View metadata, citation and similar papers at core.ac.uk
brought to you by CORE
provided by Qucosa - Publikationsserver der Universität Leipzig

Universität Leipzig
Fakultät für Mathematik und Informatik
Institut für Informatik

Segmentierung des Knochens aus T_1- und PD-gewichteten Kernspinbildern vom Kopf

Diplomarbeit

vorgelegt von
Stefan Burkhardt

Betreuer: Prof. Dr. Dietmar Saupe
Dr. Frithjof Kruggel

Leipzig, im Oktober 2000

An dieser Stelle möchte ich den Herren Prof. Dr. Dietmar Saupe und Dr. Frithjof Kruggel für die Betreuung und Unterstützung bei der Erstellung dieser Arbeit danken. Ein weiterer Dank gilt Herrn Dipl.-Math. Carsten Wolters für seine Hilfe bei den numerischen Algorithmen.

Inhaltsverzeichnis

1 Einleitung **1**
- 1.1 Motivation . . . 1
- 1.2 Zur Gliederung dieser Arbeit . . . 3

2 Grundlagen **5**
- 2.1 Bilder . . . 5
- 2.2 Histogramme . . . 6
- 2.3 Räumliche Transformationen . . . 7
  - 2.3.1 Definition . . . 7
  - 2.3.2 Vorwärts- und Rückwärtsabbildung . . . 7
  - 2.3.3 Resampling . . . 8
- 2.4 Filter . . . 9
  - 2.4.1 Lee-Filter . . . 9
  - 2.4.2 Anisotrope Diffusion . . . 9

3 Registrierung **11**
- 3.1 Bestandteile von Registrieralgorithmen . . . 11
- 3.2 Die Abbildung . . . 12
- 3.3 Die Kostenfunktion . . . 14
- 3.4 Optimierung . . . 16
  - 3.4.1 Downhill Simplex Algorithmus . . . 17
  - 3.4.2 Genetische Optimierung . . . 19
  - 3.4.3 Powells Algorithmus . . . 21
  - 3.4.4 Bewertung der Verfahren . . . 23
- 3.5 Implementierung . . . 25
  - 3.5.1 Mehrgitterrepräsentation . . . 27
  - 3.5.2 Optimierung mit Downhill-Simplex-Algorithmus . . . 27
- 3.6 Bewertung der Implementierung . . . 27
  - 3.6.1 Ergebnis der Registrierung . . . 27
  - 3.6.2 Einfluß von Filtern auf das Ergebnis . . . 28
- 3.7 Registrierung der verwendeten Kernspinbilder . . . 29

4 Segmentierung **33**
- 4.1 Der Isodata-Algorithmus . . . 33
- 4.2 Segmentierung mit dem Isodata-Algorithmus . . . 35
- 4.3 Ein adaptiver Fuzzy-C-Means Algorithmus . . . 44

4.3.1 Berechnung des Korrekturfeldes 47
4.3.2 Beispiele für die Segmentierung 54
4.4 Segmentierung mit dem AFCM-Algorithmus 57
4.4.1 Segmentierung aus dem PD-gewichteten Kernspinbild . . 57
4.4.2 Segmentierung aus dem T_1- und PD-gewichteten Kernspinbild 62
4.5 Elastische Modelle 63
4.5.1 Extraktion der Oberflächen 65
4.5.2 Vereinfachung der Dreiecksnetze 65
4.5.3 Anpassung der Dreiecksnetze 66
4.6 Implementierung 67
4.7 Bewertung 70

5 Zusammenfassung **79**

A Verwendete Operationen **81**

Literaturverzeichnis **83**

Abbildungsverzeichnis

1.1 T_1- und PD-gewichtetes Kernspinbild 1

2.1 Anordnung von Schichten, Zeilen und Spalten im Kernspinbild . 6

3.1 Der verwendete Downhill-Simplex-Algorithmus 17
3.2 Typische Struktur eines genetischen Algorithmus 20
3.3 Struktur eines Powells Algorithmus 21
3.4 Aufbau des implementierten Registrieralgorithmus 28
3.5 Registrierung der in dieser Arbeit verwendeten Kernspinbilder . . 31

4.1 Der Isodata-Algorithmus 34
4.2 Segmentierung eines T_1- und eines PD-gewichteten Bildes mit Isodata und verschiedenen Anzahlen an Klassen 35
4.3 Algorithmus für eine Segmentierung des Knochens aus einem PD-gewichteten Bild auf Basis des Isodata-Algorithmus 37
4.4 Zwischenergebnisse bei der Erstellung der Kopfmaske 38
4.5 Zwischenergebnisse bei der Erstellung der Liquormaske 38
4.6 Zwischenergebnisse bei der Erstellung der Knochenmaske 38
4.7 Kopfmasken für unterschiedliche Werte des Distanzparameters . 41
4.8 Komplette Segmentierung eines PD-gewichteten Kernspinbildes . 42
4.9 Einblendung des Randes der Kopf-, Liquor- und Knochenmaske in das Originalbild 43
4.10 Der adaptive Fuzzy-C-Means (AFCM)-Algorithmus 48
4.11 Norm des Residuums bei der Lösung des Gleichungssystems mit dem Jacobi-Verfahren 51
4.12 Das Verfahren der konjugierten Gradienten 53
4.13 Euklidische Norm des Residuums bei der Lösung des Gleichungssystems mit dem skalierten CG-Verfahren 54
4.14 Vergleich der Segmentierungen mit dem Isodata- und dem AFCM-Algorithmus 56
4.15 Auf Basis des AFCM-Algorithmus segmentierte Kopfmasken . . 58
4.16 Komplette Segmentierung eines PD-gewichteten Kernspinbildes mit einem AFCM-basierten Algorithmus 59
4.17 Einblendung des Randes der Kopf-, Liquor- und Knochenmaske in das Kernspinbild 60
4.18 Gegenüberstellung der Ergebnisse der Knochensegmentierung unter Verwendung des Isodata- und des AFCM-Algorithmus 61

4.19 Erste Segmentierung des Knochens aus einem T_1 und einem PD-gewichteten Kernspinbild 63
4.20 Zweite Segmentierung des Knochens aus einem T_1 und einem PD-gewichteten Kernspinbild 64
4.21 Initiale äußere und innere Kante des Knochens in einem Bild . . 68
4.22 Angepaßte äußere und innere Kante des Knochens 68
4.23 Erstellung des modifizierten T_1-gewichteten Bildes 70
4.24 Verbesserte initiale äußere und initiale innere Kante des Knochens in einem Bild 71
4.25 Angepaßte äußere und innere Kante des Knochens zu den initialen Kanten aus Abbildung 4.24 71
4.26 Innere und äußere Kante des Knochens im Datensatz 1 74
4.27 Innere und äußere Kante des Knochens im Datensatz 2 75
4.28 Innere und äußere Kante des Knochens im Datensatz 3 76
4.29 Innere und äußere Kante des Knochens im Datensatz 4 77
4.30 Innere und äußere Kante des Knochens im Datensatz 5 78

Tabellenverzeichnis

3.1 Wichtige Eigenschaften von Mutual Information 16
3.2 Abbildungsparameter für die Testpaare 23
3.3 Auswirkung einer Variierung der Stoppgitterweite auf das Downhill-Simplex-Verfahren 24
3.4 Auswirkung einer Änderung der maximalen Anzahl von Populationen auf die genetische Optimierung 25
3.5 Auswirkung einer Änderung der Anzahl an Stützstellen auf den Powells Algorithmus 26
3.6 Registrierungsergebnisse für die Höhe $h = 1$ der Gaußpyramide . 29
3.7 Registrierungsergebnisse für die Höhe $h = 2$ der Gaußpyramide . 29
3.8 Registrierungsergebnisse für die Höhe $h = 3$ der Gaußpyramide . 30
3.9 Auswertung und Vergleich der ermittelten Abbildungsparameter in Abhängigkeit der Höhe der Gaußpyramide 30
3.10 Registrierungsergebnisse bei einer Filterung der Bilder 30
3.11 Abbildungsparameter und benötigte Zeit für die Registrierung der Kernspinbilder 32

4.1 Benötigte Zeiten für die Segmentierung des Knochens 73

Kapitel 1

Einleitung

1.1 Motivation

Für vielfältige Anwendungen, beispielsweise bei der Lokalisation elektromagnetischer Hirnaktivität (Quellokalisation) oder der Simulation biomechanischer Eigenschaften des Kopfes, ist ein individuelles Modell des Kopfes notwendig. Bislang wird dieses Modell aus T_1-gewichteten Kernspinbildern (Abbildung 1.1, links) erstellt.

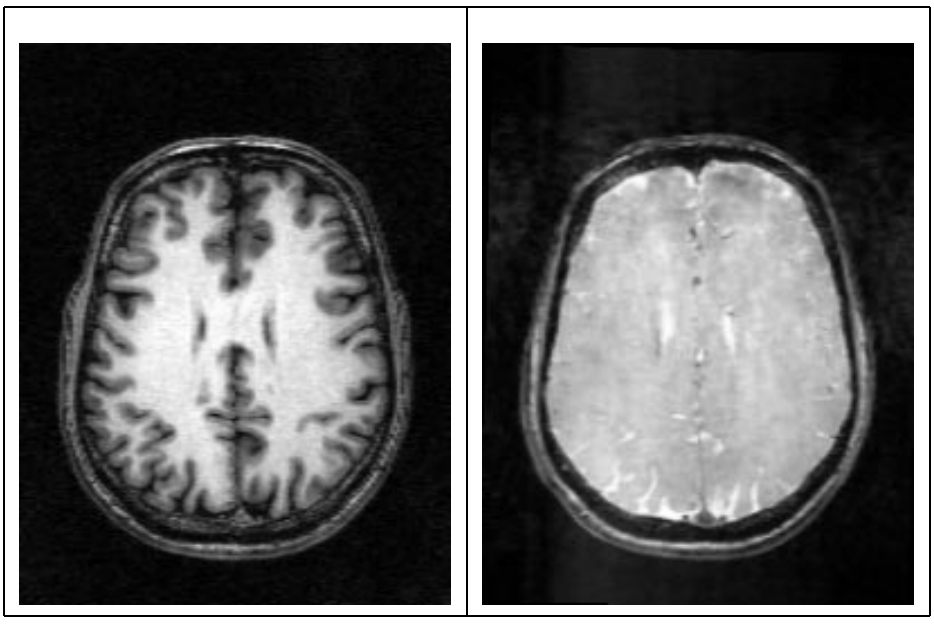


Abbildung 1.1: T_1- und PD-gewichtetes Kernspinbild

Auf einem T_1-gewichteten Bild sind die folgenden Gewebetypen zu erkennen (von außen nach innen):

- die Kopfhaut als heller Bereich,
- der Knochen und die Gehirnflüssigkeit (Liquor) als dunkler Bereich,
- die graue und die weiße Substanz des Gehirns als heller Bereich.

Als Vorteile dieser Bilder sind anzusehen:

1. Das Aufnahmeverfahren ist schnell. Dadurch besitzen die Bilder eine hohe räumliche Auflösung.

2. Es sind viele Gewebetypen unterscheidbar. In diesen Bilder kann sehr gut die Kopfhaut und die graue und weiße Substanz vom Gehirn erkannt und segmentiert werden.

Nachteilig bei diesen Bildern ist, daß kein Unterschied zwischen dem Knochen und der Gehirnflüssigkeit vorhanden ist. Um dennoch eine Trennung vorzunehmen, wird die innere Kante des Knochens geschätzt. Zu diesem Zweck wird die Gehirnoberfläche segmentiert und geglättet. Die innere Kante des Knochens wird in einem konstanten Abstand von dieser geglätteten Oberfläche angenommen.
Nach [11, 42] ist aber das Ergebnis der Quellokalisation entscheidend von der Geometrie des Knochens abhängig. Für eine bessere Segmentierung des Knochens wird ein weiteres, PD-gewichtetes Kernspinbild aufgenommen (Abbildung 1.1, rechts).
Auf dem PD-gewichteten Bild sind die folgenden Gewebetypen zu erkennen (von außen nach innen):

- die Kopfhaut als heller Bereich,

- der Knochen als dunkler Bereich,

- die Gehirnflüssigkeit und das Gehirn als heller Bereich.

Der Vorteil eines PD-gewichteten Kernspinbildes ist, daß ein großer Kontrast zwischen den protonenhaltigen (Kopfhaut, Gehirnflüssigkeit, Gehirn) und den protonenarmen Gewebetypen (Knochen) vorhanden ist. Dadurch ist der Knochen als dunkler Bereich zwischen der Kopfhaut und der Gehirnflüssigkeit zu erkennen. Nachteilig ist, daß keine Trennung zwischen Gehirnflüssigkeit, grauer und weißer Substanz möglich ist. Daher kann das Bild nur zur Unterstützung der Segmentierung aus dem T_1-gewichteten Bild verwendet werden.
Das Ziel dieser Arbeit ist es, den Knochen aus einem sogenannten *dual-echo*-Datensatz, bestehend aus dem T_1- und dem PD-gewichteten Kernspinbild, zu segmentieren. Dazu sind zwei Schritte erforderlich:

1. Registrierung des PD-gewichteten Kernspinbildes auf dem T_1-gewichteten,

2. Segmentierung des Knochens unter Verwendung der Informationen aus beiden Bildern.

Das Ergebnis dieser Arbeit sind Verfahren, die durch Verwendung von dual-echo-Datensätzen eine verbesserte Segmentierung des Knochens aus Kernspinbildern ermöglichen. Bislang erfolgt lediglich eine teilweise sehr ungenaue Schätzung der Kante zwischen Knochen und Gehirnflüssigkeit. Von Vorteil der Verfahren ist, daß diese beiden Schritte auf den vorhandenen Testdaten automatisch ausgeführt werden.

1.2 Zur Gliederung dieser Arbeit

Die vorliegende Arbeit gliedert sich in fünf Kapitel. Zunächst werden die im weiteren benötigten Definitionen und Grundlagen zusammengestellt. Das dritte Kapitel behandelt die Registrierung. Daran anschließend wird im vierten die Segmentierung beschrieben. Die Ergebnisse sind im fünften Kapitel zusammengefaßt.

Kapitel 2

Grundlagen

In diesem Kapitel werden Definitionen und Grundlagen zusammengestellt, auf die im folgenden zurückgegriffen wird.

2.1 Bilder

Mathematisch können Grauwertbilder als Abbildungen von einer Trägermenge in eine Grauwertmenge beschrieben werden.

Definition 2.1 *Ein n-dimensionales Bild b ist ein Tripel* $b = (\mathcal{T}, \mathcal{G}, I)$*, bestehend aus zwei Mengen* $\mathcal{T} \subseteq \mathbb{R}^n$ *und* $\mathcal{G} \subseteq \mathbb{R}$ *und einer Funktion* $I : \mathcal{T} \to \mathcal{G}$*. Die Menge* $\mathcal{T}$ *heißt Trägermenge, die Menge* $\mathcal{G}$ *Grauwertmenge.*

Alle Bilder werden in der Menge $\mathcal{B}$ zusammengefaßt.

Definition 2.2 *Die Menge* $\mathcal{B}$ *aller Bilder ist definiert als:*

$$\mathcal{B} := \{(\mathcal{T}, \mathcal{G}, I) \mid \mathcal{T} \subseteq \mathbb{R}^n \ \wedge \ \mathcal{G} \subseteq \mathbb{R} \ \wedge \ I : \mathcal{T} \to \mathcal{G}\}$$

Sei $\mathcal{T} \subseteq \mathbb{R}^n$. Die Menge $\mathcal{B}_\mathcal{T}$ aller Bilder mit Trägermenge $\mathcal{T}$ ist eine Teilmenge der Menge $\mathcal{B}$.

Definition 2.3 *Die Menge* $\mathcal{B}_\mathcal{T}$ *aller Bilder mit Trägermenge* $\mathcal{T}$ *ist definiert als:*

$$\mathcal{B}_\mathcal{T} := \{(\mathcal{T}, \mathcal{G}, I) \mid \mathcal{G} \subseteq \mathbb{R} \ \wedge \ I : \mathcal{T} \to \mathcal{G}\} \ .$$

Die in dieser Arbeit verwendeten Kernspinbilder sind dreidimensional mit diskreter Trägermenge

$$\mathcal{T} = \{0, 1, \ldots, n_c - 1\} \times \{0, 1, \ldots, n_r - 1\} \times \{0, 1, \ldots, n_s - 1\} \ ,$$

wobei das Bild n_s Schichten, n_r Zeilen und n_c Spalten besitzt. Die typischen Werte sind $n_s = 250$, $n_c = 250$ und $n_c = 200$. Die Anordnung von Schichten, Zeilen und Spalten ist in Abbildung 2.1 dargestellt.

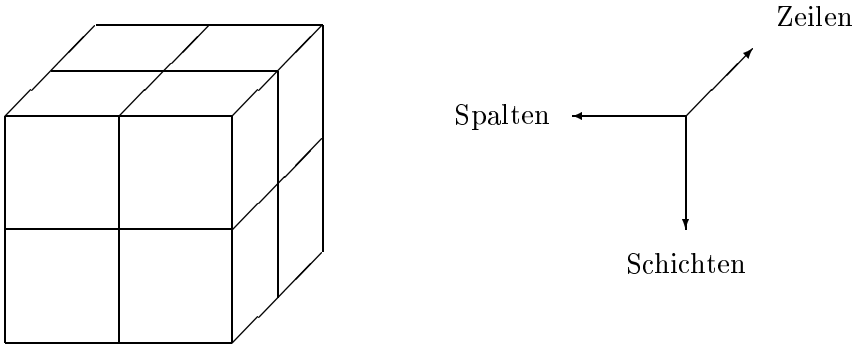


Abbildung 2.1: Anordnung von Schichten, Zeilen und Spalten im Kernspinbild

Innerhalb dieses Bildes ist der Kopf so orientiert, daß:

- die Numerierung der Schichten von oben nach unten,
- die Numerierung der Spalten von links nach rechts und
- die Numerierung der Zeilen von vorn nach hinten

erfolgt. Bei der Numerierung der Spalten ist zu beachten, daß man in Abbildung 2.1 von vorn auf den Kopf sieht.
In diesem Bild sei ein Koordinatensystem mit folgenden Eigenschaften definiert:

1. Der Ursprung befindet sich bei Voxel $\left(\frac{n_c}{2}, \frac{n_r}{2}, \frac{n_s}{2}\right)$.
2. Die positive x-Achse ist in Richtung aufsteigender Spaltennummern gerichtet, die positive y-Achse in Richtung aufsteigender Zeilennummern, die positive z-Achse in Richtung aufsteigender Schichtnummern.

In dieser Arbeit werden die drei folgenden Schnitte durch ein Bild verwendet:

1. der axiale Schnitt, dessen Schnittebene parallel zur x-y-Ebene verläuft,
2. der coronare Schnitt, dessen Schnittebene parallel zur y-z-Ebene verläuft und
3. der sagitale Schnitt, dessen Schnittebene parallel zur x-z-Ebene verläuft.

2.2 Histogramme

Ein Histogramm gibt Aufschluß über die Grauwertverteilung in einem Bild.

Definition 2.4 *Ein Histogramm von einem Bild* $b = (\mathcal{T}, \mathcal{G}, I) \in \mathcal{B}$ *ist eine Funktion*[1] $h : \mathcal{G} \to \mathbb{N}_0$, *wobei*[2] $h(g) = \mid \{\mathbf{t} | \mathbf{t} \in \mathcal{T} \wedge I(\mathbf{t}) = g\} \mid$.

[1] $\mathbb{N}_0$: Menge der natürlichen Zahlen und 0
[2] Für eine Menge $\mathcal{A}$ bezeichnet $|\mathcal{A}|$ die Anzahl ihrer Elemente.

Somit liefert das Histogramm eines Bildes $b = (\mathcal{T}, \mathcal{G}, I)$ für jeden Grauwert $g \in \mathcal{G}$ die Anzahl der Pixel im Bild, die diesen Grauwert besitzen.
Histogramme können so verallgemeinert werden, daß sie Aufschluß über die Grauwertverteilung in n Bildern mit gleicher Trägermenge $\mathcal{T}$ geben. Die n-dimensionalen Verallgemeinerungen eines Histogrammes heißen n-dimensionales Scatterplot. Im weiteren werden allerdings nur zweidimensionale Scatterplots benötigt.

Definition 2.5 *Ein zweidimensionales Scatterplot für zwei Bilder* $b_1 = (\mathcal{T}, \mathcal{G}_1, I_1)$ *und* $b_2 = (\mathcal{T}, \mathcal{G}_2, I_2)$ *ist eine Funktion* $s_2 : \mathcal{G}_1 \times \mathcal{G}_2 \to \mathbb{N}_0$, *wobei*

$$s_2(g_1, g_2) = \mid \{\mathbf{t} \mid \mathbf{t} \in \mathcal{T} \wedge I_1(\mathbf{t}) = g_1 \wedge I_2(\mathbf{t}) = g_2\} \mid .$$

2.3 Räumliche Transformationen

2.3.1 Definition

Eine räumliche Transformation definiert eine geometrische Beziehung zwischen den Pixeln zweier Bilder [12, 41].

Definition 2.6 *Sei* $n \in \mathbb{N}$. *Eine räumliche Transformation* T *ist eine Abbildung* $T : \mathbb{R}^n \to \mathbb{R}^n$.

Werden räumliche Transformationen auf die Trägermenge $\mathcal{T}_1$ eines Bildes $b_1 = (\mathcal{T}_1, \mathcal{G}, I_1)$ eingeschränkt, so ist das Ergebnis ein Bild $b_2 = (\mathcal{T}_2, \mathcal{G}, I_2)$ mit $\mathcal{T}_2 = \{T(t) \mid t \in \mathcal{T}_1\}$ und $I_2(t_2) = I_1(T^{-1}(t_2))$ für alle $t_2 \in \mathcal{T}_2$. Die Schreibweise für diese Transformation ist im folgenden $b_2 = T(b_1)$. Soll das Bild b_2 eine bestimmte vorgegebene Trägermenge $\mathcal{T}_2$ haben, wird das durch die Schreibweise $b_2 = T_{\mathcal{T}_2}(b_1)$ dargestellt.

