Y ist definiert durch:*

$$MI(X,Y) := \sum_{x \in X} \sum_{y \in Y} p(x \cap y) \log_2 \frac{p(x \cap y)}{p(x)\, p(y)}. \tag{6.14}$$

Für die Berechnung der *Mutual Information* zweier Bilder benötigen wir wieder die Häufigkeitsmatrix H (siehe Abschnitt 6.2.1).

Die Mutual Information zwischen zwei Bildern I_1 und I_2 berechnet sich aus dessen Häufigkeitsmatrix H wie folgt:

$$MI(H) := \frac{1}{N} \cdot \sum_{i=0}^{\text{cols}(H)-1} \sum_{j=0}^{\text{rows}(H)-1} h_{i,,j} \cdot \log_2 \frac{h_{i,j}}{N \cdot \sum_{m=0}^{\text{cols}(H)} h_{m,n} \cdot \sum_{n=0}^{\text{rows}(H)} h_{m,n}}, \tag{6.15}$$

dabei bezeichnet $\text{cols}(H)$ die Anzahl der Spalten und $\text{rows}(H)$ die Anzahl der Zeilen der Matrix H. Weiters ist N die Anzahl der Bildpunkte von I_1 bzw. I_2. (Für die Berechnung der Matrix H siehe Algorithmus 11 auf Seite 107.)

6.3.3 Ergebnisse

Bei der Validierung des MAP-Algorithmus wurde nur mehr der Parameter β variiert. Die Anzahl der Merkmale k wurde auf sechs gesetzt. (Bei diesem Wert war die Gesamtfehlerrate des ML-Algorithmus am geringsten.) Die Abbildungen 6.15 bis 6.18 zeigen das Mutual Information Diagramm für verschiedene Rauschanteile der segmentierten Bilder, in Abhängigkeit des Cliquen- bzw. Gibbs Potentials β.

Die folgenden Abbildungen (6.19 bis 6.22) zeigen typische Segmentierergebnisse für die verschiedenen Rauschanteile der Testbilder. Segmentiert wurde jeweils mit jenem Gibbs Potential, bei dem die Mutual Information ein Maximum ergeben hat (Abbildung 6.15 bis 6.18). Weiters ist auch noch das zum Segmentierergebnis gehörige Varianzbild zu sehen.

Die Versuche zeigen, daß das Gibbs Potential sehr stark vom Rauschanteil der Eingabebilder abhängt. Je höher das Bildrauschen, desto höher muß auch das Gibbs Potential gewählt werden. Weiters hat sich gezeigt, daß sich bei geringem Bildrauschen der Einsatz des MAP-Algorithmus nicht lohnt; in diesem Fall sollte der ML-Algorithmus eingesetzt werden. Auch bei mäßig verrauschten Bildern muß überlegt werden, ob eine, um rund zehn mal, längere Laufzeit in Kauf genommen werden kann. Auf meinem Versuchssystem[3] dauerte die Segmentierung

[3]CPU: Athlon Duron 1200MHz; Hauptspeicher: 512MB DDR PC266; BS: Mandrake Linux 9.0 (Kernelversion: 2.4.19-16); JDK: Sun j2sdk1.4.1_01-b01