2.3.2 Vorwärts- und Rückwärtsabbildung

Sei T eine Transformation, b_1 und b_2 zwei Bilder mit Trägermengen $\mathcal{T}_1$ und $\mathcal{T}_2$. Die Transformation kann auf zwei Arten ausgeführt werden:

1. Ausführung als Vorwärtsabbildung: In diesem Fall wird zu jedem Pixel $t_1 \in \mathcal{T}_1$ das korrespondierende Pixel $T(t_1) \in \mathcal{T}_2$ ermittelt.

2. Ausführung als Rückwärtsabbildung: In diesem Fall wird zu jedem Pixel $t_2 \in \mathcal{T}_2$ das entsprechende Pixel $t_1 \in \mathcal{T}_1$ mit $T(t_1) = t_2$ ermittelt.

Im Fall der hier betrachteten medizinischen Bilddaten sind die Trägermengen $\mathcal{T}_1$ und $\mathcal{T}_2$ diskrete Mengen. Zudem ist $\mathcal{T}_2$ meist gegeben und wird nicht als Bild der Menge $\mathcal{T}_1$ gebildet. Das kann dazu führen, daß es bei der Vorwärtsabbildung ein Pixel $t_1 \in \mathcal{T}_1$ mit $T(t_1) \notin \mathcal{T}_2$ gibt. Analog kann bei der Rückwärtsabbildung ein Pixel $t_2 \in \mathcal{T}_2$ existieren, für das es kein Pixel $t_1 \in \mathcal{T}_1$ mit $T(t_1) = t_2$ gibt.
Beim Auftreten solcher Fälle muß das Bild neu abgetastet werden. Bei der Vorwärtsabbildung muß das Bild b_2 aus den ermittelten Pixeln $T(t_1)$ mit $t_1 \in \mathcal{T}_1$ aufgebaut werden. Wird die Rückwärtsabbildung verwendet, muß das Bild b_1 an den Stellen $t_1 \in \mathbb{R}^n$ mit $T(t_1) \in \mathcal{T}_2$ neu abgetastet werden. Dieser Schritt heißt Resampling und wird im nächsten Teilabschnitt beschrieben.

2.3.3 Resampling

Resampling besteht aus zwei Teilschritten: der Rekonstruktion des originalen Bildes und der Abtastung des rekonstruierten Bildes an den neu festgelegten Stellen [41].
Seien b_1 und b_2 zwei Bilder mit Trägermengen $\mathcal{T}_1, \mathcal{T}_2 \in \mathbb{R}^n$ und $T : \mathbb{R}^n \to \mathbb{R}^n$ eine räumliche Transformation. Das Resampling kann auf zwei Arten erfolgen:

1. Anwendung der Transformation T auf b_1. Es entsteht ein Bild $b_2' = T(b_1)$. Aus b_2' wird das transformierte Originalbild rekonstruiert. Durch Abtastung an den Stellen, die durch $\mathcal{T}_2$ gegeben sind, entsteht das Bild b_2 oder

2. Konstruktion der Menge $\mathcal{T}_1' \subseteq \mathbb{R}^n$ mit der Eigenschaft:

$$\forall t_2 \in \mathcal{T}_2 \; \exists t \in \mathcal{T}_1' : T(t) = t_2 \; .$$

   Aus dem Bild b_1 wird das Originalbild rekonstruiert und an den Punkten, die durch $\mathcal{T}_1'$ gegeben sind, abgetastet. Das Ergebnis ist ein Bild b_1'. Das Bild b_2 ergibt sich dann durch $b_2 = T(b_1')$.

Die Variante zwei ist vorzuziehen. Der Vorteil ist, daß zu jedem Pixel t_2 das Urbild t_1 mit $t_2 = T(t_1)$ ermittelt werden kann. Das Originalbild muß dann jeweils nur an der Stelle t_1 rekonstruiert werden.

Interpolation

Die Interpolation ist eine Möglichkeit, Grauwerte von Pixeln zu erhalten, die nicht auf den Gitterpunkten der Trägermenge liegen [41].
Die numerische Genauigkeit und der Aufwand für die Interpolation werden durch den Interpolationskern bestimmt. Nach [41] sind Interpolationskerne separierbar. Damit kann die Interpolation nacheinander für jede Koordinate einzeln durchgeführt werden. Aus diesem Grund wird jetzt die Interpolation für ein eindimensionales, diskretes Signal[3] $f : \mathcal{X} \to \mathbb{R}, \mathcal{X} \subseteq \mathbb{R}, \; \mathcal{X} = \{x_k | k \in \mathbb{Z}\}$ betrachtet, das an den Stützstellen x_k gegeben ist. Der Abstand zwischen x_k und x_{k+1} sei konstant für alle k.
Der einfachste Interpolationskern ergibt sich für die sogenannte *Nearest-Neighbor-Interpolation* [41]. In diesem Fall ergibt sich das Signal $g : \mathbb{R} \to \mathbb{R}$ als:

$$g(t) = f(x_k) \quad \text{mit} \quad \|x_k - t\| \leq \|x_j - t\| \quad \forall j \in \mathbb{Z} \tag{2.1}$$

Bei der Nearest-Neighbor-Interpolation ist von Nachteil, daß nur dann ein exaktes Ergebnis entsteht, wenn das Signal stückweise konstant ist.

[3] Die verwendeten Bilder erfüllen die Voraussetzung, daß der Abstand zwischen zwei benachbarten Stützstellen konstant ist. Weiterhin ist eine Erweiterung von $\mathcal{T}$ zu $\mathbb{Z}^3$ möglich, indem die Grauwerte für Voxel außerhalb des Bildes als konstant und gleich 0 angenommen werden.

Eine verbesserte Variante bietet die lineare Interpolation. Das interpolierte Signal $g : \mathbb{R} \to \mathbb{R}$ ergibt sich als [41]:

$$g(t) = f(x_k) + (t - x_k)\frac{f(x_{k+1}) - f(x_k)}{x_{k+1} - x_k} \quad \text{mit} \quad x_k \leq t \leq x_{k+1} \tag{2.2}$$

Gegenüber der Nearest-Neighbor-Interpolation führt die lineare Interpolation zu besseren Ergebnissen. Gleichzeitig ist der Aufwand für die Berechnung noch gering.
Interpolation mit Polynomen höherer Ordnung, beispielsweise die kubische Interpolation, liefern zwar teilweise noch bessere Ergebnisse. Allerdings steigt damit auch der Berechnungsaufwand an, so daß in dieser Arbeit die lineare Interpolation verwendet wird.

2.4 Filter

Im Kapitel Registrierung finden Filter Verwendung. Zwei von ihnen werden hier kurz erläutert.

2.4.1 Lee-Filter

Der Lee-Filter [17] stellt eine einfache Möglichkeit dar, das Rauschen in einem Bild zu reduzieren und gleichzeitig Kanten zu erhalten.
Sei $b = (\mathcal{T}, \mathcal{G}, I)$, $\mathcal{T} \subseteq \mathbb{R}^3$ ein Bild. Das gefilterte Bild ist $b_{Lee} = (\mathcal{T}, \mathcal{G}, I_{Lee})$ mit

$$I_{Lee}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N} I(\mathbf{x}_i)\ . \tag{2.3}$$

Dabei bezeichnen die $\mathbf{x}_i$ die Nachbarn des Voxels $\mathbf{x}$ bezüglich der 26-er Nachbarschaft. Es werden aber nur die Nachbarn $\mathbf{x}_i$ betrachtet, für die

$$|I(\mathbf{x}_i) - I(\mathbf{x})| \leq t$$

mit einem noch zu bestimmenden Wert t gilt. N bezeichnet die Anzahl derartiger Nachbarn.
Zur Bestimmung von t werden von den Grauwerten im Bild b der Mittelwert und die Varianz σ^2 berechnet. t wird als

$$t = 2\sqrt{\sigma^2} \tag{2.4}$$

gleich der doppelten Standardabweichung gesetzt.

2.4.2 Anisotrope Diffusion

Anisotrope Diffusionsfilter spielen bei der Verbesserung von Bildern eine bedeutende Rolle. Mit ihrer Hilfe ist es möglich, vorhandenes Rauschen zu reduzieren, aber gleichzeitig Kanten zu erhalten und zu verstärken. Die folgende Beschreibung stützt sich auf die Arbeiten von J. Weickert [37, 39].

Das allgemeine Modell

Sei $b \in \mathcal{B}_{\mathcal{T}}$ ein n-dimensionales Bild mit Trägermenge $\mathcal{T} = (0, a_1) \times \cdots \times (0, a_n)$. Das Ergebnis der Filterung ist ein Bild $(\mathcal{T} \times (0, \infty), \mathbb{R}, u)$, das die Lösung des folgenden Differentialgleichungssystems[4] ist.

$$\begin{aligned} \frac{\partial u}{\partial t} &= \operatorname{div}\left(D(\nabla u_\sigma)\nabla u\right) && \text{auf} \quad \mathcal{T} \times (0, \infty) && (2.5) \\ u(\mathbf{x}, 0) &= I(\mathbf{x}) && \text{auf} \quad \mathcal{T} && (2.6) \\ \langle D(\nabla u_\sigma)\nabla u, \mathbf{n}\rangle &= 0 && \text{auf} \quad \Gamma \times (0, \infty) && (2.7) \end{aligned}$$

Dabei bezeichnet $\langle ., . \rangle$ das gewöhnliche Skalarprodukt, $\mathbf{n}$ die Normale und Γ den Rand von $\mathcal{T}$. Der Diffusionstensor $D \in \mathbb{R}^{n \times n}$ ist eine Funktion des Gradienten ∇u_σ, wobei

$$u_\sigma(\mathbf{x}, t) := \left(K_\sigma * \tilde{u}(., t)\right)(\mathbf{x}) \qquad (\sigma \geq 0) \tag{2.8}$$

aus $\tilde{u}$ durch Faltung mit einem Gaußkern entsteht, und $\tilde{u}$ ist die Erweiterung von u von $\mathcal{T}$ auf $\mathbb{R}^n$, die durch Spiegelung an Γ gebildet wird.
Entsprechend [37] wird $D(\nabla u_\sigma)$ gewählt. Sei $\{\mathbf{v}_1, \ldots, \mathbf{v}_n\}$ eine orthonormale Basis des $\mathbb{R}^n$ mit $\mathbf{v}_1 \| \nabla u_\sigma$. Dann wird die Matrix $D(\nabla u_\sigma)$ so konstruiert, daß sie symmetrisch und positiv definit ist und $\mathbf{v}_1, \ldots, \mathbf{v}_n$ Eigenvektoren zu den Eigenwerten $\lambda_1, \ldots, \lambda_n$ sind. Die Eigenwerte sind Funktionen von $\|\nabla u_\sigma\|$. Daraus folgt[5]:

$$D(\nabla u_\sigma) = (\mathbf{v}_1 | \ldots | \mathbf{v}_n) \operatorname{diag}\left(\lambda_1(\|\nabla u_\sigma\|), \ldots, \lambda_n(\|\nabla u_\sigma\|)\right) (\mathbf{v}_1 | \ldots | \mathbf{v}_n)^T \tag{2.9}$$

Die Eigenwerte werden als

$$\begin{aligned} \lambda_1(s) &= \exp\left(\frac{-s^\alpha}{\alpha \lambda^\alpha}\right) \\ \lambda_2(s) &= 1 \\ &\vdots \\ \lambda_n(s) &= 1 \end{aligned}$$

mit einem Kontrastparameter $\lambda > 0$ und einem Verzögerungsparameter $\alpha \geq 1$ gesetzt. Im folgenden wird $\alpha = 5$ gewählt, so daß die Filterung von den Parametern λ und σ abhängig ist. Gradienten, deren Norm größer als λ ist, werden als Kanten betrachtet, und die Diffusion wird reduziert. Details kleiner als σ werden als Rauschen behandelt und entfernt.

[4] In diesen Gleichungen bezeichnet ∇u den Gradient in $\mathcal{T}$

[5] $(\mathbf{v}_1 | \ldots | \mathbf{v}_n)$ bezeichnet eine Matrix, deren Spalten die Vektoren $\mathbf{v}_1, \ldots, \mathbf{v}_n$ bilden

Kapitel 3

Registrierung

Im vorliegenden Fall werden von einer Person zwei Kernspinbilder, jeweils ein T_1- und ein PD-gewichtetes, aufgenommen, und in jedem Bild wird ein Koordinatensystem definiert. Um später für die Segmentierung die Informationen ausnutzen zu können, die sich zusammen aus beiden Bildern ergeben, ist es notwendig, eine räumliche Transformation zu finden, die eines der beiden Bilder auf das andere abbildet. Diese Aufgabe wird durch die Registrierung erledigt. Im folgenden soll eine räumliche Transformation ermittelt werden, die das PD-gewichtete Bild $b_{PD} \in \mathcal{B}_{\mathcal{T}_{PD}}$ auf das T_1-gewichtete[1] $b_{T_1} \in \mathcal{B}_{T_{\mathcal{T}_1}}$ abbildet.

3.1 Bestandteile von Registrieralgorithmen

Für einen Registrieralgorithmus sind die drei folgenden Bestandteile erforderlich [16]:

1. Eine Abbildung $T_{\mathcal{T}_2} : \mathcal{B}_{\mathcal{T}_1} \times \mathbb{R}^m \to \mathcal{B}_{\mathcal{T}_2}$, die ein Bild $b \in \mathcal{B}_{\mathcal{T}_1}$ in Abhängigkeit von Parametern transformiert[2]. Die Transformation soll durch m Parameter beeinflußbar sein. Das transformierte PD-gewichtete Bild $b_{PD_t} \in \mathcal{B}_{\mathcal{T}_{T_1}}$ ergibt sich aus dem ursprünglichem Bild $b_{PD} \in \mathcal{B}_{\mathcal{T}_{PD}}$ durch Anwendung der Transformation mit den Parametern $\mathbf{x} \in \mathbb{R}^m$ gemäß Gleichung (3.1).

$$b_{PD_t} = T_{\mathcal{T}_{T_1}}(b_{PD}, \mathbf{x}) \quad \mathbf{x} \in \mathbb{R}^m \tag{3.1}$$

2. Eine Kostenfunktion $C : \mathcal{B}_{\mathcal{T}} \times \mathcal{B}_{\mathcal{T}} \to \mathbb{R}$, die bewertet, wie gut zwei Bilder aneinander angepaßt sind. Bei einer optimalen Anpassung besitzt die Kostenfunktion ein globales Maximum. Im betrachteten Fall wird die Anpassung zwischen dem Referenzbild $b_{T_1} \in \mathcal{B}_{\mathcal{T}_{T_1}}$ und dem transformierten Bild $b_{PD_t} \in \mathcal{B}_{\mathcal{T}_{T_1}}$ bewertet.

$$C = C(b_{T_1}, b_{PD_t}) \tag{3.2}$$

[1] Dieses Bild heißt auch Referenzbild.

[2] $\mathbb{R}^m$ bezeichnet den Raum der Parameter und steht in keiner Beziehung zur Trägermenge $\mathcal{T} \subseteq \mathbb{R}^n$

3. Ein Verfahren, das die Kostenfunktion in Abhängigkeit der Transformationsparameter maximiert. Während einer Registrierung sind die Bilder b_{T_1} und b_{PD} konstant. Das Ergebnis der Kostenfunktion ist demnach nur von den Parametern $\mathbf{x} \in \mathbb{R}^m$ abhängig. Das Einsetzen von Gleichung (3.1) in die Gleichung für die Kostenfunktion liefert als zu maximierende Funktion:

$$C_{b_{T_1},b_{PD}}(\mathbf{x}) = C(b_{T_1}, T_{\mathcal{T}_{T_1}}(b_{PD}, \mathbf{x})) \tag{3.3}$$

Es soll ein $\mathbf{x}_0 \in \mathbb{R}^m$ so bestimmt werden, daß die Funktion $C(\mathbf{x})$ maximiert wird. Dieses $\mathbf{x}_0$ ist durch die Gleichung (3.4) bestimmt.

$$\mathbf{x}_0 = \arg\max_{\mathbf{x}\in\mathbb{R}^m} C(\mathbf{x}) \tag{3.4}$$

3.2 Die Abbildung

Die zu registrierenden Bilder sind MR-Tomographien vom Schädel, einem Körperteil, das in erster Näherung als rigid angenommen werden kann. Durch die zeitlich getrennten Aufnahmen ist allerdings mit einer unterschiedlichen Position des Kopfes in den Bildern zu rechnen. Um diese zu korrigieren, werden Rotation und Translation benötigt.
Eine Eigenschaft des verwendeten Aufnahmeverfahrens ist es, zu nichtlinearen räumlichen Verzerrungen in den Bildern zu führen. Die Ursachen für diese Verzerrungen liegen in den physikalischen Grundlagen der Kernspintomographie. Um diese Verzerrungen näherungsweise zu korrigieren, wird zusätzlich zu Rotation und Translation eine lineare Skalierung entlang der Koordinatenachsen ausgeführt. Das führt auf eine affine Abbildung [41].
Diese Transformation setzt sich als Nacheinanderausführung einer Skalierung s, einer Rotation r und einer Translation t wie folgt zusammen:

$$T = t \circ r \circ s \ . \tag{3.5}$$

Zuerst erfolgt eine Skalierung parallel zu den Koordinatenachsen. Die Parameter für die Skalierung sind s_x, s_y und s_z. In homogenen Koordinaten hat die Skalierung die Matrixdarstellung gemäß Gleichung (3.6).

$$s(s_x, s_y, s_z) = \begin{pmatrix} s_x & 0 & 0 & 0 \\ 0 & s_y & 0 & 0 \\ 0 & 0 & s_z & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.6}$$

Daran schließt sich eine Rotation des Bildes an. Sie setzt sich aus drei einzelnen Rotationen zusammen: eine um die x-Achse mit Rotationswinkel ϕ_x, um die y-Achse mit Rotationswinkel ϕ_y und um die z-Achse mit Winkel ϕ_z.

$$r(\phi_x, \phi_y, \phi_z) = r_z(\phi_z) \circ r_y(\phi_y) \circ r_x(\phi_x) \tag{3.7}$$

Die einzelnen Rotationen haben die folgenden Matrixdarstellungen in homogenen Koordinaten:

$$r_x(\phi_x) = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi_x & -\sin\phi_x & 0 \\ 0 & \sin\phi_x & \cos\phi_x & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.8}$$

$$r_y(\phi_y) = \begin{pmatrix} \cos\phi_y & 0 & -\sin\phi_y & 0 \\ 0 & 1 & 0 & 0 \\ \sin\phi_y & 0 & \cos\phi_y & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.9}$$

$$r_z(\phi_z) = \begin{pmatrix} \cos\phi_z & -\sin\phi_z & 0 & 0 \\ \sin\phi_z & \cos\phi_z & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.10}$$

Als letzter Schritt der Abbildung erfolgt die Translation. Sie ist auch von drei Parametern abhängig: t_x für die Translation in x-Richtung, analog t_y und t_z. Die Matrixdarstellung in homogenen Koordinaten ist:

$$t(t_x, t_y, t_z) = \begin{pmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.11}$$

Damit ist die Transformation durch neun Parameter beeinflußbar: den drei Rotationswinkeln ϕ_x, ϕ_y, ϕ_z, den drei Skalierungsparametern s_x, s_y, s_z und den Translationsparametern t_x, t_y und t_z.

Das Einsetzen der Gleichungen (3.6) bis (3.11) in die Gleichung (3.5) führt auf die folgende Matrixdarstellung in homogenen Koordinaten für die Abbildung:

$$T(t_x, t_y, t_z, \phi_x, \phi_y, \phi_z, s_x, s_y, s_z) = \begin{pmatrix} a_{11} & a_{12} & a_{13} & t_x \\ a_{21} & a_{22} & a_{23} & t_y \\ a_{31} & a_{32} & a_{33} & t_z \\ 0 & 0 & 0 & 1 \end{pmatrix} \tag{3.12}$$

mit

$$
\begin{aligned}
a_{11} &= s_x \cos\phi_y \cos\phi_z \ , \\
a_{12} &= -s_y \left(\sin\phi_x \sin\phi_y \cos\phi_z + \cos\phi_x \sin\phi_z\right) \ , \\
a_{13} &= s_z \left(\sin\phi_x \sin\phi_z - \cos\phi_x \sin\phi_y \cos\phi_z\right) \ , \\
a_{21} &= s_x \cos\phi_y \sin\phi_z \ , \\
a_{22} &= s_y \left(\cos\phi_x \cos\phi_z - \sin\phi_x \sin\phi_y \sin\phi_z\right) \ , \\
a_{23} &= -s_z \left(\cos\phi_x \sin\phi_y \sin\phi_z + \sin\phi_x \cos\phi_z\right) \ , \\
a_{31} &= s_x \sin\phi_y \ , \\
a_{32} &= s_y \sin\phi_x \cos\phi_y \ , \\
a_{33} &= s_z \cos\phi_x \cos\phi_y \ .
\end{aligned}
$$

3.3 Die Kostenfunktion

Die Kostenfunktion hat im Registrieralgorithmus eine zentrale Bedeutung. Sie ist eine Funktion $C : \mathcal{B}_\mathcal{T} \times \mathcal{B}_\mathcal{T} \to \mathbb{R}$ und hat die Aufgabe zu bewerten, wie gut zwei Bilder aneinander angepaßt sind. Je besser die Anpassung ist, umso größer soll der Funktionswert werden. Darum muß sie die Eigenschaft

$$\forall b_1, b_2 \in \mathcal{B}_\mathcal{T} : C(b_1, b_2) \leq C(b_1, b_1) \tag{3.13}$$

besitzen.
Es wird eine Kostenfunktion auf Basis der *Mutual Information* verwendet. Mutual Information ist ein voxelbasiertes Verfahren, das heißt, es zieht zur Bewertung alle Voxel in beiden Bildern heran. Ansätze dieser Art wurden von Collignon [6] und später darauf aufbauend von Wells [40] und Maes [22] präsentiert. Im folgenden wird der Ansatz von Maes et al. verwendet.
Gegeben seien zwei Bilder $b_1, b_2 \in \mathcal{B}_\mathcal{T}$ mit Histogrammen h_{b_1}, h_{b_2} und zweidimensionalem Scatterplot h_{b_1,b_2}. Weiterhin sollen beide Bilder dieselbe Grauwertmenge $\mathcal{G}$ besitzen.
Beide Bilder haben N Voxel mit

$$N = \sum_{g \in \mathcal{G}} h_{b_1}(g) \tag{3.14}$$

$$= \sum_{g \in \mathcal{G}} h_{b_2}(g) \tag{3.15}$$

Jedem dieser Bilder wird eine Zufallsvariable X_{b_1} bzw. X_{b_2} zugeordnet, mit denen die Wahrscheinlichkeit ausgedrückt wird, daß ein Voxel einen bestimmten Grauwert hat.
Die Wahrscheinlichkeit $P_{b_i}(g)$, daß ein Voxel im Bild $b_i, i \in \{1, 2\}$ den Grauwert $g \in \mathcal{G}$ hat, ist definiert durch:

$$P_{b_i}(g) := P(X_{b_i} = g) = \frac{h_{b_i}(g)}{N} \tag{3.16}$$

Die Wahrscheinlichkeit $P_{b_1,b_2}(g_1, g_2)$, daß ein Voxel im Bild b_1 den Grauwert g_1 und im Bild b_2 den Grauwert g_2 hat, ist definiert durch:

$$P_{b_1,b_2}(g_1, g_2) := P((X_{b_1} = g_1) \wedge (X_{b_2} = g_2)) = \frac{h_{b_1,b_2}(g_1, g_2)}{N} \tag{3.17}$$

Bei der Abhängigkeit der Zufallsvariablen X_{b_1} und X_{b_2} können zwei Extremfälle betrachtet werden:

1. Der Fall vollständiger statistischer Unabhängigkeit. In dem Fall gilt:

$$P_{b_1,b_2}(g_1, g_2) = P_{b_1}(g_1) P_{b_2}(g_2) \tag{3.18}$$

2. Der Fall vollständiger statistischer Abhängigkeit. Dann existiert eine Funktion $f : \mathcal{G} \to \mathcal{G}$, so daß gilt:

$$P_{b_1,b_2}(g_1, f(g_1)) = P_{b_1}(g_1) \tag{3.19}$$

Als Maß der Anpassungsgüte wird der Abstand zwischen der Verteilung P_{b_1,b_2} und der Verteilung im Fall einer vollständigen Unabhängigkeit gemessen. Dies geschieht durch ein *Kullback-Leibler*-Maß. Damit ist die Mutual Information eine Funktion $MI : \mathcal{B}_\mathcal{T} \times \mathcal{B}_\mathcal{T} \to \mathbb{R}$, die wie folgt definiert ist:

$$MI(b_1, b_2) := \sum_{g_1} \sum_{g_2} P_{b_1,b_2}(g_1, g_2) \log_2 \frac{P_{b_1,b_2}(g_1, g_2)}{P_{b_1}(g_1) P_{b_2}(g_2)} \tag{3.20}$$

Das Einsetzen der Gleichungen (3.14) bis (3.19) in die Gleichung (3.20) und eine anschließende Umformung unter Verwendung der Beziehungen

$$\sum_{g_2} h_{b_1,b_2}(g_1, g_2) = h_{b_1}(g_1)$$

und

$$\sum_{g_1} h_{b_1,b_2}(g_1, g_2) = h_{b_2}(g_2)$$

führt auf:

$$MI(b_1, b_2) = H(b_1) + H(b_2) - H(b_1, b_2) \tag{3.21}$$

wobei $H(b_1), H(b_2), H(b_1, b_2)$ die Entropien sind mit:

$$H(b_i) = -\sum_{g \in \mathcal{G}} P_{b_i}(g) \log_2 P_{b_i}(g) \,, \tag{3.22}$$

$$H(b_1, b_2) = -\sum_{g_1 \in \mathcal{G}} \sum_{g_2 \in \mathcal{G}} P_{b_1,b_2}(g_1, g_2) \log_2 P_{b_1,b_2}(g_1, g_2) \,. \tag{3.23}$$

Drei wichtige Eigenschaften der Mutual Information sind in Tabelle 3.1 zusammengefaßt.

Selbstinformation	$\forall b \in \mathcal{B}_\mathcal{T} : MI(b,b) = H(b)$
Symmetrie	$\forall b_1, b_2 \in \mathcal{B}_\mathcal{T} : MI(b_1,b_2) = MI(b_2,b_1)$
Beschränktheit	$\forall b_1, b_2 \in \mathcal{B}_\mathcal{T} : MI(b_1,b_2) \leq \min(H(b_1), H(b_2))$

Tabelle 3.1: Wichtige Eigenschaften von Mutual Information

Damit kann die Kostenfunktion $C : \mathcal{B}_\mathcal{T} \times \mathcal{B}_\mathcal{T} \rightarrow \mathbb{R}$ auf Basis von Mutual Information wie folgt definiert werden:

$$C(b_1, b_2) := MI(b_1, b_2) \; . \tag{3.24}$$

Aufgrund der Eigenschaften der Mutual Information hat die Kostenfunktion die geforderte Eigenschaft $\forall b_1, b_2 \in \mathcal{B}_\mathcal{T} : C(b_1,b_2) \leq C(b_1,b_1)$.

Beweis 3.1 *Seien $b_1, b_2 \in \mathcal{B}_\mathcal{T}$ zwei Bilder. Aufgrund der Beschränktheit gilt:*

$$C(b_1, b_2) = MI(b_1, b_2) \leq \min\left(H(b_1), H(b_2)\right) \; .$$

Weiterhin gilt

$$\min\left(H(b_1), H(b_2)\right) \leq H(b_1)$$

und aufgrund der Selbstinformation

$$H(b_1) = MI(b_1, b_1) = C(b_1, b_1) \; .$$

Zusammengefaßt folgt daraus

$$C(b_1, b_2) \leq C(b_1, b_1) \; .$$

q.e.d.

3.4 Optimierung

Der dritte Bestandteil eines Registrieralgorithmus ist die Optimierung. Ausgehend von dem Bild b_{PD} ergibt sich das transformierte PD-gewichtete Kernspinbild $b_{PD_t} \in \mathcal{B}_{\mathcal{T}_{T_1}}$ durch die Transformation nach Gleichung (3.1) unter Anwendung der Abbildung aus Gleichung (3.12) in Abhängigkeit von neun Transformationsparametern. Das Einsetzen von Gleichung (3.1) und der Gleichung (3.24) für die Kostenfunktion in Gleichung (3.3) führt auf die folgende zu maximierende Funktion:

$$C_{b_{T_1}, b_{PD}}(\mathbf{x}) = MI(b_{T_1}, T_{\mathcal{T}_{T_1}}(b_{PD}, \mathbf{x})) \; . \tag{3.25}$$

Die Aufgabe des Optimierers ist es, ein $\mathbf{x}_0 \in \mathbb{R}^m$ mit

$$\forall \mathbf{x} \in \mathbb{R}^m : C(\mathbf{x}_0) \geq C(\mathbf{x}) \tag{3.26}$$

zu finden. Dieses $\mathbf{x}_0$ ist durch

$$\mathbf{x}_0 = \arg\max_{\mathbf{x} \in \mathbb{R}^m} C(\mathbf{x}) \tag{3.27}$$

bestimmt.
Im folgenden werden drei Verfahren vorgestellt, um $\mathbf{x}_0$ zu bestimmen.

3.4.1 Downhill Simplex Algorithmus

Der Downhill-Simplex-Algorithmus ist ein einfaches Verfahren, um Funktionen der Art $C : \mathbb{R}^m \to \mathbb{R}$ zu maximieren. Der Grundgedanke des Verfahrens ist es, einen geometrischen Körper, ein Simplex, mit $(m+1)$ Punkten $\mathbf{v}_1, \ldots, \mathbf{v}_{m+1} \in \mathbb{R}^m$ zu definieren. In Abhängigkeit der Funktionswerte $C(\mathbf{v}_i)$, $i = 1, \ldots, (m+1)$ werden die Punkte $\mathbf{v}_i$ solange ersetzt, bis eine Abbruchbedingung erreicht ist. Das Optimierungsergebnis ist der Punkt mit dem größten Funktionswert.

Downhill-Simplex-Algorithmus zum Maximieren einer Funktion $C : \mathbb{R}^m \to \mathbb{R}$

Start mit Parametern $\mathbf{P}_0, d_{start}, d_{stop}, \lambda_1, \ldots, \lambda_m$*;*

$d = d_{start}$;

$\mathbf{t} = \mathbf{P}_0$;

WHILE $(d \geq d_{stop})$ DO

{

Initialisierung:

$$\begin{aligned} \mathbf{v}_1 &= \mathbf{t} \\ \mathbf{v}_{i+1} &= \lambda_i \cdot d \cdot \mathbf{e}_i + \mathbf{v}_i \quad \text{für} \quad i = 1, \ldots, m \end{aligned}$$

DO

{

Bestimmung eines $\sigma \in S_{m+1}$ *mit* $C(\mathbf{v}_{\sigma(i)}) \leq C(\mathbf{v}_{\sigma(i+1)})$ *für alle* $i = 1, \ldots, m$

FOR $i = 1, \ldots, m$ DO

IF $(C(\text{refl}(\mathbf{v}_{\sigma(i)})) > C(\mathbf{v}_{\sigma(i)}))$ THEN

{

$\mathbf{v}_{\sigma(i)} = \text{refl}(\mathbf{v}_{\sigma(i)})$;

Verlassen der FOR-*Schleife*;

}

}

WHILE (*Im letzten Schleifendurchlauf wurde ein* $\mathbf{v}_i$ *ersetzt*)

$\mathbf{t} = \mathbf{v}_{\sigma(m+1)}$

$d = d/2$;

}

Ende mit Optimierungsergebnis $\mathbf{t}$

Abbildung 3.1: Der verwendete Downhill-Simplex-Algorithmus

In [29] wird ein Downhill-Simplex-Verfahren vorgestellt. Dieses hat sich jedoch als ungeeignet erwiesen, um damit die vorgestellte Kostenfunktion zu maximie-

ren. Es wird aber als Ausgangspunkt genommen, um daraus ein Verfahren auf Basis der Freudenthal-Triangulierung [1] zu entwickeln.
Bei dieser Triangulierung ist das anfängliche Simplex durch eine affine Abbildung $A : \mathbb{R}^m \to \mathbb{R}^m$ definiert, die die Punkte $\mathbf{a}_1, \ldots, \mathbf{a}_{m+1}$ mit

$$\begin{aligned} \mathbf{a}_1 &= 0 \\ \mathbf{a}_{i+1} &= \mathbf{a}_i + \mathbf{e}_i \qquad i = 1, \ldots, m \end{aligned}$$

auf die Punkte $\mathbf{v}_1, \ldots \mathbf{v}_{m+1}$ durch $\mathbf{v}_i = A(\mathbf{a}_i)$ abbildet. $\mathbf{e}_i \in \mathbb{R}^m$ bezeichnet den i-ten Einheitsvektor.
Weiterhin werden für jeden Punkt $\mathbf{v}_i$ Vorgänger $\mathrm{pre}(\mathbf{v}_i)$ und Nachfolger $\mathrm{suc}(\mathbf{v}_i)$ wie folgt definiert:

$$\mathrm{suc}(\mathbf{v}_i) := \begin{cases} \mathbf{v}_{i+1} & \text{für} \quad i = 1, \ldots, m \\ \mathbf{v}_1 & \text{für} \quad i = m+1 \end{cases} \tag{3.28}$$

$$\mathrm{pre}(\mathbf{v}_i) := \begin{cases} \mathbf{v}_{i-1} & \text{für} \quad i = 2, \ldots, (m+1) \\ \mathbf{v}_{m+1} & \text{für} \quad i = 1 \end{cases} \tag{3.29}$$

Die Reflektion von $\mathbf{v}_i$ in der Triangulierung ist durch

$$\mathrm{refl}(\mathbf{v}_i) := \mathrm{pre}(\mathbf{v}_i) - \mathbf{v}_i + \mathrm{suc}(\mathbf{v}_i) \tag{3.30}$$

erklärt.

Implementierung

Die Implementierung (Abbildung 3.1) verwendet mehrere Optimierungsdurchläufe. Ein Durchlauf läßt sich mit den folgenden Parametern steuern: einem Startpunkt $\mathbf{P}_0 \in \mathbb{R}^m$, der Gitterweite der Triangulierung $d \in \mathbb{R}$ und m Skalierungsfaktoren $\lambda_1, \ldots, \lambda_m$. Das ursprüngliche Simplex $\mathbf{v}_1, \ldots, \mathbf{v}_{m+1}$ ist durch

$$\mathbf{v}_1 = \mathbf{P}_0 \tag{3.31}$$

$$\mathbf{v}_{i+1} = \lambda_i \cdot d \cdot \mathbf{e}_i + \mathbf{v}_i \qquad i = 1, \ldots, m \tag{3.32}$$

definiert.
Beginnend mit diesen Werten von $\mathbf{v}_1$ bis $\mathbf{v}_{m+1}$ wird die Kostenfunktion C für die Argumente $\mathbf{v}_i$ ausgewertet. Aus dem Ergebnis läßt sich eine Permutation $\sigma \in S_{m+1}$ der natürlichen Zahlen $1, ..., (m+1)$ ableiten, so daß für alle $i = 1, ..., m$ gilt: $C(\mathbf{v}_{\sigma(i)}) \leq C(\mathbf{v}_{\sigma(i+1)})$. Beginnend mit $i = 1$ wird getestet, ob $C\left(\mathrm{refl}(\mathbf{v}_{\sigma(i)})\right) > C(\mathbf{v}_{\sigma(i)})$. Ist das der Fall, wird $\mathbf{v}_{\sigma(i)}$ durch $\mathrm{refl}(\mathbf{v}_{\sigma(i)})$ ersetzt. Ansonsten wird i um eins erhöht und der Test erneut durchgeführt. Ist der Test auch für $i = m+1$ fehlgeschlagen, wird der Durchlauf beendet und das Optimierungsergebnis ist $\mathbf{v}_{\sigma(m+1)}$.
Die Optimierung mit dem Downhill-Simplex-Algorithmus verwendet mehrere Optimierungsdurchläufe mit verschiedenen Gitterweiten d. Begonnen wird mit

$d = d_{start}$. Ist das Optimierungsergebnis erreicht, wird d halbiert. Ist die neue Gitterweite d größer als d_{stop}, wird ein weiterer Durchlauf mit dieser und einem Startpunkt, der gleich dem Ergebnis des vorherigen Durchlaufs ist, gestartet. Das Optimierungsergebnis des Downhill-Simplex-Algorithmus ist das Ergebnis des letzten Optimierungsdurchlaufs.
Damit läßt sich der gesamte Algorithmus durch $m + 3$ Parameter steuern: der Startgitterweite d_{start}, der Stoppgitterweite d_{stop}, dem Startpunkt $\mathbf{P}_0$ und den Skalierungsfaktoren $\lambda_1, \ldots, \lambda_m$.

3.4.2 Genetische Optimierung

Staib und Lei stellen in [33] ein Registrierungsverfahren vor, bei dem eine kantenbasierte Kostenfunktion mit Hilfe eines genetischen Algorithmus optimiert wird. Ihr Ansatz dient als Ausgangspunkt, um damit die vorgestellte Kostenfunktion zu maximieren. Die hier verwendete genetische Optimierung ist ein Bestandteil der Brian-Software [17].

Das Prinzip eines Genetischen Algorithmus

Die typische Struktur eines genetischen Algorithmus ist in Abbildung 3.2 dargestellt. Zunächst wird eine Anfangspopulation gebildet. Sie hat, wie auch jede neu entstehende Population, eine vorgegebene Anzahl (N_{IP}) von Individuen. Jedes Individuum hat wiederum eine bestimmte Anzahl von Genen, in denen dessen Eigenschaften gespeichert sind. Weiterhin gehören zu dem Algorithmus drei Operatoren: Selektion, Kreuzung und Mutation. Die Selektion bildet den Anfang einer neuen Population durch Auswahl der am besten angepaßten Individuen. Kreuzung und Mutation sind den entsprechenden Vorgängen von Genmutation und -kreuzung in der Natur nachempfunden. Kreuzung bildet aus zwei Individuen ein neues. Jedes Gen des neues Individuums entsteht aus den Genen der beiden anderen. Die Mutation verändert zufällig eines oder mehrere Gene eines Individuums. Durch Anwendung dieser drei Operationen werden aus der Ausgangspopulation schrittweise die nachfolgenden Populationen gebildet.
Der ganze Zyklus wird solange durchlaufen bis eine Abbruchbedingung erfüllt ist. Mögliche Abbruchbedingungen sind beispielsweise das Erreichen einer maximalen Anzahl von Populationen oder eine zu geringe Verbesserung der Kostenfunktion zwischen zwei aufeinanderfolgenden Populationen. Das Optimierungsergebnis ist in jedem Fall das am besten angepaßte Individuum.

Implementierung

Sei die Funktion $C : \mathcal{X} \to \mathbb{R}$, $\mathcal{X} = [X_{1_{min}}, X_{1_{max}}] \times \cdots \times [X_{m_{min}}, X_{m_{max}}] \subset \mathbb{R}^m$, zu maximieren. Jedes Argument der Funktion wird durch ein einzelnes Gen kodiert. Ein Gen wird durch einen Bitstring der Länge l repräsentiert. Um die Kostenfunktion für ein Individuum auszuwerten, müssen die Argumente, die durch die Gene dargestellt sind, dekodiert werden. Sei das Argument X_i durch das Gen $G_i = (G_i)_1, (G_i)_2, \ldots, (G_i)_l$ repräsentiert. Dann ergibt sich X_i durch Anwendung der Dekodierungsfunktion $D : \{0,1\}^l \times \mathbb{R}^2 \to [X_{i_{min}}, X_{i_{max}}]$, die

Genetischer Algorithmus

Erzeugen der Anfangspopulation $P_0 = (I_1, \ldots, I_{N_{IP}})$

t = 0;

WHILE (*Abbruchbedingung nicht erfüllt*) DO

{

$P^{(1)}_{(t+1)}$ = Selektion (P_t);
$P^{(2)}_{(t+1)}$ = Kreuzung $\left(P^{(1)}_{(t+1)}\right)$;
$P^{(3)}_{(t+1)}$ = Mutation $\left(P^{(2)}_{(t+1)}\right)$;
$P_{(t+1)} = P^{(3)}_{(t+1)}$;
t = t + 1;

}

Abbildung 3.2: Typische Struktur eines genetischen Algorithmus

folgendermaßen definiert ist:

$$D((g_1, g_2, \ldots, g_l), X_{i_{min}}, X_{i_{max}}) := X_{i_{min}} + \frac{X_{i_{max}} - X_{i_{min}}}{2^l - 1} \sum_{k=1}^{l} g_k \cdot 2^{(l-k)} \ . \tag{3.33}$$

Soll die Angepaßtheit eines Individuums mit den Genen $G_1, G_2, \ldots, G_m$ bewertet werden, so ist die Kostenfunktion $C(X_1, X_2, \ldots, X_m)$ mit Argumenten $X_i = D(G_i, X_{i_{min}}, X_{i_{max}})$, $i = 1, \ldots, m$ auszuwerten. Die Angepaßtheit soll dabei proportional zum Funktionswert sein. Wenn I_1 und I_2 zwei Individuen mit Genen $G_{1_1}, \ldots, G_{1_m}$ bzw. $G_{2_1}, \ldots, G_{2_m}$ sind, dann ist I_1 besser angepaßt als I_2, wenn $C(X_{1_1}, \ldots, C_{1_m}) > C(X_{2_1}, \ldots, X_{2_m})$ mit $X_{i_j} = D(G_{i_j}, X_{j_{min}}, X_{j_{max}})$, $i \in \{1, 2\}, j = 1, \ldots, m$.
Auf dieser Basis werden die drei Operatoren Selektion, Kreuzung und Mutation implementiert. Die Selektion bewertet die Angepaßtheit jedes Individuums in der Population und wählt die N_{IN} Individuen aus, die am besten angepaßt sind. Die Kreuzung bildet aus zwei Individuen I_1 und I_2 mit Genen $G_{1_1}, \ldots, G_{1_m}$ bzw. $G_{2_1}, \ldots, G_{2_m}$ ein Individuum I_3 mit Genen $G_{3_1}, \ldots, G_{3_m}$, wobei für jedes Gen G_{3_i} gilt: $G_{3_i} = G_{1_i}$ oder $G_{3_i} = G_{2_i}$ oder es existiert ein j mit $1 < j < l$, so daß die ersten j Bits von G_{1_i} und G_{3_i} identisch sind und die restlichen von G_{2_i} und G_{3_i}. Die Mutation verändert zufällig die Gene eines Individuums. Dabei wird pro Gen höchstens ein Bit verändert.
Der implementierte Algorithmus läßt sich über vier Parameter steuern: der Anzahl N_{IP} von Individuen in einer Population, der maximalen Anzahl von Populationen N_P, der Anzahl N_{IN} der Individuen, die die Selektionsopera-

tion in die nächste Population übernimmt, und der Mutationsrate r_m, die die Wahrscheinlichkeit bestimmt, mit der ein Bit in einem Gen verändert wird. Als Abbruchbedingung wird das Erreichen der maximalen Anzahl von Populationen verwendet. Das Optimierungsergebnis ist das am besten angepaßte Individuum der letzten Population. Seine Gene werden dekodiert und als Parameter für die Transformation verwendet.

3.4.3 Powells Algorithmus

Maes et al. verwenden in [22] einen Powells-Algorithmus zur Optimierung der Kostenfunktion. Dieser Algorithmus wird auch in [29] ausführlich vorgestellt. Zu maximieren ist wieder die Funktion $C : \mathcal{X} \to \mathbb{R}$, $\mathcal{X} = [X_{1_{min}}, X_{1_{max}}] \times \cdots \times [X_{m_{min}}, X_{m_{max}}] \subset \mathbb{R}^m$, die auf einer abgeschlossenen Teilmenge des $\mathbb{R}^m$ definiert ist. Die Struktur von Powells Algorithmus ist in Abbildung 3.3 dargestellt. Die Optimierung erfolgt ausgehend von einem Startpunkt $\mathbf{P}_0 \in \mathbb{R}^m$ entlang der Optimierungsrichtungen $\mathbf{r}_i \in \mathbb{R}^m$, die in jedem Durchlauf neu gesetzt werden.