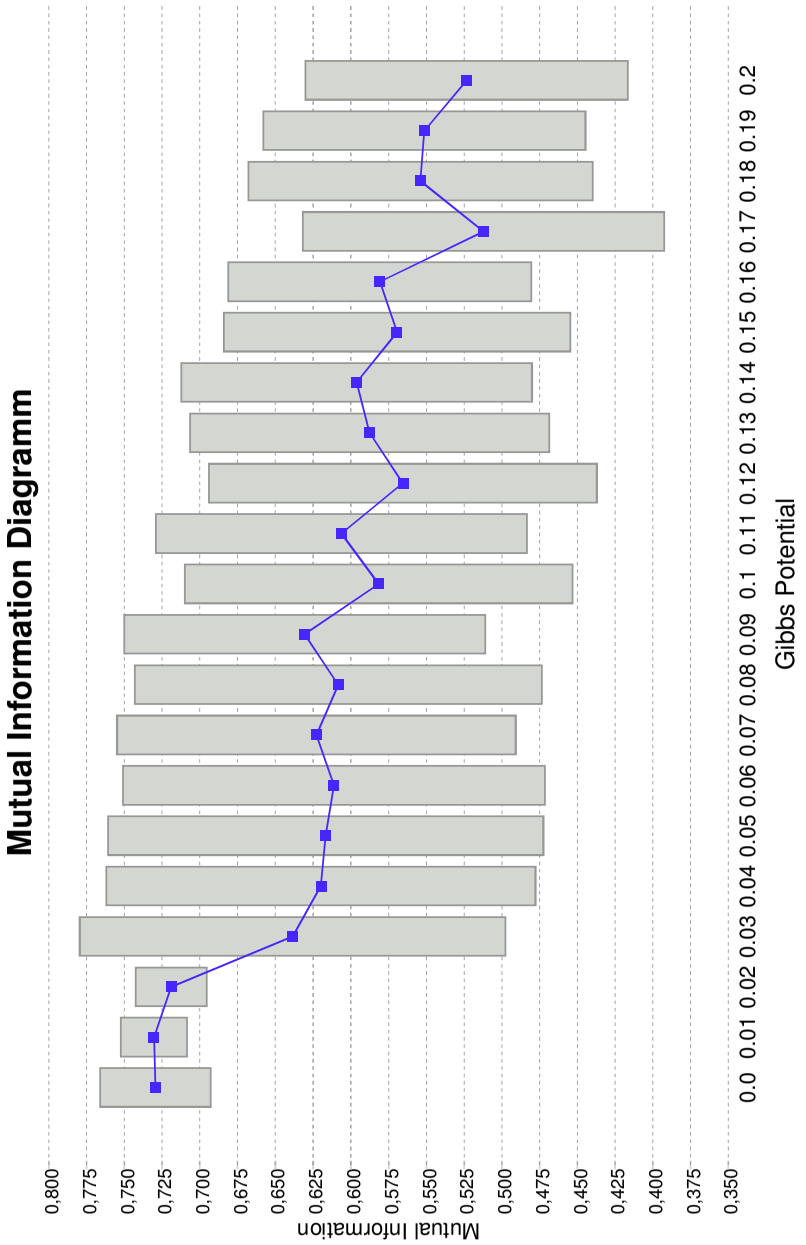


Abbildung 6.15: Mutual Information Diagramm. Rauschanteil des segmentierten, T_1 gewichteten Bildes: 3 %.