Powells-Algorithmus zum Maximieren der Funktion $C : \mathcal{X} \to \mathbb{R}$

Start mit Parameter $\mathbf{P}_0$

FOR $i = 1, \ldots, m$ DO $\mathbf{r}_i = \mathbf{e}_i$;

DO

{

FOR $i = 1, \ldots, m$ DO

{

$\mathbf{P}_i = \arg\max_{\mathbf{x}}\{C(\mathbf{x}) \mid \mathbf{x} \in (\{\mathbf{P}_{i-1} + \lambda \mathbf{r}_i \mid \lambda \in \mathbb{R}\} \cap \mathcal{X})\}$;

}

$\mathbf{P}_{m+1} = \arg\max_{\mathbf{x}}\{C(\mathbf{x}) \mid \mathbf{x} \in (\{\mathbf{P}_m + \lambda(\mathbf{P}_m - \mathbf{P}_0) \mid \lambda \in \mathbb{R}\} \cap \mathcal{X})\}$;

FOR $i = 1, \ldots, (m-1)$ DO $\mathbf{r}_i = \mathbf{r}_{i+1}$;

$\mathbf{r}_m = \mathbf{P}_{m+1} - \mathbf{P}_0$;

$\varepsilon = \|\mathbf{P}_{m+1} - \mathbf{P}_0\|_2$;

$\mathbf{P}_0 = \mathbf{P}_{m+1}$;

}

WHILE ($\varepsilon >$ *threshold*)

Ende mit Ergebnis $\mathbf{P}_0$

Abbildung 3.3: Struktur eines Powells Algorithmus

Die Optimierungsrichtungen werden mit den m Einheitsvektoren initialisiert, so daß $\mathbf{r}_i = \mathbf{e}_i$. Anschließend erfolgt nacheinander eine Optimierung entlang die-

ser Richtungen, wobei schrittweise die Punkte $\mathbf{P}_1, \mathbf{P}_2, \ldots, \mathbf{P}_m$ ermittelt werden durch:

$$\mathbf{P}_i := \arg\max_{\mathbf{x}} \{\ C(\mathbf{x}) \mid \mathbf{x} \in (\{\mathbf{P}_{i-1} + \lambda \mathbf{r}_i \mid \lambda \in \mathbb{R}\} \cap \mathcal{X})\ \}\ . \tag{3.34}$$

Anschließend erfolgt noch eine Optimierung mit Optimierungsrichtung $(\mathbf{P}_m - \mathbf{P}_0)$, so daß der Punkt $\mathbf{P}_{m+1}$ mit

$$\mathbf{P}_{m+1} := \arg\max_{\mathbf{x}} \{\ C(\mathbf{x}) \mid \mathbf{x} \in (\{\mathbf{P}_m + \lambda(\mathbf{P}_m - \mathbf{P}_0) \mid \lambda \in \mathbb{R}\ \} \cap \mathcal{X})\} \tag{3.35}$$

ermittelt wird. Unterschreitet die euklidische Norm zwischen $\mathbf{P}_0$ und $\mathbf{P}_{m+1}$ einen vorgegebenen Schwellwert, wird die Optimierung beendet, und das Ergebnis ist der Punkt $\mathbf{P}_{m+1}$. Ansonsten werden die neuen Optimierungsrichtungen als

$$\mathbf{r}_i = \mathbf{r}_{i+1} \quad i = 1, \ldots, m-1 \tag{3.36}$$

$$\mathbf{r}_m = \mathbf{P}_{m+1} - \mathbf{P}_0 \tag{3.37}$$

gesetzt. Der Punkt $\mathbf{P}_{m+1}$ wird zum Startpunkt für einen neuen Optimierungsdurchlauf.

Implementierung

Für eine Implementierung wird ein Verfahren benötigt, das das Maximum der Funktion $C : \mathcal{X} \to \mathbb{R}$ auf einem Teil ihres Definitionsbereiches bestimmt, einer Strecke zwischen zwei Punkten $\mathbf{P}_a$ und $\mathbf{P}_b$. Die verwendete Implementierung [17] benutzt dafür eine erweiterte Version des Brents Algorithmus [29].
Auf der Strecke $\mathbf{P}_a\mathbf{P}_b$ werden N äquidistante Stützstellen $\mathbf{s}_1, \ldots, \mathbf{s}_N$ mit

$$\mathbf{s}_i = \mathbf{P}_a + (i-1) \cdot \mathbf{r} \quad \text{mit} \quad \mathbf{r} = \frac{\mathbf{P}_b - \mathbf{P}_a}{N-1}$$

gesetzt, an denen die Funktion ausgewertet wird. Anschließend wird ein $i \in \{2, \ldots, (N-1)\}$ bestimmt, für das $C(\mathbf{s}_i) \geq C(\mathbf{s}_j)$ für alle $j = 1, \ldots, N$ gilt. Die Stützstellen $\mathbf{s}_{i-1}, \mathbf{s}_i, \mathbf{s}_{i+1}$ sind die Startwerte für die weitere Optimierung.
Bei dieser wird eine Interpolation mit einem Polynom zweiten Grades und den Stützstellen

$$\begin{aligned} \mathbf{s}_{i-1} &= \mathbf{P}_a + t_1 \cdot \mathbf{r} \\ \mathbf{s}_i &= \mathbf{P}_a + t_2 \cdot \mathbf{r} \\ \mathbf{s}_{i+1} &= \mathbf{P}_a + t_3 \cdot \mathbf{r} \end{aligned}$$

mit entsprechenden $t_1, t_2, t_3 \in \mathbb{R}$ durchgeführt. Man erhält eine Funktion $\mathrm{p} : \mathbb{R} \to \mathbb{R}$ mit $\mathrm{p}(t_1) = C(\mathbf{s}_{i-1})$, $\mathrm{p}(t_2) = C(\mathbf{s}_i)$ und $\mathrm{p}(t_3) = C(\mathbf{s}_{i+1})$. Das Maximum der Funktion wird für ein $t = \arg\max_t \mathrm{p}(t)$ angenommen. Ist

$$|C(\mathbf{s}_i) - C(\mathbf{P}_a + t \cdot \mathbf{r})| < \varepsilon$$

mit festem ε, dann ist die Optimierung beendet, und das Ergebnis ist $\mathbf{P}_a + t \cdot \mathbf{r}$. Sonst wird $\mathbf{s}_{i-1}$ durch $\mathbf{s}_i$ ersetzt, wenn $t < t_2$, oder andernfalls $\mathbf{s}_{i+1}$ durch $\mathbf{s}_i$. $\mathbf{s}_i$ wird danach durch $\mathbf{P}_a + t \cdot \mathbf{r}$ ersetzt, und es wird mit einer weiteren Interpolation fortgefahren.

3.4.4 Bewertung der Verfahren

Die Bewertung der Verfahren erfolgt anhand von fünf Testpaaren von Bildern. Dazu wurden die Abbildungsparameter zufällig festgelegt und das zu registrierende Bild aus einem T_1-gewichteten Kernspinbild durch Anwendung der Umkehrabbildung mit den gegebenen Parametern erzeugt. Die Parameter für die Testpaare sind in Tabelle 3.2 aufgeführt.

	Translation in Voxeln			Rotationswinkel in Grad			Skalierung		
Nr.	t_x	t_y	t_z	ϕ_x	ϕ_y	ϕ_z	s_x	s_y	s_z
1	1,59	-17,00	-37,76	-5,62	1,90	-6,02	1,01	0,99	1,02
2	3,04	0,10	24,14	6,23	-18,52	3,96	0,96	0,95	0,99
3	-7,14	-4,13	-18,04	-14,20	4,38	7,71	1,03	1,00	1,02
4	21,94	-7,28	10,30	9,34	-1,29	-10,90	0,99	0,97	1,03
5	-17,83	19,13	28,45	1,20	-7,30	12,45	1,04	1,02	0,97

Tabelle 3.2: Abbildungsparameter für die Testpaare

Für jedes Testpaar sollten die Optimierungsverfahren die Abbildungsparameter bestimmen. Es wurde gemessen, wie groß die Abweichung zwischen der gefundenen und der realen Abbildung ist und wieviel Zeit (in Minuten) für die Optimierung benötigt wurde. Die Testreihen wurden mit verschiedenen Parametern für die Optimierungsverfahren durchgeführt.
In den nächsten Abschnitten erfolgt eine Bewertung der drei Verfahren anhand der Genauigkeit und der benötigten Zeit[3] für die Registrierung. Zur Bewertung der Genauigkeit wird ein Punkt mit festen Koordinaten betrachtet. Einmal wird sein Bildpunkt bezüglich der Abbildungsparameter aus Tabelle 3.2 ermittelt und einmal der Bildpunkt bezüglich der während der Registrierung ermittelten Parameter. Der Abstand zwischen beiden Punkten wird gemessen und ergibt die Abweichung. Sie wird in Voxeln angegeben. Je kleiner die Abweichung ist, umso genauer erfolgt die Registrierung. Die Bewertung erfolgt mit dem Punkt $(100, 100, 100)^T$. Dieser liegt bei den verwendeten Bildern außerhalb des Kopfes. Damit wird eine obere Schranke für die Abweichung jedes Voxels angegeben.

Downhill Simplex Algorithmus

Zuerst erfolgt eine Bewertung des Downhill-Simplex-Algorithmus. Die zu variierenden Parameter sind die Startgitterweite d_{start} und die Stoppgitterweite d_{stop}. Allerdings beeinflußt lediglich d_{stop} die Genauigkeit. Als Startpunkt wird stets der Punkt $P = (0, 0, 0, 0, 0, 0, 1, 1, 1)^T \in \mathbb{R}^9$ gewählt. Das entspricht den Parametern für eine identische Abbildung. Eine Wahl des Startpunktes in Nähe des Optimierungsergebnisses beschleunigt zwar den Algorithmus. Allerdings ist in realen Anwendungen das Ergebnis nicht bekannt, so daß dort die identische Abbildung als Ausgang gerechtfertigt ist. Die Skalierungsfaktoren λ_i sind im folgenden für die Translationsparameter auf 5 , für die Rotationsparameter auf 2,5 und für die Skalierungsparameter auf 0,03 gesetzt.

[3] Diese und alle folgenden Zeitmessungen in dieser Arbeit wurden auf einer Origin 2000 von Silicon Graphics (Mountain View, CA) durchgeführt.

Für den Test wurde stellvertretend für Datensatz 1 aus Tabelle 3.2 die Abhängigkeit zwischen d_{stop}, Genauigkeit und Geschwindigkeit untersucht. Die Ergebnisse sind in der Tabelle 3.3 aufgeführt.

d_{stop}	1	0,5	0,2	0,1	0,05	0,02	0,01
t_x	5,00	2,50	2,50	2,50	1,88	1,56	1,64
t_y	-15,00	-17,50	-17,50	-16,88	-16,88	-17,03	-17,03
t_z	-35,00	-37,50	-36,25	-36,25	-36,56	-37,50	-37,66
ϕ_x	-5,00	-5,00	-5,62	-5,62	-5,78	-5,70	-5,66
ϕ_y	0,00	0,00	0,62	0,94	1,41	1,88	1,88
ϕ_z	-5,00	-5,00	-5,62	-5,94	-5,94	-6,02	-6,02
s_x	1,00	1,01	1,01	1,01	1,01	1,01	1,01
s_y	0,97	1,00	0,99	0,99	0,99	0,99	0,99
s_z	0,94	0,95	0,98	0,98	0,99	1,01	1,02
Abweichung	2,76	4,61	1,85	1,11	1,42	0,86	0,07
Zeit	71	117	171	200	264	503	601

Tabelle 3.3: Auswirkung einer Variierung der Stoppgitterweite auf das Downhill-Simplex-Verfahren

Eine Auswertung der Registrierung mit dem Downhill-Simplex-Algorithmus zeigt, daß sich die Genauigkeit der Registrierung mit zwei Ausnahmen ($d_{stop} = 0,5$ und $d_{stop} = 0,05$) durch eine kleinere Stoppgitterweite erhöht. Allerdings erhöht sich damit gleichzeitig die benötigte Zeit. Bei der Auswertung der beiden anderen Algorithmen wird sich jedoch zeigen, daß der Downhill-Simplex-Algorithmus in kürzester Zeit die besten Ergebnisse liefert.

Genetische Optimierung

Bei der genetischen Optimierung können vier Parameter variiert werden: die Anzahl der Individuen einer Population N_{IP}, die maximale Anzahl von Populationen N_P, die Anzahl N_{IN} der Individuen, die von einer in die nächste Population übernommen werden, und die Mutationsrate r_m. Der Gedanke bei diesem Verfahren ist, möglichst viele Individuen zu erzeugen und dadurch mit großer Wahrscheinlichkeit mindestens eines zu haben, das nahe dem Optimum ist. Im folgenden wird stellvertretend die Wirkung des Parameters N_P auf den Algorithmus untersucht. In der Tabelle 3.4 sind die Auswirkungen einer Änderung der maximalen Anzahl von Populationen auf die genetische Optimierung dargestellt.
Die Werte zeigen, daß die Optimierung sehr zeitaufwendig ist und langsam konvergiert. Nachteilig ist zudem, daß das Ergebnis zufällig ist, gesteuert durch die Mutationen, und sich nicht reproduzieren läßt. Um mit großer Sicherheit ein gutes Ergebnis (Abweichung $\leq 0,5$) zu erreichen, sind offenbar weit mehr als 100 Generationen erforderlich, was bei den verwendeten Bildern sehr lange Berechnungszeiten erfordern würde. Da eine genetische Optimierung bei genügend großer Anzahl von Generationen aber nahezu den gesamten Parameterbereich durchsucht, kann dieser Algorithmus unter Umständen dazu verwendet werden, das Optimierungsergebnis näherungsweise zu bestimmen. Dieses könnten

N_P	50	75	100
t_x	7,2	-4,3	5,2
t_y	-24,60	-19,2	-22,9
t_z	-8,4	-25,9	-26,4
ϕ_x	-15,00	-8,73	-12,44
ϕ_y	7,42	5,96	-2,63
ϕ_z	-10,56	-6,02	-1,71
s_x	0,98	1,00	0,99
s_y	1,00	1,00	0,97
s_z	0,98	1,00	0,95
Abweichung	6,23	4,83	15,39
Zeit	1043	1477	1814

Tabelle 3.4: Auswirkung einer Änderung der maximalen Anzahl von Populationen auf die genetische Optimierung

dann andere Algorithmen als Startpunkt zur Nachoptimierung verwenden. Die näherungsweise Bestimmung müßte allerdings auf Bildern mit sehr geringer Auflösung erfolgen, bei denen die Auswertung der Kostenfunktion auch erheblich weniger Zeit beansprucht.

Powells Algorithmus

Die Optimierung mittels Powells Algorithmus läßt sich über die Anzahl der Stützstellen im Brents Algorithmus steuern. Nach [22] sollen die initialen Optimierungsrichtungen so gesetzt werden, daß die Parameter in der Reihenfolge t_x, t_y, ϕ_z, t_z, ϕ_x, ϕ_y, s_x, s_y, s_z optimiert werden. Anhand von Testdatensatz 1 aus Tabelle 3.2 werden die ermittelten Abbildungsparameter, die Abweichung und die benötigte Zeit in Abhängigkeit von der Anzahl der Stützstellen N ermittelt (Ergebnisse siehe Abbildung 3.5). Wie beim Downhill-Simplex-Verfahren wird als Startpunkt stets $(0,0,0,0,0,0,1,1,1)^T \in \mathbb{R}^9$ genommen, der den Parametern für die identische Abbildung entspricht.
Aus den Ergebnissen sind zwei Dinge zu erkennen:

1. Die Genauigkeit der Registrierung wird von einer Erhöhung der Stützstellen offenbar nicht beeinflußt.

2. Die benötigte Zeit steigt mit der Erhöhung der Stützstellen an.

Im Vergleich zum Downhill-Simplex-Algorithmus fällt auf, daß der Powells-Algorithmus bei gleicher Zeit ein schlechteres Ergebnis liefert.

3.5 Implementierung

Die Bewertung der Optimierungsverfahren hat deren Vor- und Nachteile deutlich gezeigt. Für den Downhill-Simplex-Algorithmus hat sich bei der Bewertung der drei Verfahren gezeigt, daß er die besten Ergebnisse bei kürzester Zeit liefert.

N	8	16	32	64	128
t_x	3,67	3,31	3,78	3,85	3,83
t_y	-16,70	-16,66	-16,54	-16,41	-16,53
t_z	-37,59	-37,75	-37,71	-37,73	-37,70
ϕ_x	-5,37	-5,35	-5,26	-5,20	-5,25
ϕ_y	0,79	1,00	0,71	0,67	0,68
ϕ_z	-5,80	-5,83	-5,78	-5,80	-5,77
s_x	1,01	1,01	1,01	1,01	1,01
s_y	0,99	0,99	0,99	0,99	0,99
s_z	1,02	1,02	1,02	1,02	1,02
Abweichung	2,69	2,18	2,92	3,08	3,01
Zeit	197	175	240	393	707

Tabelle 3.5: Auswirkung einer Änderung der Anzahl an Stützstellen N auf den Powells Algorithmus. Es wurde der Datensatz 1 aus Tabelle 3.2 verwendet

Die für die Optimierung benötigte Zeit setzt sich aus zwei Teilen für die näherungsweise Bestimmung des Optimums und die Nachoptimierung zusammen. Eine Verkürzung der benötigten Zeit läßt sich dadurch erreichen, daß man zur näherungsweisen Bestimmung des Optimums Bilder mit einer geringeren Auflösung verwendet, wodurch die Auswertung der Kostenfunktion weniger Zeit beansprucht[4]. Das so bestimmte Ergebnis wird dann als Startwert für die Optimierung mit voller Auflösung der Bilder verwendet.

Deshalb wird die Registrierung als Mehrgitterverfahren aufgebaut. Von dem zu registrierenden Bild und dem Referenzbild wird eine Gaußpyramide mit einer vorgegebenen Anzahl von Ebenen erstellt. Ausgehend von der höchsten Ebene wird das auf einer Ebene gefundene Registrierungsergebnis für die Optimierung auf der nächsttieferen Ebene als Startpunkt verwendet. Das endgültige Ergebnis liefert die Optimierung auf der untersten Ebene, die auf den Bildern mit voller Auflösung durchgeführt wird.

Zuerst wurde für die Optimierung auf der obersten Ebene eine genetische Optimierung verwendet. Dadurch sollte erreicht werden, daß eine Näherungslösung in der Nähe des Optimums als Startpunkt für die weitere Optimierung bestimmt wird. Das Maximum sollte dann in einer kleinen Umgebung um den Startpunkt mittels Powells-Algorithmus gesucht werden. Die Eigenschaft der genetischen Optimierung, teilweise Ergebnisse zu liefern, die sehr weit vom globalen Maximum entfernt sind, führt dazu, daß diese Umgebung entsprechend größer gewählt werden muß. Dadurch benötigt die Registrierung oft mehr Zeit als mit dem Powells-Algorithmus allein. Aus diesem Grund ist die verwendete Optimierung nur auf Basis des Downhill-Simplex-Algorithmus implementiert.

[4] Der Aufwand für die Auswertung der Kostenfunktion setzt sich aus zwei Teilen zusammen: 1. der Anwendung der Abbildung und der Aufbau des Scatterplots und 2. dessen Auswertung. Der Aufwand für den ersten Teil wächst linear mit der Anzahl der Voxel, der für den zweiten Teil ist konstant und kann vernachlässigt werden.

3.5.1 Mehrgitterrepräsentation

Der erste Schritt ist der Aufbau einer Gaußpyramide mit einer gegebenen Höhe h. Ausgehend von den Bildern b_{PD} und b_{T_1} werden zwei Folgen $\left(b_{PD}^{(i)}\right)_{i=0,\ldots,h}$ und $\left(b_{T_1}^{(i)}\right)_{i=0,\ldots,h}$ mit $b_{PD}^{(0)} = b_{PD}$, $b_{T_1}^{(0)} = b_{T_1}$, $b_{PD}^{(i+1)} = \mathcal{D}b_{PD}^{(i)}$ und $b_{T_1}^{(i+1)} = \mathcal{D}b_{T_1}^{(i)}$ erzeugt.
Der Operator $\mathcal{D}$ halbiert die Auflösung eines Bildes und reduziert dadurch auch die Bildgröße entsprechend. Sei $b = (\mathcal{T}, \mathcal{G}, I)$ ein Bild mit Trägermenge $\mathcal{T} = \{0, \ldots, n_c - 1\} \times \{0, \ldots, n_r - 1\} \times \{0, \ldots, n_s - 1\}$. Ohne Beschränkung der Allgemeinheit seien n_c, n_r und n_s nicht durch zwei teilbar. Dann ergibt sich das Bild $b_2 = Db$ wie folgt:

1. Filterung des Bildes b mit einem Gaußfilter mit $\sigma = 1$. Das Ergebnis ist das Bild $b_G = \text{gauss}(b, \sigma) = (\mathcal{T}, \mathcal{G}, I_g)$,

2. Konstruktion des Bildes $b_2 = (\mathcal{T}_2, \mathcal{G}, I_2)$ mit

$$\mathcal{T}_2 = \{0, \ldots, \frac{n_c - 1}{2}\} \times \{0, \ldots, \frac{n_r - 1}{2}\} \times \{0, \ldots, \frac{n_s - 1}{2}\}$$

   und $I_2((x, y, z)^T) = I_g\left((2x, 2y, 2z)^T\right)$ für alle $(x, y, z)^T \in \mathcal{T}_2$.

3.5.2 Optimierung mit Downhill-Simplex-Algorithmus

Entsprechend dem beschriebenen Vorgehen erfolgt die Optimierung der Kostenfunktion. Die einzige Steuerung ist die Veränderung der Höhe der Gaußpyramide. Das führt zu dem Registrieralgorithmus, der in Abbildung 3.4 dargestellt ist (Eingabeparameter sind das zu registrierende Bild b_{PD}, das Referenzbild b_{T_1} und die Höhe h der Gaußpyramide).

3.6 Bewertung der Implementierung

Die Bewertung der Implementierung erfolgt anhand der fünf Testdatensätze aus Tabelle 3.2. Es werden sowohl die gefundenen Translationsparameter, als auch die daraus errechnete Abweichung und die benötigte Zeit angegeben.

3.6.1 Ergebnis der Registrierung

Die Implementierung wurde mit verschiedenen Höhen der Gaußpyramide getestet. Die Ergebnisse der Registrierung sind in den Tabellen 3.6, 3.7 und 3.8 aufgeführt. Die daraus errechnete Abweichung (in Voxeln) und die für die Registrierung benötigte Zeit (in Minuten) befinden sich in Tabelle 3.9.
Die Bilder haben eine Auflösung von einem Voxel, so daß die Meßgenauigkeit ein halbes Voxel beträgt. Die gemessenen Abweichungen lagen unabhängig von der Höhe der Gaußpyramide stets unter einem halben Voxel, so daß sich in Bezug auf die Genauigkeit keine Unterschiede zwischen den verschiedenen Höhen erkennen lassen. Die benötigte Zeit war am höchsten, wenn die Auflösungspyramide eine Höhe von $h = 1$ hatte. Bei der Höhe $h = 2$ ist sie mit einer Ausnahme

Algorithmus zum Registrieren eines *PD*-gewichteten Kernspinbildes auf einem T_1-gewichteten

Start mit Punkt $\mathbf{P}_0 = (0,0,0,0,0,0,1,1,1)^T \in \mathbb{R}^9$

Berechnen der Gaußpyramiden $\left(b_{PD}^{(i)}\right)_{i=0,\ldots,h}$, $\left(b_{T_1}^{(i)}\right)_{i=0,\ldots,h}$

$j = h$;

WHILE $(j \geq 0)$ DO

{

Optimierung der Kostenfunktion $C_{b_{T_1}^{(j)}, b_{PD}^{(j)}}(\mathbf{x})$ *mit dem Downhill-Simplex-Verfahren mit Startpunkt* $\mathbf{P}_0$, $d_{start} = 1$, $d_{stop} = 0,01$. *Das Ergebnis der Optimierung sei* $\mathbf{P}$

Kopieren von $\mathbf{P}$ *in* $\mathbf{P}_0$, *dabei Verdopplung aller Translationsparameter*

$j = j - 1$;

}

Ende mit Ergebnis $\mathbf{P}$

Abbildung 3.4: Aufbau des implementierten Registrieralgorithmus

(Testdatensatz 5) am geringsten, so daß sich ein Optimum bei dieser Höhe ablesen läßt. Für die Implementierung wird deshalb $h = 2$ verwendet.

3.6.2 Einfluß von Filtern auf das Ergebnis

Abschließend wird noch untersucht, ob und wie sich eine Filterung der Bilder auf das Ergebnis der Registrierung auswirkt. Dazu wurden vor Beginn der Registrierung sowohl das zu registrierende Bild als auch das Referenzbild mit demselben Filter und denselben Filterparametern gefiltert.
Die verwendeten Filter sind ein Gaußfilter und ein Filter auf Basis von anisotroper Diffusion mit jeweils drei verschiedenen Parametereinstellungen sowie der Lee-Filter. Die Tests wurden an dem Testdatensatz Nummer eins aus Tabelle 3.2 durchgeführt. Die Ergebnisse sind in Tabelle 3.10 aufgeführt.
Vergleicht man die ermittelten Abweichungen mit der Abweichung, die sich bei der Registrierung ohne vorherige Filterung ergibt, so sind keine nennenswerten Unterschiede erkennbar. Die Abweichung ohne Filterung beträgt bei dem verwendeten Datensatz und der verwendeten Höhe der Gaußpyramide 0,30 (siehe Tabelle 3.9). Offenbar lassen sich durch Anwendung des Gaußfilters, des Leefilters oder der anisotropen Diffusion mit einer kleinen Zeit t noch Verbesserungen erzielen. Allerdings liegen diese unterhalb einer sinnvoll anzusetzenden Genau-

	Translation in Voxel			Rotationswinkel in Grad			Skalierung		
Nr.	t_x	t_y	t_z	ϕ_x	ϕ_y	ϕ_z	s_x	s_y	s_z
1	1,6	-17,0	-37,8	-5,57	1,81	-6,02	1,01	0,99	1,02
2	3,1	0,1	24,2	6,18	-18,53	3,98	0,96	0,95	0,99
3	-7,2	-4,2	-18,0	-14,18	4,38	7,68	1,03	1,00	1,02
4	21,9	-7,3	10,3	9,30	-1,32	-10,88	0,99	0,97	1,03
5	-17,8	19,2	28,3	1,22	-7,25	12,35	1,04	1,02	0,97

Tabelle 3.6: Registrierungsergebnisse für die Höhe $h = 1$ der Gaußpyramide

	Translation in Voxel			Rotationswinkel in Grad			Skalierung		
Nr.	t_x	t_y	t_z	ϕ_x	ϕ_y	ϕ_z	s_x	s_y	s_z
1	1,6	-17,0	-37,7	-5,59	1,84	-5,96	1,01	0,99	1,02
2	3,1	0,2	24,2	6,27	-18,61	3,96	0,96	0,95	0,99
3	-7,2	-4,2	-18,0	-14,18	4,42	7,73	1,03	1,00	1,02
4	21,9	-7,3	10,3	9,28	-1,31	-10,91	0,99	0,97	1,03
5	-17,8	19,1	28,4	1,15	-7,35	12,44	1,04	1,02	0,97

Tabelle 3.7: Registrierungsergebnisse für die Höhe $h = 2$ der Gaußpyramide

igkeit von einem halben Voxel. Diese Ergebnisse sind allerdings nicht durch größere Testreihen gesichert.

3.7 Registrierung der verwendeten Kernspinbilder

In diesem Abschnitt werden die im folgenden verwendeten Paare aus jeweils einem T_1- und einem PD-gewichteten Kernspinbild registriert. Für jeden der fünf Datensätze ist jeweils ein axialer Schnitt durch das originale T_1- und PD-gewichtete Bild dargestellt und ein axialer Schnitt durch das registrierte PD-gewichtete Bild (Abbildung 3.5). Alle axialen Schnitte für einen Datensatz erfolgen in derselben Schicht.
Um die Registrierung zu beschleunigen, wurde zusätzlich der vorhandene Parallelrechner genutzt. Die Berechnung der Abbildung und die Erstellung des Scatterplots sind auf der Origin unter Ausnutzung von acht Prozessoren parallelisiert worden. In Tabelle 3.11 befinden sich die ermittelten Abbildungsparameter und die Zeiten für die Registrierung.

	Translation in Voxel			Rotationswinkel in Grad			Skalierung		
Nr.	t_x	t_y	t_z	ϕ_x	ϕ_y	ϕ_z	s_x	s_y	s_z
1	1,6	-17,0	-37,8	-5,58	1,81	-5,98	1,01	0,99	1,02
2	3,0	0,2	24,2	6,25	-18,48	3,95	0,96	0,95	0,99
3	-7,2	-4,0	-18,2	-14,12	4,42	7,73	1,03	1,00	1,02
4	21,9	-7,3	10,3	9,30	-1,31	-10,88	0,99	0,97	1,03
5	-17,8	19,1	28,3	1,21	-7,36	12,46	1,04	1,02	0,97

Tabelle 3.8: Registrierungsergebnisse für die Höhe $h = 3$ der Gaußpyramide

	Abweichung in Voxeln			Zeit in Minuten		
Nr.	$h = 1$	$h = 2$	$h = 3$	$h = 1$	$h = 2$	$h = 3$
1	0,26	0,30	0,28	220	146	178
2	0,12	0,26	0,10	155	116	147
3	0,17	0,08	0,10	239	200	211
4	0,15	0,11	0,14	224	184	184
5	0,39	0,08	0,13	168	122	98

Tabelle 3.9: Auswertung und Vergleich der ermittelten Abbildungsparameter in Abhängigkeit der Höhe der Gaußpyramide

	Gaußfilter			Leef.	Anisotr. Diff.		
	$\sigma = 0,5$	$\sigma = 1$	$\sigma = 1,5$		$t = 10$	$t = 20$	$t = 30$
t_x	1,6	1,7	1,7	1,6	1,6	1,6	1,6
t_y	-17,0	-17,0	-17,0	-17,0	-16,9	-16,8	-16,8
t_z	-37,6	-37,7	-37,7	-37,7	-37,6	-37,6	-37,6
ϕ_x	-5,63	-5,61	-5,59	-5,65	-5,57	-5,62	-5,56
ϕ_y	1,88	1,80	1,84	1,88	1,91	1,91	1,84
ϕ_z	-6,02	-5,98	-5,98	-6,02	-6,00	-6,01	-5,98
s_x	1,01	1,01	1,01	1,01	1,01	1,01	1,01
s_y	0,99	0,99	0,99	0,99	0,99	0,99	0,99
s_z	1,02	1,02	1,02	1,02	1,02	1,02	1,02
Abweichung	0,18	0,29	0,24	0,06	0,24	0,26	0,46

Tabelle 3.10: Registrierungsergebnisse bei einer Filterung der Bilder vor der Registrierung mit einem Gaußfilter, einem Leefilter und bei Filterung mit anisotroper Diffusion

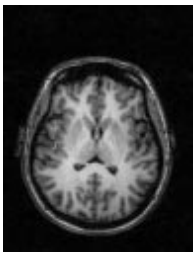
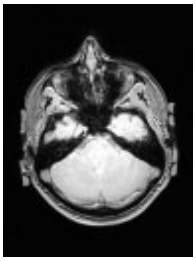
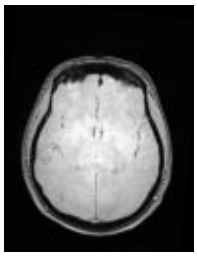
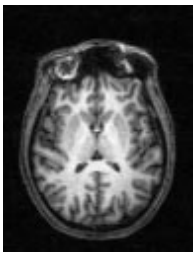
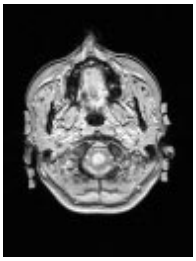
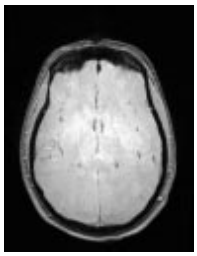
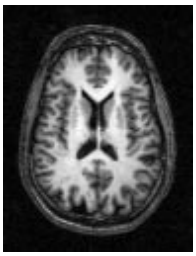
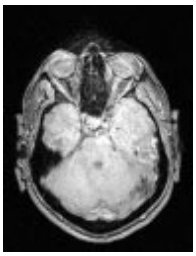
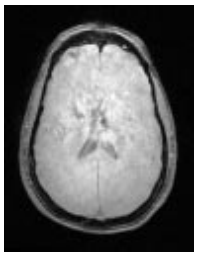
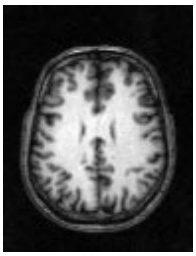
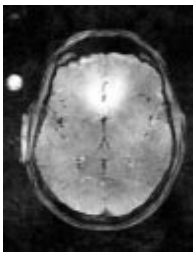
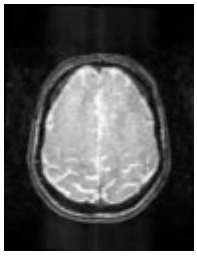
Bild	T_1-gew.	PD-gew.	reg. PD-gew.
1			
2			
3			
4			
5			

Abbildung 3.5: Registrierung der in dieser Arbeit verwendeten PD-gewichteten Kernspinbilder auf den zugehörigen T_1-gewichteten, axiale Schnittdarstellung

Bild	1	2	3	4	5
t_x	0,2	0,2	1,6	6,9	-1,6
t_y	-8,4	-5,6	6,6	-6,3	-1,8
t_z	-43,9	-75,4	-14,3	-22,0	-12,2
ϕ_x	-5,32	-7,20	-11,90	-0,15	0,40
ϕ_y	-1,76	0,18	-4,82	-0,28	-0,77
ϕ_z	-6,14	-3,86	-4,34	1,19	-1,41
s_x	1,01	0,97	1,01	1,00	1,00
s_y	0,99	0,87	0,98	0,99	1,01
s_z	1,02	0,98	1,04	1,02	1,00
Zeit					
sequentiell	222	190	324	139	155
parallel	41	39	60	27	36
Speedup	5,4	4,9	5,4	5,1	4,3

Tabelle 3.11: Abbildungsparameter und benötigte Zeit (sequentiell und auf acht Prozessoren parallelisiert) in Minuten für die Registrierung der Kernspinbilder

Kapitel 4

Segmentierung

Dieses Kapitel beschäftigt sich mit der Segmentierung des Knochens aus einem T_1- und einem PD-gewichteten Kernspinbild. Die Voraussetzung dafür ist, daß das PD-gewichtete Bild auf dem T_1-gewichteten registriert ist. Das Ergebnis der Segmentierung wird eine Knochenmaske sein. Diese Maske ist ein Bild $(\mathcal{T}, \{0,1\}, I_{maske})$ mit der gleichen Trägermenge $\mathcal{T}$ und binärer Grauwertmenge. Liegt ein Voxel $\mathbf{x}$ innerhalb des Knochens, ist $I_{maske}(\mathbf{x}) = 1$, andernfalls ist $I_{maske}(\mathbf{x}) = 0$.
Durch die Knochenmaske wird die Kante zwischen dem Knochen und der Gehirnflüssigkeit (innere Kante) und die Kante zwischen dem Knochen und der Kopfhaut (äußere Kante) repräsentiert. Zwischen diesen Kanten befindet sich der Knochen. Die äußere Kante ist sowohl im T_1- als auch im PD-gewichteten Bild vorhanden, die innere Kante nur im PD-gewichteten Bild.
Im folgenden werden verschiedene Ansätze für eine Segmentierung vorgestellt und daraus ein Algorithmus dafür entwickelt.

4.1 Der Isodata-Algorithmus

Schwellwertbasierte Segmentierungsansätze gehören zu den einfachsten Verfahren. Alle Voxel in einem Bild $b \in \mathcal{B}$, deren Grauwerte innerhalb eines vorgegebenen Bereichs $[g_{min}, g_{max}]$ liegen, werden als Vordergrund klassifiziert:

$$b_{seg} = \text{binarize}(b, g_{min}, g_{max}) \; . \tag{4.1}$$

Diese Verfahren haben allerdings den Nachteil, daß für die Wahl von g_{min} und g_{max} manuell eingegriffen werden muß, um diese Werte an das zu segmentierende Bild anzupassen. Der Wunsch, diese Werte automatisch festzulegen, hat zur Entwicklung von sogenannten unüberwachten Klassifizierungsalgorithmen geführt. Ein Vertreter dieser Algorithmen ist der Isodata-Algorithmus (siehe [21]).
Der Isodata-Algorithmus unterteilt die Voxel eines Bildes $b \in \mathcal{B}_\mathcal{T}$ anhand ihrer Grauwerte in k Klassen $C_1, \ldots, C_k \subseteq \mathcal{T}$, so daß jedes Voxel zu genau einer dieser Klassen gehört. Jede Klasse $C_j, j = 1, \ldots, k$ besitzt ein Klassenzentrum v_j mit

$$v_j = \frac{1}{|C_j|} \sum_{\mathbf{x} \in C_j} I(\mathbf{x}) \ , \tag{4.2}$$

das gleich dem Mittelwert der Grauwerte aller Voxel in dieser Klasse ist. Die Zuordnung der Voxel zu den Klassen erfolgt so, daß

$$J_{isodata} = \sum_{j=1}^{k} \sum_{\mathbf{x} \in C_j} |I(\mathbf{x}) - v_j| \tag{4.3}$$

minimal ist.
Der Isodata-Algorithmus übernimmt diese Zuordnung durch ein iteratives Vorgehen (siehe Abbildung 4.1).

Isodata-Algorithmus zur Segmentierung eines Bildes in k Klassen

Initialisierung $v_1^{(0)}, \ldots, v_k^{(0)}$

$l = 0$;

DO

{

$l = l + 1$;

FOR ALL $\mathbf{x} \in \mathcal{T}$ DO

Zuordnung von $\mathbf{x}$ *zur Klasse* C_i, $i \in \{1, \ldots, k\}$, *so daß für alle* $j \in \{1, \ldots, k\}$ *gilt:* $|I(\mathbf{x}) - v_i^{(l-1)}| \leq |I(\mathbf{x}) - v_j^{(l-1)}|$

FOR ALL $i \in \{1, \ldots, k\}$ DO

$v_i^{(l)} = \frac{1}{|C_i|} \sum_{\mathbf{x} \in C_i} I(\mathbf{x})$

}

WHILE $(\sum_{i=1}^{k} |v_i^{(l)} - v_i^{(l-1)}| \geq \varepsilon)$

Abbildung 4.1: Der Isodata-Algorithmus

Die Anzahl an Klassen k wird vom Anwender festgelegt. Zunächst erfolgt eine Initialisierung der Klassenzentren $v_1^{(0)}, \ldots, v_k^{(0)}$. Im Iterationsschritt l wird jedes Voxel der Klasse C_i zugeordnet, für die

$$|I(\mathbf{x}) - v_i^{(l-1)}| \leq |I(\mathbf{x}) - v_j^{(l-1)}| \quad j = 1, \ldots, k \tag{4.4}$$

gilt. Als nächstes werden die Klassenzentren $v_1^{(l)}, \ldots, v_k^{(l)}$ nach der Zuweisung neu berechnet. Ist

$$\sum_{i=1}^{k} |v_i^{(l)} - v_i^{(l-1)}| < \varepsilon \ , \tag{4.5}$$

wird der Algorithmus beendet mit den Klassen $C_1, \ldots, C_k$, ansonsten wird ein neuer Iterationsschritt durchgeführt. Das Ergebnis ist ein Bild $b_{seg} = (\mathcal{T}, \mathbb{N}_0, I_{seg})$ mit

$$I_{seg}(\mathbf{x}) = i - 1 \quad \text{falls} \quad \mathbf{x} \in C_i$$

bei dem für jedes Voxel $\mathbf{x}$ der Wert $I_{seg}(\mathbf{x})$ die Zuordnung des Voxels zu einer Klasse anzeigt. Die Werte sind aus dem Bereich $0, \ldots, (k-1)$. $I(\mathbf{x}) = i$ bedeutet $\mathbf{x} \in C_{i+1}$. Im folgenden wird dafür die Schreibweise

$$b_{seg} = \text{isodata}(b, k)$$

verwendet. Sie hat die Bedeutung, daß das Bild b_{seg} die Zuordnung der Voxel zu den Klassen enthält, die durch Anwendung des Isodata-Algorithmus mit k Klassen auf das Bild b gebildet worden sind.
Der Isodata-Algorithmus kann durch zwei Parameter gesteuert werden: die Anzahl an Klassen k und die Abbruchbedingung ε. Die Abbruchbedingung ε wird für den Algorithmus fest vorgegeben. Die Anzahl an Klassen ist nur von der Art der zu segmentierenden Bilder abhängig, so daß eine Segmentierung ohne manuellen Eingriff möglich ist.
Im weiteren ist $\varepsilon = 0,1$. Die Anzahl an Klassen wurde so gewählt, daß jeder Gewebetyp möglichst eine eigene Klasse besitzt und andererseits kein Gewebetyp über mehrere Klassen verteilt ist. Das führt zu drei Klassen im T_1-gewichteten und zu zwei Klassen im PD-gewichteten Bild (siehe Abbildung 4.2).

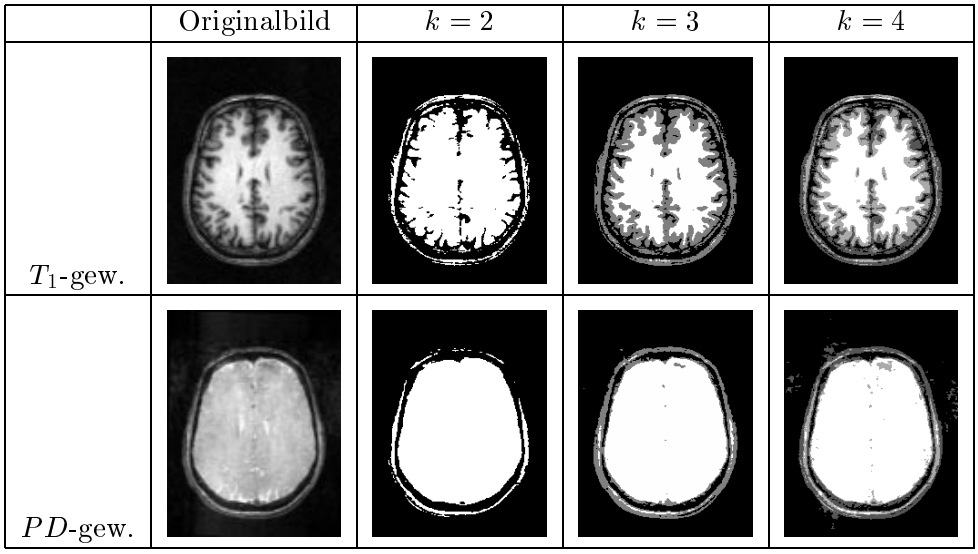


Abbildung 4.2: Segmentierung eines T_1- und eines PD-gewichteten Bildes mit Isodata und verschiedenen Anzahlen an Klassen

4.2 Segmentierung mit dem Isodata-Algorithmus

Der erste verwendete Ansatz zur Segmentierung des Knochens basiert auf Arbeiten von G. Palubinskas (siehe [26]). Sie erfolgt mit Hilfe von PD-gewichteten Kernspinbildern. Das Vorgehen gliedert sich in drei Schritte (Abbildung 4.3):

1. Erstellung der Kopfmaske,
2. Erstellung der Maske für die Hirnflüssigkeit und das Gehirn (im folgenden Liquormaske genannt),
3. Erstellung der Maske für den Knochen mit Hilfe der Kopf- und der Liquormaske.