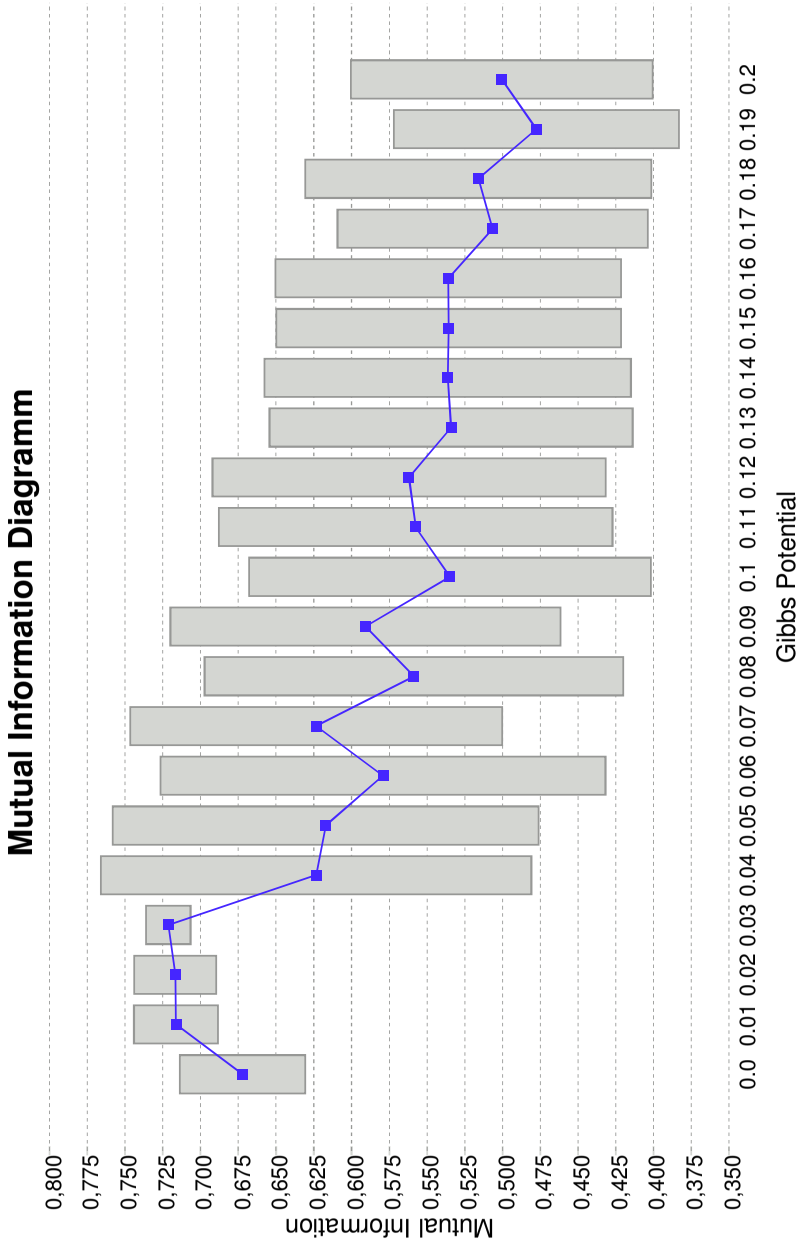


Abbildung 6.16: Mutual Information Diagramm. Rauschanteil des segmentierten,T_1 gewichteten Bildes: 5 %.