Als erstes erfolgt stets eine Segmentierung des PD-gewichteten Kernspinbildes b_{PD} mit Hilfe des Isodata-Algorithmus in zwei Klassen. In einer Klasse sind der Knochen, Lufteinschlüsse und der Hintergrund enthalten, die andere Klasse umfaßt alle protonenreichen Gewebe (Kopfhaut, Gehirnflüssigkeit und Gehirn). Gehört ein Voxel zu der Klasse mit Knochen und Hintergrund, so ist sein Grauwert im Zwischenbild z_1 gleich 0, sonst gleich 1. Alle verwendeten Distanzen sind in Millimetern zu verstehen.
Der erste Schritt der Segmentierung, die Erstellung der Kopfmaske, umfaßt die folgenden Schritte (Abbildung 4.4):

1. Segmentierung des PD-gewichteten Bildes mit dem Isodata-Algorithmus in zwei Klassen. Die eine Klasse umfaßt den Knochen und den Hintergrund (Klasse 0), zu der anderen Klasse gehört die Kopfhaut, die Hirnflüssigkeit und das Gehirn (Klasse 1), Auswahl der Klasse 1,
2. Distanztransformation und Selektion aller Voxel mit einem Abstand von mindestens d_1 von Kopfhaut, Hirnflüssigkeit oder Gehirn,
3. Auswahl der größten zusammenhängenden Komponente,
4. Distanztransformation und wieder Selektion aller Voxel mit einem Abstand von mindestens d_1. Das Ergebnis ist die Kopfmaske.

Das Vorgehen baut auf der Tatsache auf, daß alle Voxel, die einen hohen Grauwert besitzen, zum Kopf gehören. Diese Voxel befinden sich nach dem ersten Schritt in der Klasse 1. Damit ist eine erste Kopfmaske erstellt. Diese enthält jedoch an Stellen, an denen sich beispielsweise der Knochen befindet, Löcher, die in den Schritten zwei bis vier geschlossen werden.
Im zweiten Schritt wird die Liquormaske erstellt. Alle Voxel im PD-gewichteten Bild, die innerhalb der Liquormaske liegen, besitzen hohe Grauwerte. Deshalb werden analog wie bei der Erstellung der Kopfmaske die Voxel mit dem Isodata-Algorithmus in zwei Klassen unterteilt und alle Voxel, die zur Klasse 1 gehören, ausgewählt. Das sind alle Voxel, die zur Hirnflüssigkeit, zum Gehirn oder zur Kopfhaut gehören. Innerhalb dieser initialen Maske sind die Bereiche für Hirnflüssigkeit und Gehirn mit dem Bereich für die Kopfhaut durch schmale Brücken verbunden. Werden diese entfernt, stellt der Bereich für Hirnflüssigkeit und Gehirn die größte zusammenhängende Komponente im Bild dar. Diese wird selektiert und der Fehler, der nach der Entfernung der Brücken auftritt, durch eine Überlagerung mit der initialen Maske korrigiert. Anschließend werden kleine Löcher in der Maske geschlossen.
Damit umfaßt der zweite Schritt, die Erstellung der Liquormaske, die folgenden Teile (Abbildung 4.5):

ERSTELLUNG DER KOPFMASKE
{

$$
\begin{aligned}
z_1 &= \text{binarize}(\text{isodata}(b_{PD}, 2), 1, 1); \\
z_2 &= \text{binarize}(\text{dist}(z_1), d_1, \infty); \\
z_3 &= \text{biggestcomp}(z_2); \\
b_{KOPF} &= \text{binarize}(\text{dist}(z_3), d_1, \infty);
\end{aligned}
$$

}

ERSTELLUNG DER MASKE FÜR HIRNFLÜSSIGKEIT UND GEHIRN
{

$$
\begin{aligned}
z_1 &= \text{binarize}(\text{isodata}(b_{PD}, 2), 1, 1); \\
z_2 &= \text{invert}(z_1); \\
z_3 &= \text{binarize}(\text{dist}(z_2), 5, \infty); \\
z_4 &= \text{biggestcomp}(z_3); \\
z_5 &= \text{binarize}(\text{dist}(z_4), 0, 5); \\
z_6 &= \text{and}(z_1, z_5); \\
z_7 &= \text{invert}(z_6); \\
z_8 &= \text{biggestcomp}(z_7); \\
b_{LIQUOR} &= \text{invert}(z_8);
\end{aligned}
$$

}

ERSTELLUNG DER MASKE FÜR DEN KNOCHEN
{

$$
\begin{aligned}
z_1 &= \text{binarize}(\text{isodata}(b_{PD}, 2), 0, 0); \\
z_2 &= \text{invert}(b_{KOPF}); \\
z_3 &= \text{binarize}(\text{dist}(z_2), d_2, \infty) \\
z_4 &= \text{and}(z_1, z_3); \\
z_5 &= \text{binarize}(\text{dist}(b_{LIQUOR}), 0, d_3) \\
b_{KNOCHEN} &= \text{and}(z_4, z_5);
\end{aligned}
$$

}

Abbildung 4.3: Algorithmus für eine Segmentierung des Knochens aus einem PD-gewichteten Bild auf Basis des Isodata-Algorithmus

Abbildung 4.4: Zwischenergebnisse bei der Erstellung der Kopfmaske, von links nach rechts: PD-gewichtetes Bild, Schritte 1 bis 4

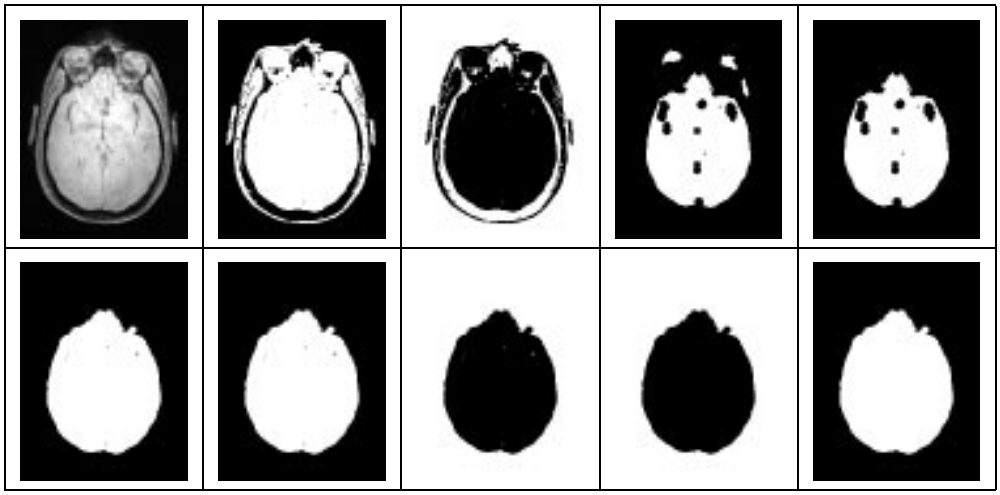


Abbildung 4.5: Zwischenergebnisse bei der Erstellung der Liquormaske, von links nach rechts: oben: PD-gewichtetes Bild und Schritte 1 bis 4, unten Schritte 5 bis 9

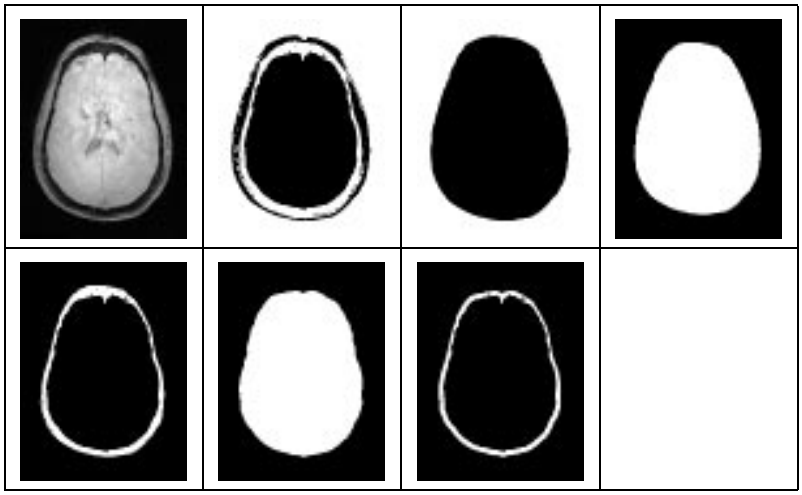


Abbildung 4.6: Zwischenergebnisse bei der Erstellung der Knochenmaske, von links nach rechts: oben: PD-gewichtetes Bild und Schritte 1 bis 3, unten Schritte 4 bis 6

1. Segmentierung des Bildes in zwei Klassen und Auswahl aller Voxel, die zur Klasse 1, d.h. der Klasse mit Kopfhaut, Hirnflüssigkeit und Gehirn, gehören,
2. Invertieren des entstandenen Bildes,
3. Distanztransformation und Selektion aller Voxel mit einem Abstand größer oder gleich 5mm,
4. Auswahl der größten zusammenhängenden Komponente,
5. Distanztransformation und Selektion aller Voxel mit einem Abstand bis zu 5mm,
6. Überlagerung mit dem im Schritt 1 entstandenen Bild,
7. Invertierung der im letzten Teilschritt entstandenen Maske,
8. Auswahl der größten zusammenhängenden Komponente,
9. erneute Invertierung.

Das Ergebnis ist die Liquormaske. Die Schritte 1 bis 6 liefern aus dem Bild die Maske. In Schritt 1 erfolgt eine Klassifizierung der Voxel. Die Schritte 2 bis 6 extrahieren daraus die Liquormaske. In den Schritten 7 bis 9 werden kleine Löcher in der Maske geschlossen.
Sind Kopf- und Liquormaske erstellt, wird im dritten Schritt die Maske für den Knochen mit Hilfe der beiden generiert. Zum Knochen gehören die Voxel, die einen niedrigen Grauwert besitzen, innerhalb des Kopfes und außerhalb der Gehirnflüssigkeit liegen. Deshalb müssen die Voxel zunächst mit dem Isodata-Algorithmus in zwei Klassen geteilt werden. Anschließend wird die Klasse 0 ausgewählt, die die Voxel mit den niedrigen Grauwerten umfaßt. Damit sind alle Voxel ausgewählt, die entweder zum Knochen oder zum Hintergrund gehören. Zum Knochen gehören die Voxel, die innerhalb der Kopfmaske und außerhalb der Liquormaske liegen. Um eine bessere Trennung vom Hintergund vorzunehmen, können nur Voxel zum Knochen gehören, die mindestens einen Abstand d_2 vom Rand der Kopfmaske und maximal einen Abstand d_3 vom Rand der Liquormaske besitzen. Für die Erstellung der Knochenmaske werden die folgenden Schritte ausgeführt, die auch in Abbildung 4.6 dargestellt sind:

1. Segmentierung des Bildes in zwei Klassen und Auswahl aller Voxel, die zur Klasse 0, d.h. zu der Klasse mit Hintergrund und Knochen gehören,
2. Invertierung der Kopfmaske,
3. Distanztransformation auf der invertierten Kopfmaske und Auswahl aller Voxel mit einem Abstand von mindestens d_2,
4. Überlagerung der Masken aus Schritt 1 und 3,
5. Distanztransformation auf der Liquormaske und Auswahl aller Voxel bis einem Abstand von kleiner oder gleich d_3,

6. Überlagerung des letzten Teilergebnisses mit der Maske aus Schritt 4.

Das Ergebnis ist eine Maske für den Knochen. Ihre Erstellung kann durch die Distanzparameter d_2 und d_3 gesteuert werden.
Die gesamte Segmentierung wird durch die drei Parameter d_1, d_2 und d_3 gesteuert. Wie in [26] werden $d_2 = 7$ und $d_3 = 6$ gesetzt. Als nächstes erfolgt eine Untersuchung der Auswirkung des Parameters d_1.
Dieser wirkt auf die Erzeugung der Kopfmaske. Das Ergebnis für verschiedene Werte von d_1 ist in der Abbildung 4.7 dargestellt. Dabei wurden die Werte $d_1 = 2$, $d_1 = 4$ und $d_1 = 6$ (Distanzen in Millimetern) verwendet und die entstandene Maske dargestellt (Abbildung 4.7). Aus der Abbildung ist ersichtlich, daß die Kopfmasken bei kleinen Werten von d_1 unvollständig sind. So sind bei $d_1 = 2$ alle und bei $d_1 = 4$ mit einer Ausnahme (Bild 3) auch alle Masken unvollständig. Erst bei $d_1 = 6$ lassen sich geschlossene Masken für alle Bilder erstellen. Für $d_1 = 6$ wurde der Rand der Maske in die originalen PD-gewichteten Kernspinbilder eingeblendet (Abbildung 4.9). Dort ist zu erkennen, daß auch bei $d_1 = 6$ die Kopfmasken fehlerbehaftet sind, insbesondere bei den Bildern 4 und 5, wo Teile des Kopfes, insbesondere des Knochens und der Hirnflüssigkeit, außerhalb der Kopfmaske liegen. Eine weitere Erhöhung von d_1 würde zwar dieses Problem beheben. Allerdings würden dadurch an anderen Stellen größere Teile des Hintergrundes innerhalb der Kopfmaske liegen.
Für die Erstellung der Liquormaske sind keine Parametereinstellungen erforderlich. Diese Maske läßt sich aus allen Bildern ohne Problem erstellen. Die Ergebnisse der Segmentierung sind in Abbildung 4.8 dargestellt. Der Rand dieser Maske entspricht der inneren Kante des Knochens. Die Einblendung dieses Randes in die Bilder ist in Abbildung 4.9 zu sehen.
Auf die Segmentierung des Knochens haben die Parameter d_2 und d_3 Auswirkung. Ist d_2 klein, so werden unter Umständen Teile des Hintergrundes als Knochen segmentiert. Wird d_2 größer gewählt, so ist die Segmentierung des Knochens unvollständig. Gleiche Überlegungen erfordert die Wahl von d_3.
In Abbildung 4.8 ist das Ergebnis der Segmentierung für $d_1 = 6$, $d_2 = 7$ und $d_3 = 6$ dargestellt. In Abbildung 4.9 ist der Rand der Masken in die zu segmentierenden Bilder eingeblendet. Bei den Bildern 1 - 3 ist eine gute Knochensegmentierung zu erkennen, allerdings stimmt die segmentierte äußere Kante des Knochens nicht mit der in den Bildern sichtbaren überein. Dieser Fehler kann durch eine Änderung der Parameter d_2 und d_3 behoben werden. Bei den Bildern 4 und 5 fehlen Teile des Knochens in der Knochenmaske. Da allerdings bei diesen Bildern schon die Kopfmaske unvollständig ist und dadurch Teile des Knochens als Hintergrund klassifiziert werden, haben darauf auch die Parameter d_2 und d_3 keinen Einfluß.
Bei der Segmentierung unter Anwendung des Isodata-Algorithmus fällt auf, daß dieser Algorithmus zu teilweise großen Segmentierungsfehlern führt. Diese sind umso größer, je größer das Intervall ist, in dem die Grauwerte für Kopfhaut, Hirnflüssigkeit und Gehirn liegen. Theoretisch müßten diese Gewebetypen einen einheitlichen Grauwert besitzen, allerdings treten Intensitätsinhomogenitäten auf. Daher wird im nächsten Abschnitt ein Algorithmus vorgestellt, der Bilder mit vorhandenen Intensitätsinhomogenitäten segmentiert.

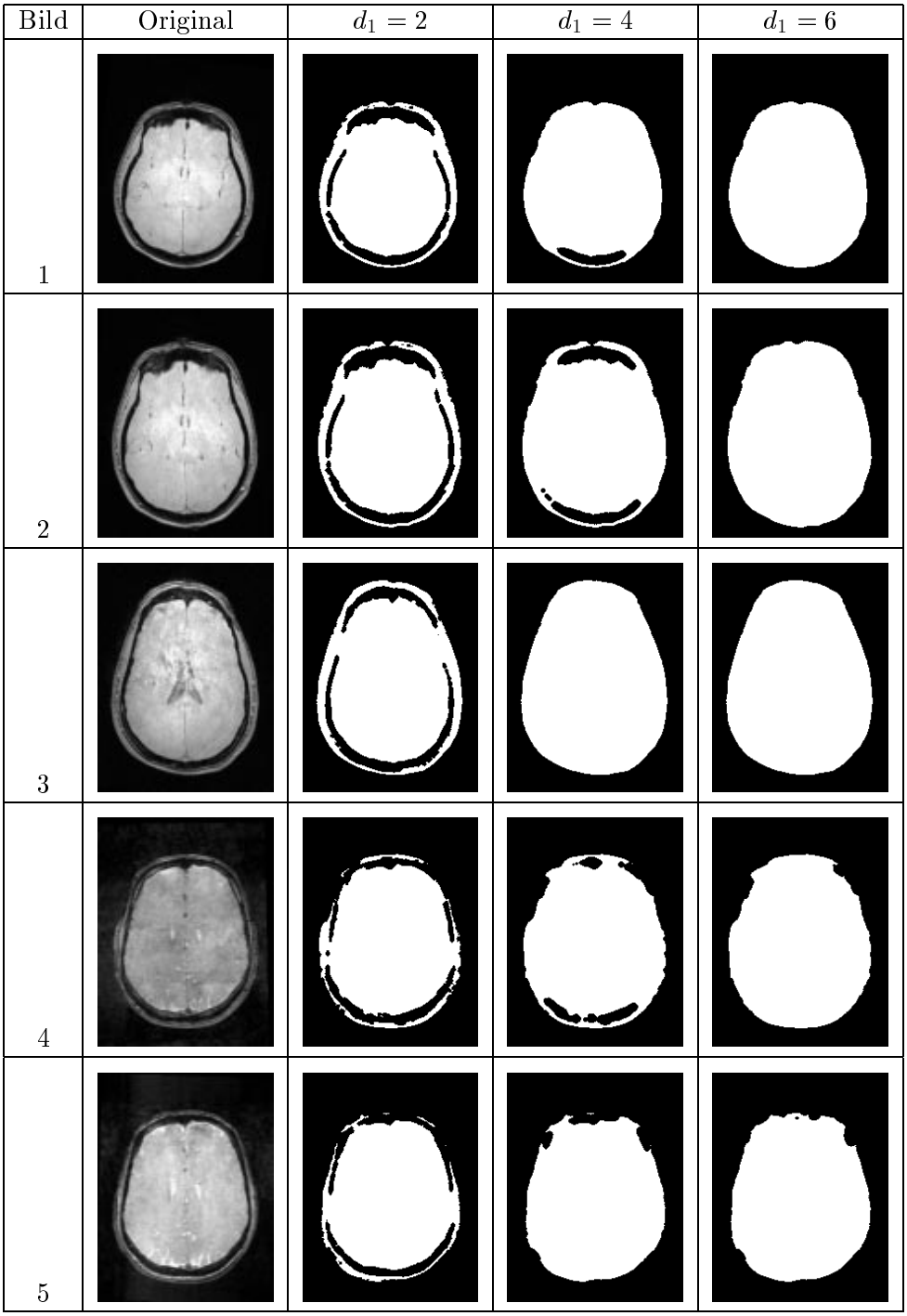


Abbildung 4.7: Kopfmasken für unterschiedliche Werte des Distanzparameters d_1 bei fünf Bildern, jeweils ein axialer Schnitt

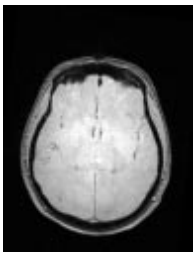
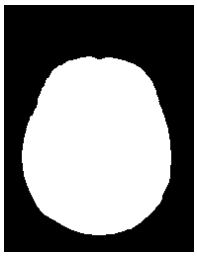
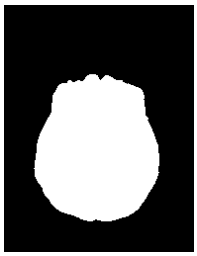
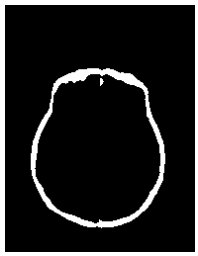
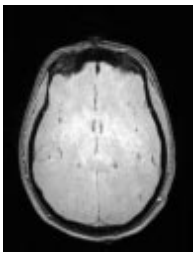
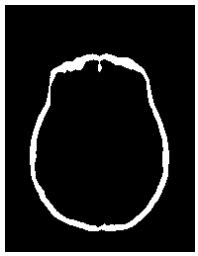
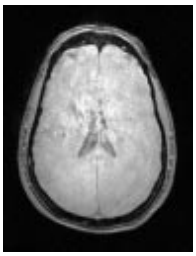
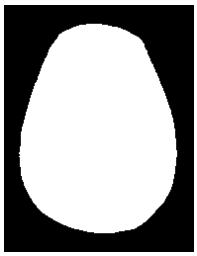
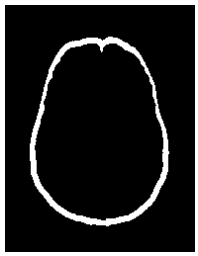
Bild	Original	Kopfmaske	Liquormaske	Knochen
1				
2				
3				
4				
5				

Abbildung 4.8: Komplette Segmentierung eines PD-gewichteten Kernspinbildes für fünf Bilder, jeweils ein axialer Schnitt

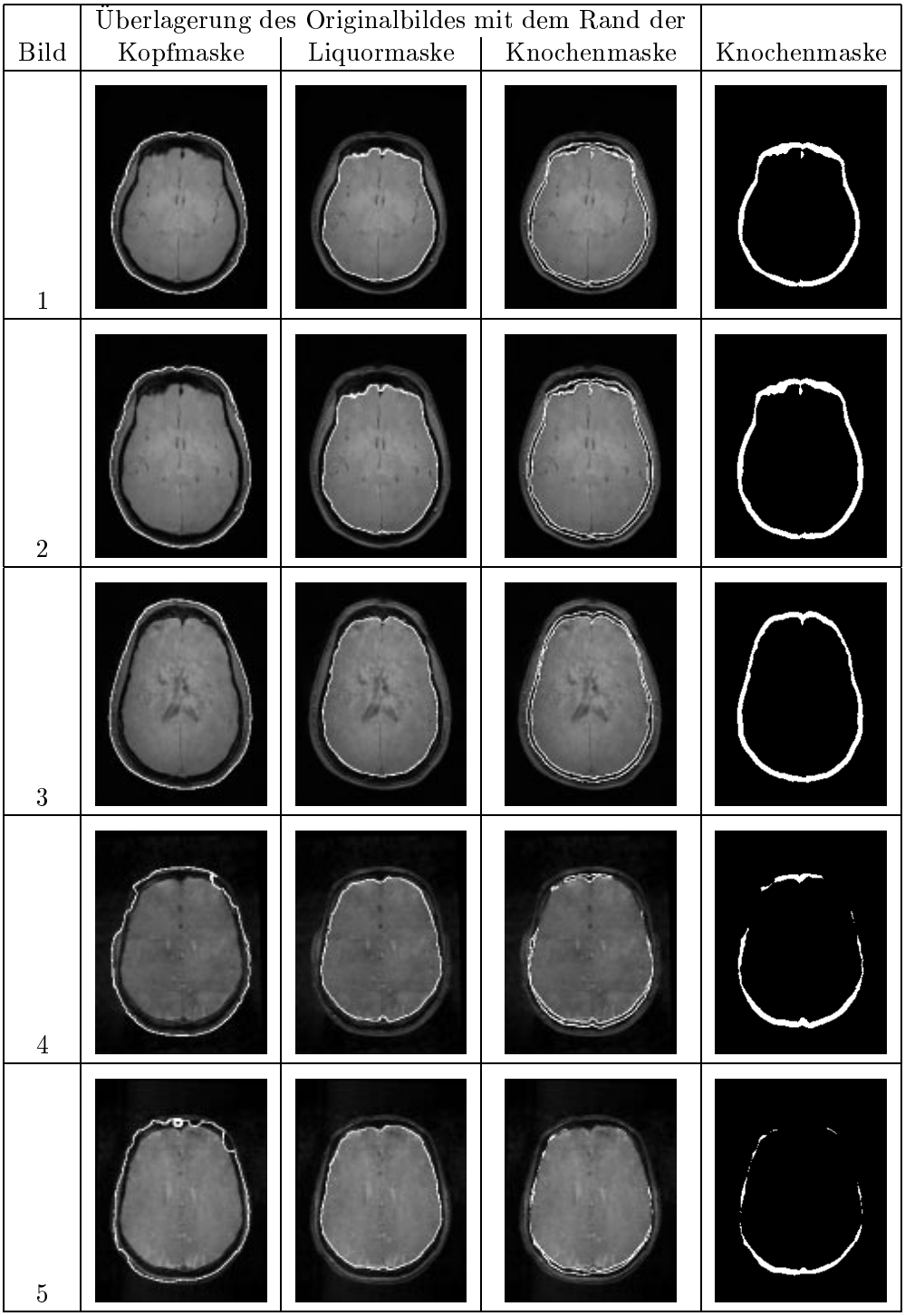


Abbildung 4.9: Einblendung des Randes der Kopf-, Liquor- und Knochenmaske in das Originalbild für fünf verschiedene Bilder, rechts zum Vergleich: Knochenmaske, axialer Schnitt

4.3 Ein adaptiver Fuzzy-C-Means Algorithmus

Der Isodata-Algorithmus hat den Nachteil, daß jedes Voxel genau einer Klasse zugeordnet wird. Bei den sogenannten *fuzzy* Segmentieralgorithmen besteht die Möglichkeit, daß jedes Voxel mehreren Klassen zugeordnet werden kann und damit mehr Informationen in der Segmentierung enthalten sind.
Der *Fuzzy-C-Means*-Algorithmus (kurz: FCM) segmentiert das Bild $b \in \mathcal{B}_\mathcal{T}$ so, daß J_{FCM} mit

$$J_{FCM} = \sum_{\mathbf{x}\in\mathcal{T}} \sum_{j=1}^{k} u_j(\mathbf{x})^q \|I(\mathbf{x}) - v_j\|^2 \tag{4.6}$$

minimiert wird. k bezeichnet die Anzahl der Klassen, v_j das Zentrum der Klasse j und $u_j(\mathbf{x}) \in [0,1]$ den Zugehörigkeitswert[1] von Voxel $\mathbf{x} \in \mathcal{T}$ in der Klasse j. q steuert die Unschärfe[2] der Segmentierung. Für die Zugehörigkeitswerte gilt:

$$\sum_{j=1}^{k} u_j(\mathbf{x}) = 1 \quad \text{für alle} \quad \mathbf{x} \in \mathcal{T}\,. \tag{4.7}$$

Entsprechend [28] wird im folgenden $q = 2$ gesetzt. J_{FCM} in Gleichung (4.6) wird minimiert, wenn hohe Zugehörigkeitswerte Voxeln zugewiesen werden, deren Grauwerte nahe dem Klassenzentrum liegen.
Der Vorteil des FCM-Algorithmus liegt darin, daß sich bei einem verrauschten Voxel die Segmentierung nur geringfügig ändert, während sich beim Isodata-Algorithmus die gesamte Klassifizierung ändern kann.
Dzung und Pham stellen in [28] einen modifizierten FCM-Algorithmus, den sogenannten *adaptiven Fuzzy-C-Means*-Algorithmus (kurz: AFCM), vor, der auf Bilder mit Intensitätsinhomogenitäten angewendet werden kann. In diesem Ansatz wird die Inhomogenität durch die Multiplikation der Klassenzentren mit einem unbekannten Korrekturfeld $m(\mathbf{x})$ modelliert. Es wird angenommen, daß sich das Korrekturfeld $m : \mathcal{T} \to \mathbb{R}$ langsam in Bezug auf $\mathbf{x}$ ändert.
Damit ist J_{AFCM} mit

$$\begin{aligned} J_{AFCM} \;=\; & \sum_{\mathbf{x}\in\mathcal{T}} \sum_{j=1}^{k} u_j(\mathbf{x})^2 \|I(\mathbf{x}) - m(\mathbf{x})v_j\|^2 \\ & + \lambda_1 \sum_{\mathbf{x}\in\mathcal{T}} \Big(((D_x * m)(\mathbf{x}))^2 + ((D_y * m)(\mathbf{x}))^2 + ((D_z * m)(\mathbf{x}))^2 \Big) \\ & + \lambda_2 \sum_{\mathbf{x}\in\mathcal{T}} \Big(((D_{xx} * m)(\mathbf{x}))^2 + ((D_{yy} * m)(\mathbf{x}))^2 + ((D_{zz} * m)(\mathbf{x}))^2 \\ & \quad + 2\,((D_{xy} * m)(\mathbf{x}))^2 + 2\,((D_{xz} * m)(\mathbf{x}))^2 \\ & \quad + 2\,((D_{yz} * m)(\mathbf{x}))^2 \Big) \end{aligned} \tag{4.8}$$

[1] Übersetzung aus dem Englischen: *membership value*
[2] Übersetzung aus dem Englischen: *fuzziness*

in Abhängigkeit von $u_j(\mathbf{x}), v_j$ und $m(\mathbf{x})$ zu minimieren. D_i sind die Standard-Vorwärtsdifferenzen

$$(D_i * m)(\mathbf{x}) = m(\mathbf{x} + \mathbf{e}_i) - m(\mathbf{x}) \qquad i \in \{x, y, z\}$$

mit $\mathbf{e}_x = (1,0,0)^T, \mathbf{e}_y = (0,1,0)^T$ und $\mathbf{e}_z = (0,0,1)^T$. Die D_{ij} ergeben sich durch

$$(D_{ij} * m)(\mathbf{x}) = (D_i * (D_j * m))(\mathbf{x}) \qquad i, j \in \{x, y, z\} .$$

Die beiden Terme, die von λ_1 und λ_2 gewichtet werden, sind Regularisierungsterme erster und zweiter Ordnung für $m(\mathbf{x})$. Sind die Klassenzentren v_j und die Zugehörigkeitswerte $u_j(\mathbf{x})$ bekannt, minimiert $m(\mathbf{x})$ J_{AFCM}, indem die Klassenzentren so multipliziert werden, daß sie nahe den Grauwerten der einzelnen Voxel sind. Sind jeweils zwei der drei Terme $u_j(\mathbf{x})$, v_j und $m(\mathbf{x})$ bekannt, so kann der dritte so bestimmt werden, daß J_{AFCM} minimal ist.
Sind die v_j und $m(\mathbf{x})$ bekannt, so kann $u_j(\mathbf{x})$ bestimmt werden. Dazu sind für jedes $\mathbf{x} \in \mathcal{T}$ die $u_j(\mathbf{x})$ so zu bestimmen, daß $J_\mathbf{x}$ mit

$$J_\mathbf{x} = \sum_{j=1}^{k} u_j(\mathbf{x})^2 \|I(\mathbf{x}) - m(\mathbf{x})v_j\|^2 \tag{4.9}$$

minimiert wird unter Einhaltung der Bedingung in Gleichung 4.7. Die Anwendung der Lagrangeschen Multiplikatorregel führt auf $J^*_\mathbf{x}$ mit

$$J^*_\mathbf{x} = \sum_{j=1}^{k} u_j(\mathbf{x})^2 \|I(\mathbf{x}) - m(\mathbf{x})v_j\|^2 + \mu \left(\sum_{j=1}^{k} u_j(\mathbf{x}) - 1 \right) \tag{4.10}$$

als zu minimierende Funktion. Die Ableitung nach $u_i(\mathbf{x})$ ist

$$\frac{\partial J^*_\mathbf{x}}{\partial u_i(\mathbf{x})} = 2u_i(\mathbf{x})\|I(\mathbf{x}) - m(\mathbf{x})v_i\|^2 + \mu \tag{4.11}$$

Das Nullsetzen der Ableitung und die Umstellung der entstehenden Gleichung führt auf die folgende Gleichung zur Bestimmung von $u_i(\mathbf{x})$:

$$u_i(\mathbf{x}) = -\frac{1}{2}\mu\|I(\mathbf{x}) - m(\mathbf{x})v_i\|^{-2} \tag{4.12}$$

wobei nun noch μ bestimmt werden muß. Ein Umstellen der letzten Gleichung liefert

$$\mu = -2u_i(\mathbf{x})\|I(\mathbf{x}) - m(\mathbf{x})v_i\|^2 \tag{4.13}$$

Das Ersetzen von $u_i(\mathbf{x})$ durch

$$1 - \sum_{\substack{j=1 \\ i \neq j}}^{k} u_j(\mathbf{x})$$

und Ersetzen der $u_j(\mathbf{x})$ unter Verwendung von Gleichung 4.12 führt auf

$$\mu = -2 \left(1 + \frac{1}{2}\mu \sum_{\substack{j=1 \\ i \neq j}}^{k} \|I(\mathbf{x}) - m(\mathbf{x})v_j\|^{-2} \right) \|I(\mathbf{x}) - m(\mathbf{x})v_i\|^2 \,. \tag{4.14}$$

Das Umstellen nach μ führt schließlich auf

$$\mu = -\frac{2}{\sum_{j=1}^{k} \|I(\mathbf{x}) - m(\mathbf{x})v_j\|^{-2}} \tag{4.15}$$

und das Einsetzen in Gleichung 4.12 führt auf die Gleichung

$$u_i(\mathbf{x}) = \frac{\|I(\mathbf{x}) - m(\mathbf{x})v_i\|^{-2}}{\sum_{j=1}^{k} \|I(\mathbf{x}) - m(\mathbf{x})v_j\|^{-2}} \,, \tag{4.16}$$

nach der alle $u_i(\mathbf{x})$, $i = 1, \ldots, k$, $\mathbf{x} \in \mathcal{T}$ berechnet werden können.
Für die Berechnung der v_i, $i = 1, \ldots, k$, wird die Ableitung von J_{AFCM} nach v_i gebildet. Dabei wird ausgenutzt, daß $I(\mathbf{x}), m(\mathbf{x})$ und v_i reelle Zahlen sind und damit

$$\|I(\mathbf{x}) - m(\mathbf{x})v_i\|^2 = (I(\mathbf{x}) - m(\mathbf{x})v_i)^2$$

gilt. Das führt auf

$$\frac{\partial J_{AFCM}}{\partial v_i} = \sum_{\mathbf{x} \in \mathcal{T}} 2u_i(\mathbf{x})^2 \left(m(\mathbf{x})^2 v_i - m(\mathbf{x})I(\mathbf{x}) \right) \,. \tag{4.17}$$

Das Nullsetzen der Ableitung und Umstellen der entstehenden Gleichung führt auf

$$v_i = \frac{\sum_{\mathbf{x} \in \mathcal{T}} u_i(\mathbf{x})^2 m(\mathbf{x}) I(\mathbf{x})}{\sum_{\mathbf{x} \in \mathcal{T}} u_i(\mathbf{x})^2 m(\mathbf{x})^2} \tag{4.18}$$

Sind $u_j(\mathbf{x})$ und v_j bekannt, so kann daraus $m(\mathbf{x})$ berechnet werden. Die Ableitung von J_{AFCM} nach $m(\mathbf{x})$ führt auf

$$\begin{aligned} \frac{\partial J_{AFCM}}{\partial m(\mathbf{x})} &= \sum_{j=1}^{k} -2u_j(\mathbf{x})^2 v_j (I(\mathbf{x}) - m(\mathbf{x})v_j) \\ &\quad + 2\lambda_1 \left(H_1 * m \right)(\mathbf{x}) + 2\lambda_2 \left(H_2 * m \right)(\mathbf{x}) \end{aligned} \tag{4.19}$$

mit

$$\begin{aligned}
(H_1 * m)(\mathbf{x}) &= 6m(\mathbf{x}) - m(\mathbf{x}+\mathbf{e}_x) - m(\mathbf{x}-\mathbf{e}_x) \\
&\quad - m(\mathbf{x}+\mathbf{e}_y) - m(\mathbf{x}-\mathbf{e}_y) - m(\mathbf{x}+\mathbf{e}_z) - m(\mathbf{x}-\mathbf{e}_z)\ , \\
(H_2 * m)(\mathbf{x}) &= 42m(\mathbf{x}) \\
&\quad - 12m(\mathbf{x}+\mathbf{e}_x) - 12m(\mathbf{x}-\mathbf{e}_x) \\
&\quad - 12m(\mathbf{x}+\mathbf{e}_y) - 12m(\mathbf{x}-\mathbf{e}_y) \\
&\quad - 12m(\mathbf{x}+\mathbf{e}_z) - 12m(\mathbf{x}-\mathbf{e}_z) \\
&\quad + 2m(\mathbf{x}+\mathbf{e}_x+\mathbf{e}_y) + 2m(\mathbf{x}+\mathbf{e}_x-\mathbf{e}_y) \\
&\quad + 2m(\mathbf{x}-\mathbf{e}_x+\mathbf{e}_y) + 2m(\mathbf{x}-\mathbf{e}_x-\mathbf{e}_y) \\
&\quad + 2m(\mathbf{x}+\mathbf{e}_x+\mathbf{e}_z) + 2m(\mathbf{x}+\mathbf{e}_x-\mathbf{e}_z) \\
&\quad + 2m(\mathbf{x}-\mathbf{e}_x+\mathbf{e}_z) + 2m(\mathbf{x}-\mathbf{e}_x-\mathbf{e}_z) \\
&\quad + 2m(\mathbf{x}+\mathbf{e}_y+\mathbf{e}_z) + 2m(\mathbf{x}+\mathbf{e}_y-\mathbf{e}_z) \\
&\quad + 2m(\mathbf{x}-\mathbf{e}_y+\mathbf{e}_z) + 2m(\mathbf{x}-\mathbf{e}_y-\mathbf{e}_z) \\
&\quad + m(\mathbf{x}+2\mathbf{e}_x) + m(\mathbf{x}-2\mathbf{e}_x) \\
&\quad + m(\mathbf{x}+2\mathbf{e}_y) + m(\mathbf{x}-2\mathbf{e}_y) \\
&\quad + m(\mathbf{x}+2\mathbf{e}_z) + m(\mathbf{x}-2\mathbf{e}_z)\ .
\end{aligned}$$

Das Nullsetzen der Ableitung und Umstellen der Gleichung führt auf

$$\begin{aligned}
I(\mathbf{x}) \sum_{j=1}^{k} u_j(\mathbf{x})^2 v_j &= m(\mathbf{x}) \sum_{j=1}^{k} u_j(\mathbf{x})^2 v_j^2 \\
&\quad + \lambda_1 (H_1 * m)(\mathbf{x}) + \lambda_2 (H_2 * m)(\mathbf{x})\ . \qquad (4.20)
\end{aligned}$$

Die Anwendung der Gleichungen für $u_j(\mathbf{x})$, v_j und $m(\mathbf{x})$ ergibt den iterativen Algorithmus, der in Abbildung 4.10 dargestellt ist.
Der Algorithmus besteht aus fünf Schritten. Nach der Initialisierung der Klassenzentren und des Korrekturfeldes werden für alle Voxel die Zugehörigkeitswerte zu den Klassen und daraus die neuen Klassenzentren berechnet. Im vierten Schritt wird das Korrekturfeld m neu berechnet. Ist die maximale Änderung in den Zugehörigkeitswerten über alle Voxel und Klassen kleiner als eine Schranke ε, wird der Algorithmus beendet. Ansonsten wird mit der erneuten Berechnung der Zugehörigkeitswerte fortgefahren. Wie in [28] wird $\epsilon = 0,01$ gesetzt.

4.3.1 Berechnung des Korrekturfeldes

Im AFCM-Algorithmus wird bei jeder Iteration das Korrekturfeld m als Lösung der Gleichung

$$\begin{aligned}
I(\mathbf{x}) \sum_{j=1}^{k} u_j(\mathbf{x})^2 v_j &= m(\mathbf{x}) \sum_{j=1}^{k} u_j(\mathbf{x})^2 v_j^2 \\
&\quad + \lambda_1 (H_1 * m)(\mathbf{x}) + \lambda_2 (H_2 * m)(\mathbf{x}) \qquad (4.21)
\end{aligned}$$

Der AFCM-Algorithmus zur Segmentierung eines Bildes in k Klassen

Initialisierung der Klassenzentren $v_j^{(0)}$ *und Setzen des Korrekturfeldes* $m^{(0)}(\mathbf{x}) = 1$ *für alle* $\mathbf{x} \in \mathcal{T}$.

$i = 0$;

DO

{

$i = i + 1$;

Berechnung der Zugehörigkeitswerte:

$$u_j^{(i)}(\mathbf{x}) = \frac{\| I(\mathbf{x}) - \left(m^{(i-1)}(\mathbf{x})\right) \left(v_j^{(i-1)}\right) \|^{-2}}{\sum_{l=1}^{k} \| I(\mathbf{x}) - \left(m^{(i-1)}(\mathbf{x})\right) \left(v_l^{(i-1)}\right) \|^{-2}}$$

für alle $j = 1, \ldots, k$ *und* $\mathbf{x} \in \mathcal{T}$.

Berechnung der neuen Klassenzentren:

$$v_j^{(i)} = \frac{\sum_{\mathbf{x} \in \mathcal{T}} \left(u_j^{(i)}(\mathbf{x})\right)^2 \left(m^{(i-1)}(\mathbf{x})\right) I(\mathbf{x})}{\sum_{\mathbf{x} \in \mathcal{T}} \left(u_j^{(i)}(\mathbf{x})\right)^2 \left(m^{(i-1)}(\mathbf{x})\right)^2}$$

für alle $j = 1, \ldots, k$.

Berechnung des neuen Korrekturfeldes durch Lösen der Gleichung:

$$\begin{aligned} I(\mathbf{x}) \sum_{j=1}^{k} \left(u_j^{(i)}(\mathbf{x})\right)^2 v_j^{(i)} \quad = \quad & m^{(i)}(\mathbf{x}) \sum_{j=1}^{k} \left(u_j^{(i)}(\mathbf{x})\right)^2 \left(v_j^{(i)}\right)^2 \\ & + \lambda_1 \left(H_1 * m^{(i)}\right)(\mathbf{x}) \\ & + \lambda_2 \left(H_2 * m^{(i)}\right)(\mathbf{x}) \end{aligned}$$

}

WHILE $(\max_{\mathbf{x} \in \mathcal{T}, j=1,\ldots,k} |u_j^{(i)}(\mathbf{x}) - u_j^{(i-1)}(\mathbf{x})| \geq \varepsilon)$

Abbildung 4.10: Der adaptive Fuzzy-C-Means (AFCM)-Algorithmus

berechnet. Gleichung (4.21) kann auch als Vektor-Matrix-Gleichung

$$\mathbf{f} = \mathbf{W}\mathbf{m} + (\lambda_1\mathbf{H}_1 + \lambda_2\mathbf{H}_2)\mathbf{m} \tag{4.22}$$

geschrieben werden, wobei $\mathbf{f}$ und $\mathbf{m}$ die Werte von $I(\mathbf{x})\sum_{j=1}^{k} u_j(\mathbf{x})^2 v_j$ bzw. $m(\mathbf{x})$, als Vektor angeordnet, enthalten. $\mathbf{W}$ ist eine Diagonalmatrix mit den Elementen $\sum_{j=1}^{k} u_j(\mathbf{x})^2 v_j^2$. $\mathbf{H}_1$ und $\mathbf{H}_2$ sind die Matrix-Varianten von H_1 und H_2. Damit kann $m(\mathbf{x})$ als Lösung dieses linearen Gleichungssystems berechnet werden.