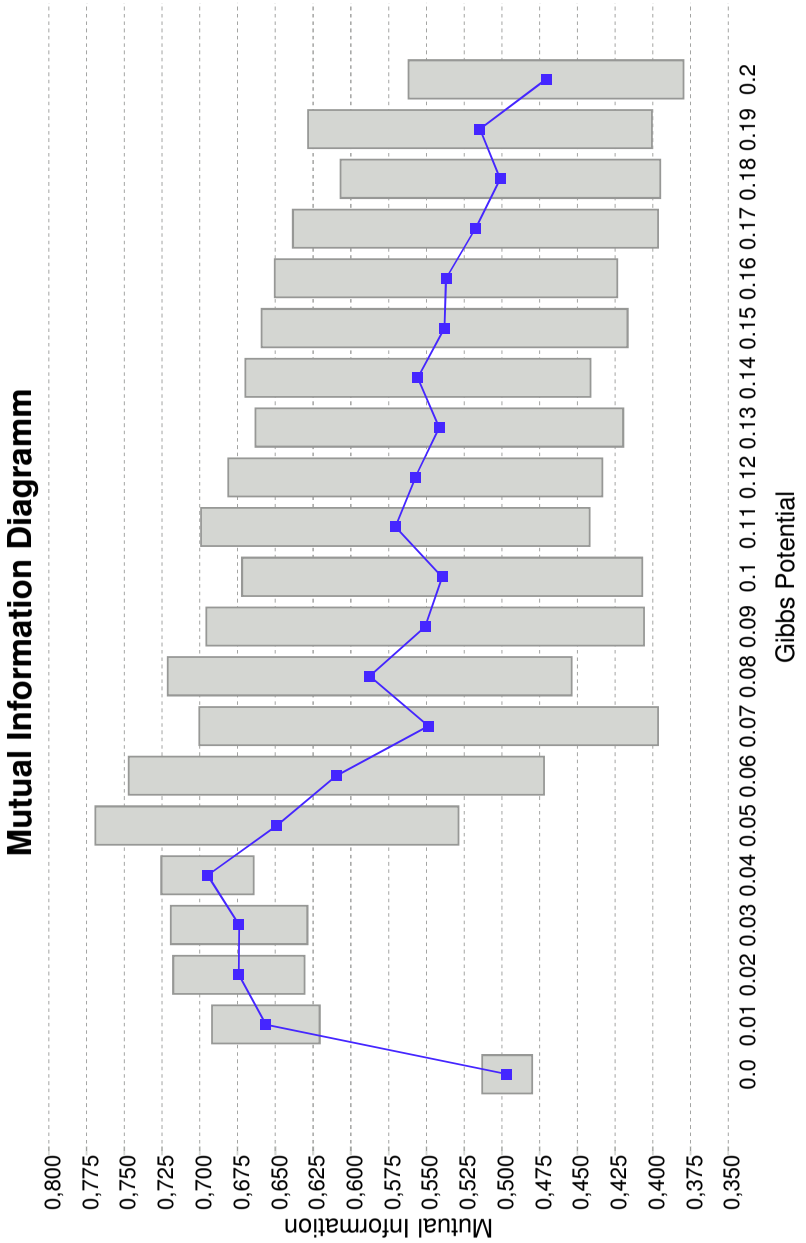


Abbildung 6.17: Mutual Information Diagramm. Rauschanteil des segmentierten,T_1 gewichteten Bildes: 7 %.

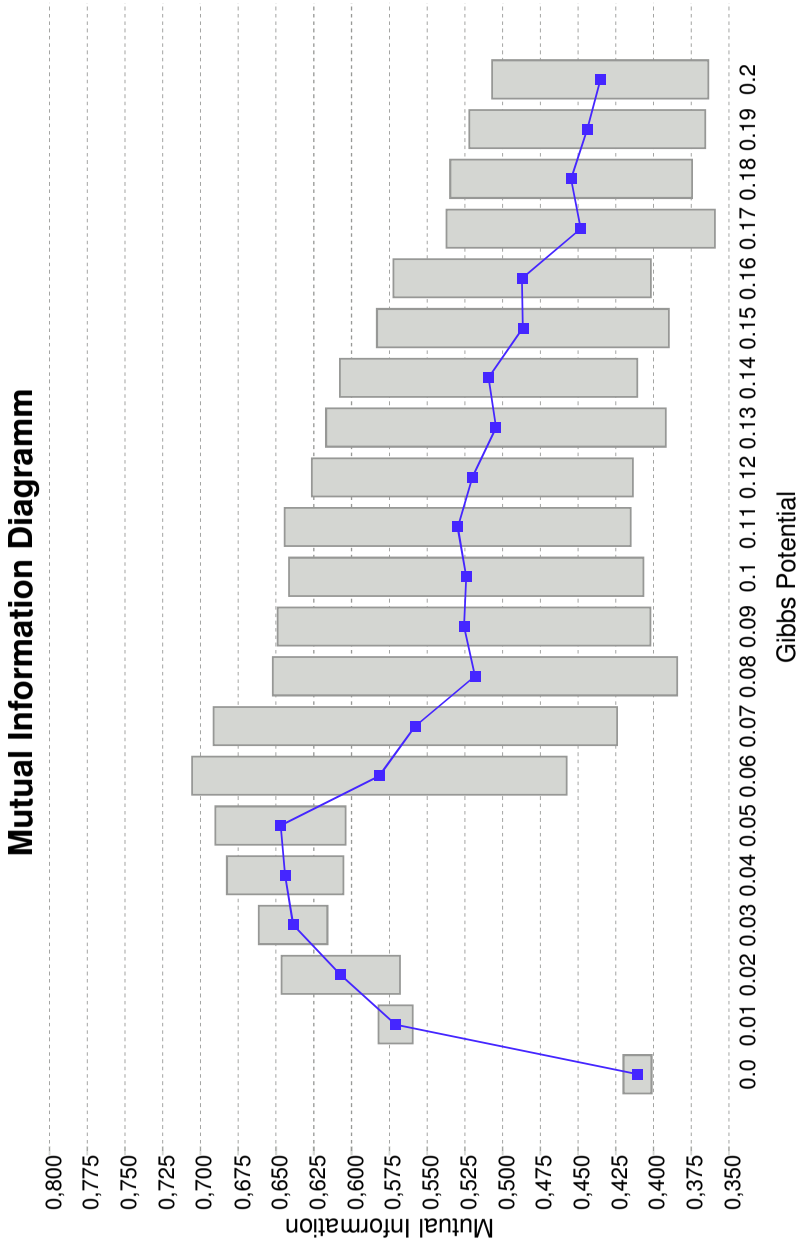
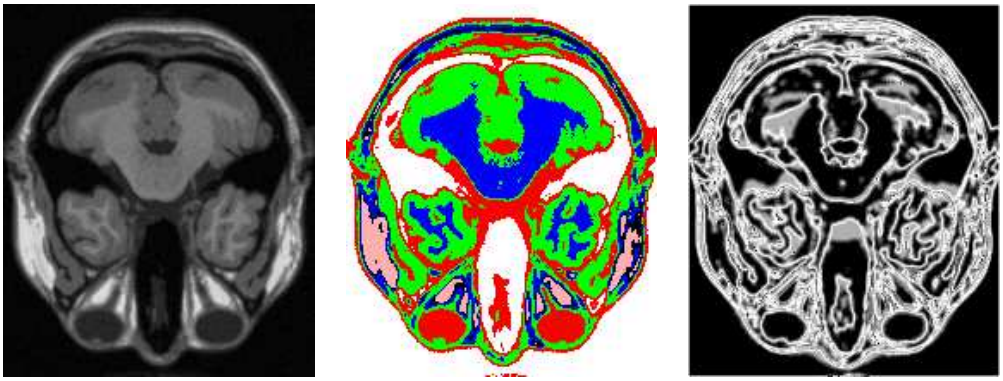
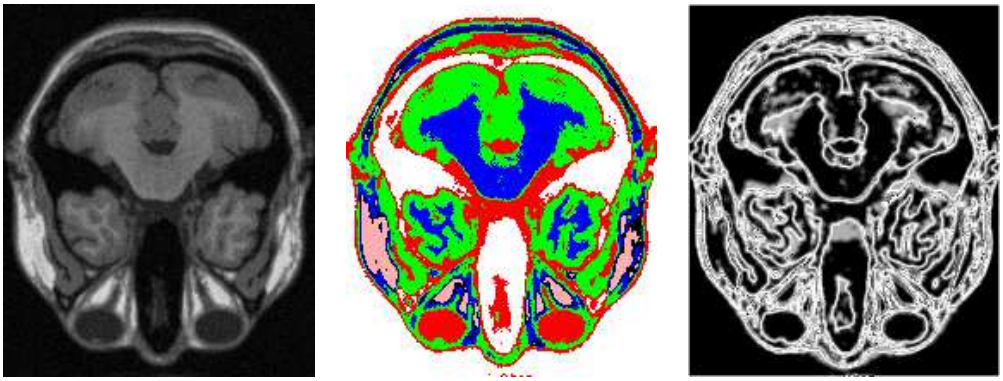


Abbildung 6.18: Mutual Information Diagramm. Rauschanteil des segmentierten,T_1 gewichteten Bildes: 9 %.



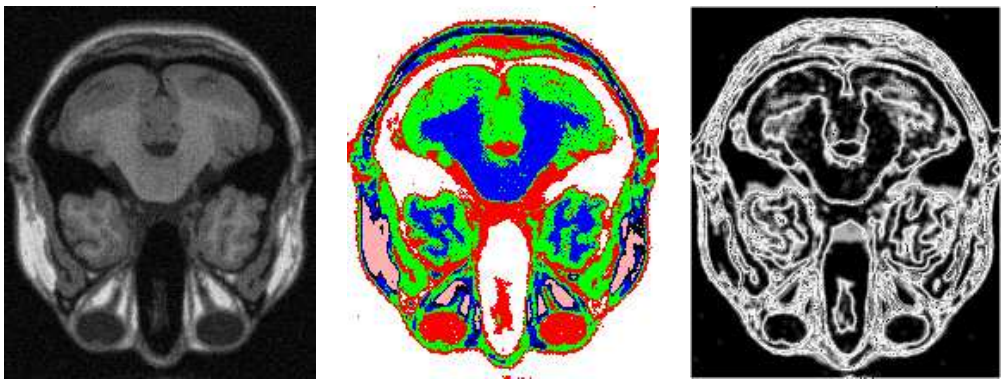
(a) (b) (c)

Abbildung 6.19: Bild (a) zeigt Schicht 35 des segmentierten Datensatzes (mit 3 % Rauschanteil), Bild (b) zeigt das Segmentierergebnis ($\beta = 0,01$) und Bild (c) zeigt das Varianzbild des Segmentierergebnisses.



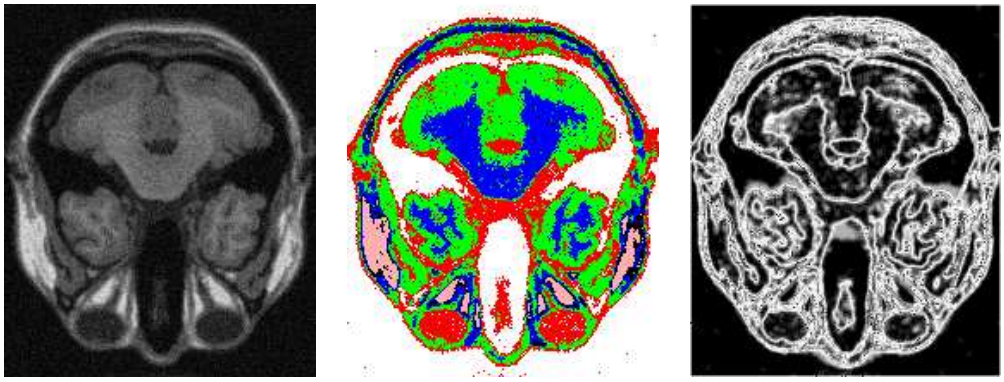
(a) (b) (c)

Abbildung 6.20: Bild (a) zeigt Schicht 35 des segmentierten Datensatzes (mit 5 % Rauschanteil), Bild (b) zeigt das Segmentierergebnis ($\beta = 0,03$) und Bild (c) zeigt das Varianzbild des Segmentierergebnisses.



(a) (b) (c)

Abbildung 6.21: Bild (a) zeigt Schicht 35 des segmentierten Datensatzes (mit 7 % Rauschanteil), Bild (b) zeigt das Segmentierergebnis ($\beta = 0,04$) und Bild (c) zeigt das Varianzbild des Segmentierergebnisses.



(a) (b) (c)

Abbildung 6.22: Bild (a) zeigt Schicht 35 des segmentierten Datensatzes (mit 9 % Rauschanteil), Bild (b) zeigt das Segmentierergebnis ($\beta = 0,05$) und Bild (c) zeigt das Varianzbild des Segmentierergebnisses.

eines Datensatzes[4] (in sechs Klassen) durch den MAP-Algorithmus ca. 30 Minuten, der ML-Algotihmus benötigte dazu etwas mehr als zwei Minuten.[5] Die Stärken des MAP-Algorithmus liegen somit in der Segmentierung von stark verrauschten Bildern, zum Preis einer nicht unerheblichen Segmentierungsdauer.

[4]Die Abmessung eines vollständigen Datensatzes betragen 181x217x181, also 7 109 137 Bildpunkte.

[5]Diese Angaben sollen nur die Größenordnung der Algorithmenlaufzeiten veranschaulichen. Explizite Laufzeitmessungen wurden nicht durchgeführt.

Anhang A

Glossar

Bindegewebe Organunspezifisches Füll-, Hüll- und Stützgewebe. Zu seinen Aufgaben im Körper zählt die Bündelung von Blutgefäßen und Nerven, die Auffüllung von organfreien Räumen und die Organkapselung.

CSF (*cerebrospinal fluid*) Gehirnflüssigkeit. (*lat.: cerebrospinalis*, das Gehirn und Rückenmark betreffen.)

Ektoderm Keimblatt der Säuger und des Menschen. Aus ihm geht die Anlage des Zentralnervensystems und der Sinnesorgane hervor.

Glia Hirngewebe. Das vom Ektoderm abstammende Zellgewebe des Nervensystems, das die Räume zwischen Nervenzellen und Blutgefäßen bis auf einen 20 nm breiten Spalt ausfüllt.

invasiv eindringend.

invasive Diagnostik Diagnostik unter Verletzung der Körperintegrität.

Läsion Verletzung oder Störung der Funktion eines Organs oder Körperglieds.

Markscheide Um die Nervenzellen durch Myelin gebildete Isolierschicht.

Markscheidenzerfall Myelinscheidenzerfall, Entmarkung; führt zu Funktionsverlust.

Multiple Sklerose Relativ häufige Entmarkungskrankheit des Zentralnervensystems (herdförmiger regellos verteilter Markscheidenzerfall).

Literaturverzeichnis

[1] K. Alsabti, S. Ranka, and V. Singh. An Efficient Parallel Algorithm for High Dimensional Similarity Join. In *IPPS: 11th International Parallel Processing Symposium*. IEEE Computer Society Press, 1998. URL `http://citeseer.nj.nec.com/alsabti98efficient.html`.

[2] E. Behrends. *Introduction to Markov Chains*. Vieweg, 2000. ISBN 3-528-06986-4.

[3] A. Bieniek and A. Moga. A Connected Component Approach to the Watershed Segmentation. *Mathematical Morphology and Its Applications to Image and Signal Processing*, 12:215–222, 1998.

[4] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistic Society, Series B (Methodological)*, 39(1):1–38, 1977.

[5] O. Dösel. *Bildgebende Verfahren in der Medizin: von der Technik zur medizinischen Anwendung*. Springer, 2000. ISBN 3-540-66014-3.

[6] V. Faber. Clustering and the Continouse k-means Algorithm. *Los Alamos Science*, (22):138–144, 1994.

[7] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Addison-Wesley, 1992. ISBN 0-201-50803-6.

[8] H. Handels. *Medizinische Bildverarbeitung*. B.G. Teubner, 2000. ISBN 3-519-02947-2.

[9] D. Judd, P. McKinley, and A. Jain. Large-Scale Parallel Data Clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(8):871–876, August 1996.

[10] R. K. Kwan, A. C. Evans, and G. B. Pike. An Extensible MRI Simulator for Post-Processing Evaluation. *Lecture Notes in Computer Science*, 1131:135–140, September 1996. URL `http://www.bic.mni.mcgill.ca/brainweb`.

[11] R. K. Kwan, A. C. Evans, and G. B. Pike. MRI Simulation-Based Evaluation of Image-Processing and Classification Methods. *IEEE Transactions on Medical Imaging*, 18(11):1085–1097, November 1999.

[12] T. McInerney and D. Terzopoulos. Deformable Models in Medical Image Analysis: A Survey. *Medical Image Analysis*, 1(2):91–108, 1996.

[13] T. McInerney and D. Terzopoulos. T-Snakes: Topology Adaptive Snakes. *Medical Image Analysis*, 4(2):73–91, 2000. URL `http://citeseer.nj.nec.com/mcinerney99tsnakes.html`.

[14] G. J. McLachlan and K. E. Basford. *Mixture Models; Inverence and Applications to Clustering*. Marcel Dekker, 1988. ISBN 0-8247-7691-7.

[15] J. Roerdink and A. Meijster. The Watershed Transform: Definitions, Algorithms and Parallelization Strategies. *Fundamenta Informaticae*, 41(1-2):187–228, 2000. URL `http://citeseer.nj.nec.com/roerdink00watershed.html`.

[16] P. Smyth. Clustering using Monte Carlo Cross-Validation. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, pages 126–133. AAAI Press, August 1996.

[17] M. X. H. Yan and J. S. Karp. An Adaptive Bayesian Approach to Three-Dimensional MR Brain Segmentation. University of Pennsylvania, 1995. URL `http://www.uphs.upenn.edu/bbl/software/-files/segm-0.90.src.tgz`.