Probleme bei der ursprünglichen Implementierung

Eine erste Implementierung [17] verwendet zur Lösung der Gleichung (4.22) ein Jacobi-Verfahren.
Gleichung (4.22) kann als

$$\mathbf{f} = \mathbf{A}\mathbf{m} \quad \text{mit} \quad \mathbf{A} = (\mathbf{W} + \lambda_1\mathbf{H}_1 + \lambda_2\mathbf{H}_2)$$

geschrieben werden. Die Matrix $\mathbf{A}$ kann durch

$$\mathbf{A} = \mathbf{D} - \mathbf{L} - \mathbf{R} \tag{4.23}$$

in eine Diagonalmatrix $\mathbf{D}$, eine untere Dreiecksmatrix $\mathbf{L}$ und eine obere Dreiecksmatrix $\mathbf{R}$ zerlegt werden. Das Jacobi-Verfahren berechnet ausgehend von einem Startwert $\mathbf{m}^{(0)}$ iterativ die Lösung des Gleichungssystems, wobei ein Iterationschritt durch

$$\mathbf{m}^{(i)} = \left[(1-\omega)\mathbf{E} + \omega\mathbf{D}^{-1}(\mathbf{L}+\mathbf{U})\right]\mathbf{m}^{(i-1)} + \omega\mathbf{D}^{-1}\mathbf{f} \tag{4.24}$$

gegeben ist. Dabei bezeichnet $\mathbf{E}$ die Einheitsmatrix, und mit dem Parameter ω kann die Konvergenzgeschwindigkeit des Verfahrens gesteuert werden.
Werden λ_1 und λ_2 nicht zu groß gewählt, so erfüllt die Matrix $\mathbf{A}$ wegen der positiven Diagonalelemente in $\mathbf{W}$ sowohl das Zeilensummen- als auch das positive Definitheitskriterium [32]. Damit konvergiert das Verfahren für jeden Wert $\omega \in (0,2)$, und es gilt das notwendige Konvergenzkriterium:

$$\lim_{i\to\infty} \|\mathbf{m}^{(i)} - \mathbf{m}^{(i-1)}\| = 0\,. \tag{4.25}$$

Die Iterationen wurden in dem Schritt i abgebrochen, in dem erstmals

$$\|\mathbf{m}^{(i)} - \mathbf{m}^{(i-1)}\| < \varepsilon \tag{4.26}$$

galt, das heißt, der Abstand zwischen zwei aufeinanderfolgenden Näherungslösungen eine vorgegebene Schranke unterschritt.
Bei der Implementierung dieses Verfahrens fiel auf, daß trotz erfülltem Konvergenzkriterium der Wert $\|\mathbf{m}^{(i)} - \mathbf{m}^{(i-1)}\|$ nicht gegen 0 konvergiert, sondern vielmehr einen bestimmten Wert, der von ω abhängig ist, nicht unterschreitet. Für eine genauere Untersuchung dieser Tatsache wurde die Lösungsgenauigkeit verwendet.
Ist $\mathbf{m}$ eine Lösung der Gleichung $\mathbf{A}\mathbf{m} = \mathbf{f}$, dann gilt

$$\mathbf{Am} - \mathbf{f} = 0$$

Im Iterationsschritt i kann das *Residuum* $\mathbf{r}^{(i)}$ durch

$$\mathbf{r}^{(i)} := \mathbf{Am}^{(i)} - \mathbf{f} \tag{4.27}$$

berechnet werden. Ist $\mathbf{m}$ eine Lösung, so ist $\mathbf{r} = 0$. Konvergiert die Folge $\mathbf{m}^{(i)}$ gegen die Lösung des Gleichungssystems, gilt

$$\lim_{i \to \infty} \mathbf{r}^{(i)} = 0 \tag{4.28}$$

und damit auch

$$\lim_{i \to \infty} \|\mathbf{r}^{(i)}\| = 0 \tag{4.29}$$

bezüglich einer beliebigen Norm. Im folgenden wird stets die euklidische Norm verwendet.
In Abbildung 4.11 ist der Verlauf von $\|\mathbf{r}^{(i)}\|$ für verschiedene Werte von ω dargestellt. Dabei ist die Norm anfangs fallend, steigt dann aber ab einer von ω abhängigen Iterationszahl an. In diesem Versuch wurden die ersten 15000 Iterationen betrachtet. In diesem Bereich fällt die Norm des Residuums nicht mehr. Sollte das Verfahren konvergieren, so sind mehr als 15000 Iterationen erforderlich. Aus diesem Grund wurde für die Implementierung auf ein anderes Verfahren zum Lösen des linearen Gleichungssystems zurückgegriffen.

Der CG-Algorithmus

Eine weitere Möglichkeit, das lineare Gleichungssystem zu lösen, ist das Verfahren der konjugierten Gradienten. Die Beschreibung des Algorithmus ist an H.R.Schwarz (siehe [32]) angelehnt, wo sich auch eine detailliertere Herleitung finden läßt.
Das Lösen des Gleichungssystems

$$\mathbf{Am} - \mathbf{f} = 0$$

ist gleichbedeutend damit, das Funktional

$$F(\mathbf{m}) = \frac{1}{2}\langle \mathbf{m}, \mathbf{Am} \rangle - \langle \mathbf{f}, \mathbf{m} \rangle \tag{4.30}$$

zu minimieren. Das Minimum von $F(\mathbf{m})$ wird iterativ bestimmt, indem in jedem Schritt die Richtung des Gradienten verwendet wird. Der Gradient

$$\nabla F(\mathbf{m}) = \mathbf{Am} - \mathbf{f} = \mathbf{r} \tag{4.31}$$

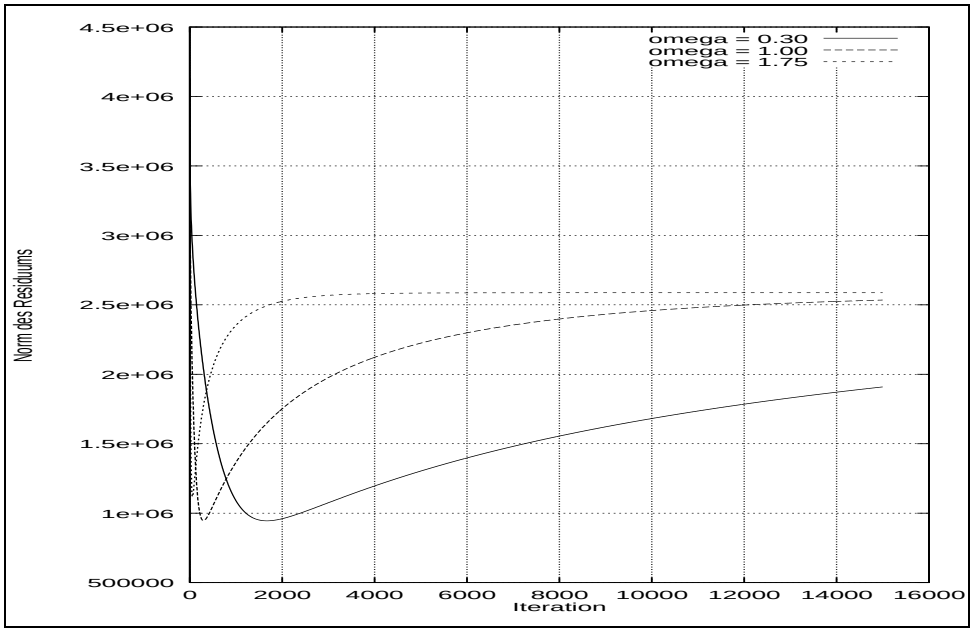


Abbildung 4.11: Norm des Residuums bei der Lösung des Gleichungssystems mit dem Jacobi-Verfahren und verschiedenen Werten von ω während der ersten 15000 Iterationen

ist gleich dem Residuum. Weil der Gradient in die Richtung des stärksten Anstiegs von $F(\mathbf{m})$ zeigt, wird die Richtung des negativen Gradienten verwendet, um das Minimum zu finden.
Sei $\mathbf{m}^{(0)}$ ein Startvektor. Um im ersten Iterationsschritt $\mathbf{m}^{(1)}$ zu bestimmen mit $F(\mathbf{m}^{(1)}) < F(\mathbf{m}^{(0)})$, wird als Relaxationsrichtung $\mathbf{p}^{(1)}$ der negative Residuenvektor $\mathbf{r}^{(0)} = \mathbf{A}\mathbf{m}^{(0)} - \mathbf{f}$ verwendet:

$$\mathbf{p}^{(1)} = -\mathbf{r}^{(0)} = -\nabla F(\mathbf{m}^{(0)}) = -\left(\mathbf{A}\mathbf{m}^{(0)} - \mathbf{f}\right) . \tag{4.32}$$

Von $\mathbf{m}^{(0)}$ ausgehend wird in Richtung $\mathbf{p}^{(1)}$ das Minimum von $F(\mathbf{m})$ gesucht. Es ist durch

$$\mathbf{m}^{(1)} = \mathbf{m}^{(0)} + q_1\mathbf{p}^{(1)} \tag{4.33}$$

mit einem noch zu ermittelndem Wert q_1 bestimmt. Einsetzen von $\mathbf{m}^{(1)}$ in Gleichung 4.30 führt auf die Gleichung

$$F(\mathbf{m}^{(1)}) = \frac{1}{2}q_1^2\langle\mathbf{p}^{(1)}, \mathbf{A}\mathbf{p}^{(1)}\rangle + q_1\langle\mathbf{p}^{(1)}, \mathbf{r}^{(0)}\rangle + F(\mathbf{m}^{(0)}) , \tag{4.34}$$

die zu minimieren ist. Die Ableitung nach q_1 führt auf

$$\frac{\mathrm{d}F}{\mathrm{d}q_1} = q_1\langle\mathbf{p}^{(1)}, \mathbf{A}\mathbf{p}^{(1)}\rangle + \langle\mathbf{p}^{(1)}, \mathbf{r}^{(0)}\rangle . \tag{4.35}$$

Durch Nullsetzen der Ableitung ergibt sich q_1 als

$$q_1 = -\frac{\langle \mathbf{p}^{(1)}, \mathbf{r}^{(0)} \rangle}{\langle \mathbf{p}^{(1)}, \mathbf{A}\mathbf{p}^{(1)} \rangle} = \frac{\langle \mathbf{r}^{(0)}, \mathbf{r}^{(0)} \rangle}{\langle \mathbf{p}^{(1)}, \mathbf{A}\mathbf{p}^{(1)} \rangle} \; . \tag{4.36}$$

Allgemein wird im k-ten Iterationsschritt die Relaxationsrichtung $\mathbf{p}^{(k)}$ als Linearkombination von $\mathbf{r}^{(k-1)}$ und $\mathbf{p}^{(k-1)}$ so bestimmt, daß die beiden Relaxationsrichtungen konjugiert zueinander sind, d.h. daß

$$\langle \mathbf{p}^{(k)}, \mathbf{A}\mathbf{p}^{(k-1)} \rangle = 0 \tag{4.37}$$

gilt. Diese Forderung kann geometrisch begründet werden (siehe [32]). Damit ist der Ansatz für die Relaxationsrichtung

$$\mathbf{p}^{(k)} = -\mathbf{r}^{(k-1)} + e_{k-1}\mathbf{p}^{(k-1)} \tag{4.38}$$

mit $k \geq 2$. Unter Verwendung der Gleichung 4.37 ist e_{k-1} durch

$$e_{k-1} = \frac{\langle \mathbf{r}^{(k-1)}, \mathbf{A}\mathbf{p}^{(k-1)} \rangle}{\langle \mathbf{p}^{(k-1)}, \mathbf{A}\mathbf{p}^{(k-1)} \rangle} \tag{4.39}$$

bestimmt. Mit dieser und der Gleichung 4.37 ergibt sich die Näherungslösung im Iterationsschritt k durch

$$\mathbf{m}^{(k)} = \mathbf{m}^{(k-1)} + q_k \mathbf{p}^{(k)} \; . \tag{4.40}$$

Analog der Herleitung zu Gleichung 4.36 ergibt sich q_k als

$$q_k = -\frac{\langle \mathbf{p}^{(k)}, \mathbf{r}^{(k-1)} \rangle}{\langle \mathbf{p}^{(k)}, \mathbf{A}\mathbf{p}^{(k)} \rangle} \; . \tag{4.41}$$

Die Iterationsschritte werden solange ausgeführt, bis die Norm des Residuums einen festen, vorgegebenen Wert unterschreitet.
Das führt auf den Algorithmus, der in Abbildung 4.12 dargestellt ist.

Skalierung

Eine einfache Maßnahme, die Anzahl der Iterationsschritte bis zur Konvergenz zu verringern, ist die Skalierung der Matrix $\mathbf{A}$. Dabei wird die i-te Zeile und Spalte mit einem positiven Wert $d_i > 0$ multipliziert. Um die Symmetrie der Matrix zu erhalten, müssen gleichzeitig Zeilen- und Spaltenskalierungen durchgeführt werden.
Eine optimale Skalierung ist erreicht, wenn für alle Zeilen i

$$d_i \cdot \sqrt{\sum_{j=1}^{n} (a_{ij} d_j)^2} = 1 \tag{4.42}$$

Verfahren der konjugierten Gradienten zum Lösen des linearen Gleichungssystems $\mathbf{Am} - \mathbf{f} = 0$

Start mit $\mathbf{m}^{(0)}$

$\mathbf{r}^{(0)} = \mathbf{Am}^{(0)} - \mathbf{f}$;

$\mathbf{p}^{(1)} = -\mathbf{r}^{(0)}$;

$k = 0$;

WHILE $(\|\mathbf{r}^{(k)}\| > \varepsilon)$ DO

{

$k = k + 1$;

IF $(k \geq 2)$

{

$e_{k-1} = \frac{\langle \mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \rangle}{\langle \mathbf{r}^{(k-2)}, \mathbf{r}^{(k-2)} \rangle}$;

$\mathbf{p}^{(k)} = -\mathbf{r}^{(k-1)} + e_{k-1}\mathbf{p}^{(k-1)}$;

}

$q_k = \frac{\langle \mathbf{r}^{(k-1)}, \mathbf{r}^{(k-1)} \rangle}{\langle \mathbf{p}^{(k)}, \mathbf{Ap}^{(k)} \rangle}$;

$\mathbf{m}^{(k)} = \mathbf{m}^{(k-1)} + q_k \mathbf{p}^{(k)}$;

$\mathbf{r}^{(k)} = \mathbf{r}^{(k-1)} + q_k \left(\mathbf{Ap}^{(k)}\right)$;

}

Ende mit Ergebnis $\mathbf{m}^{(k)}$

Abbildung 4.12: Das Verfahren der konjugierten Gradienten zum Lösen des linearen Gleichungssystems $\mathbf{Am} - \mathbf{f} = 0$

gilt. Das Auflösen dieser nichtlinearen Gleichung zur Bestimmung der Skalierungsfaktoren d_i ist jedoch sehr aufwendig. Allerdings läßt sich in vielen Fällen durch die Wahl von

$$d_i = \frac{1}{\sqrt{a_{ii}}} \tag{4.43}$$

bereits die Anzahl der Iterationsschritte veringern.

In Abbildung 4.13 ist der Verlauf der Norm des Residuums für ein Beispiel bei Verwendung des skalierten CG-Verfahrens dargestellt. Im Gegensatz zum Jacobi-Verfahren für dasselbe Beispiel (siehe Abbildung 4.11) konvergiert es.

Das Verfahren wurde auf der Origin implementiert. Alle Vektoroperationen sind unter Verwendung von acht Prozessoren parallelisiert worden.

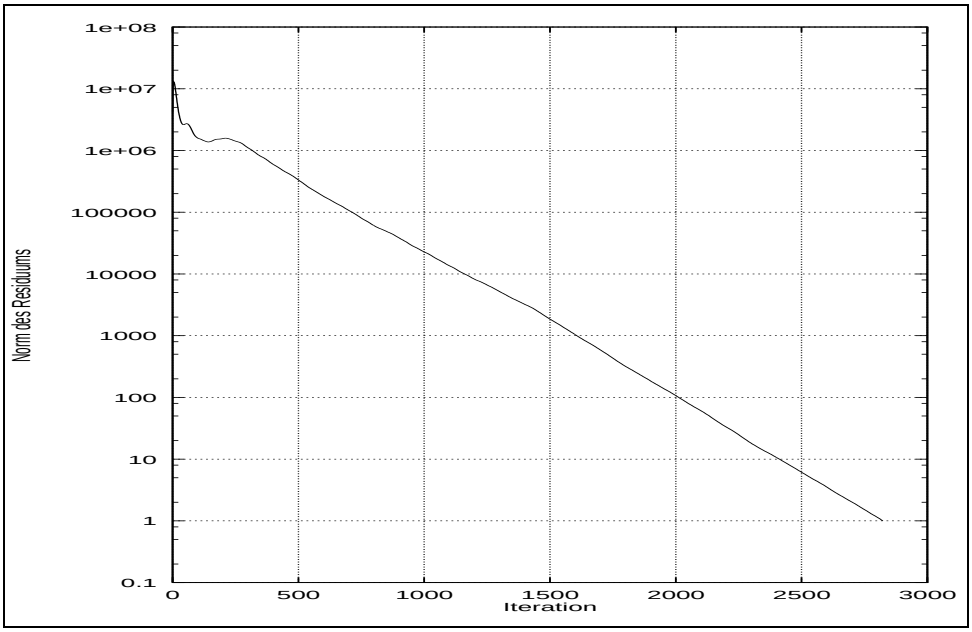


Abbildung 4.13: Euklidische Norm des Residuums bei der Lösung des Gleichungssystems mit dem skalierten CG-Verfahren

4.3.2 Beispiele für die Segmentierung

Als Ergebnis für ein Bild $b = (\mathcal{T}, \mathcal{G}, I)$ liefert der AFCM-Algorithmus für jedes Voxel $\mathbf{x} \in \mathcal{T}$ einen Wert des Korrekturfeldes $m(\mathbf{x})$ und k Zugehörigkeitswerte $u_1(\mathbf{x}), \ldots, u_k(\mathbf{x})$. Anhand des Korrekturfeldes kann ein intensitätskorrigiertes Bild $b_k = (\mathcal{T}, \mathcal{G}, I_k)$ mit $I_k(\mathbf{x}) = I(\mathbf{x})/m(\mathbf{x})$ erstellt werden.
Zur weiteren Arbeit mit der Segmentierung können die Zugehörigkeitswerte verwendet werden. Es ist aber auch möglich, anhand dieser Werte jedes Voxel ähnlich wie beim Isodata-Algorithmus genau einer Klasse zuzuweisen. Im folgenden wird die sogenannte *maximum-membership*-Segmentierung verwendet. Dabei wird ein Voxel $\mathbf{x}$ der Klasse i genau dann zugewiesen, wenn

$$u_i(\mathbf{x}) \geq u_j(\mathbf{x}) \quad j = 1, \ldots, k$$

gilt. Das Ergebnis ist ein Bild $b_{seg} = (\mathcal{T}, \mathbb{N}_0, I_{seg})$, wobei $I_{seg}(\mathbf{x})$ die Klasse bezeichnet, zu der das Voxel $\mathbf{x}$ gehört. $I_{seg}(\mathbf{x})$ wird als

$$I_{seg}(\mathbf{x}) = i - 1 \quad \text{genau dann, wenn} \quad u_i(\mathbf{x}) \geq u_j(\mathbf{x}) \quad j = 1, \ldots, k$$

definiert. Die Numerierung der Klassen beginnt dabei bei 0 und geht bis zu $k - 1$. Im weiteren wird dafür die Schreibweise

$$b_{seg} = \mathrm{afcm}(b, k) \ ,$$

mit der Bedeutung verwendet, daß ein Bild b unter Anwendung des AFCM-Algorithmus in k Klassen klassifiziert wird. Jedes Voxel wird gemäß der maximum-membership-Segmentierung einer Klasse zugewiesen.

Als Beispiele für eine Segmentierung mit dem AFCM-Algorithmus sollen drei Bilder dienen: ein synthetisches Bild, ein T_1- und ein PD-gewichtetes Kernspinbild.
Das synthetische Bild mit einer Größe von $128 \times 128 \times 128$ Voxeln ist eine Dreischalenkugel. Der Mittelpunkt der Kugel befindet sich bei dem Voxel $(64, 64, 64)^T$, die Schalen haben die Radien 50, 45 und 40 Voxel. Der Hintergrund und die mittlere Schicht besitzen den Grauwert 50, die beiden anderen Schalen den Grauwert 150. In diesem Bild ist eine Inhomogenität modelliert, indem der Grauwert jedes Voxels $\mathbf{x}$ mit einem bestimmten Wert $k(\mathbf{x})$ multipliziert wurde, der sich als

$$k(\mathbf{x}) = 1 + \frac{1}{2} \sin\left(\frac{x - 64}{64}\pi\right) \sin\left(\frac{y - 64}{64}\pi\right) \sin\left(\frac{z - 64}{64}\pi\right) \tag{4.44}$$

mit $\mathbf{x} = (x, y, z)^T$ ergibt.
Für den Test der Segmentierung erfolgt eine Klassifizierung des synthetischen und des PD-gewichteten Bildes in zwei Klassen und des T_1-gewichteten Bildes in drei Klassen. In Abbildung 4.14 ist jeweils das Originalbild, das intensitätskorrigierte Bild, die Segmentierung unter Verwendung des AFCM-Algorithmus und zum Vergleich das Resultat mit dem Isodata-Algorithmus dargestellt.
Es läßt sich erkennen, daß der AFCM-Algorithmus ein deutlich besseres Ergebnis liefert. So ist die Segmentierung des synthetischen Bildes in zwei Klassen mit dem AFCM-Algorithmus korrekt, während der Isodata-Algorithmus beispielsweise Teile der äußeren Schale als Hintergrund klassifiziert. Außerdem ist bei den Kernspinbildern zu sehen, daß der AFCM-Algorithmus auch die gesamte Kopfhaut als Vordergrund segmentiert. Dagegen ordnet der Isodata-Algorithmus Teile der Kopfhaut dem Knochen bzw. dem Hintergrund zu.
Der AFCM-Algorithmus in der implementierten Form hat aber auch einen schwerwiegenden Nachteil. Eine Segmentierung braucht selbst mit der Parallelisierung bis zu vier Stunden. Der Isodata-Algorithmus benötigt dagegen nur Bruchteile einer Sekunde. Dieser Unterschied sollte aber durch die Verwendung eines Multigrid-Lösers deutlich verringert werden können.

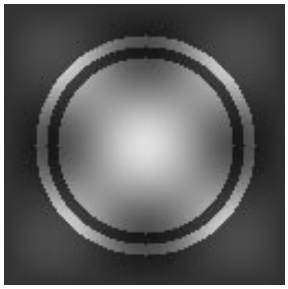
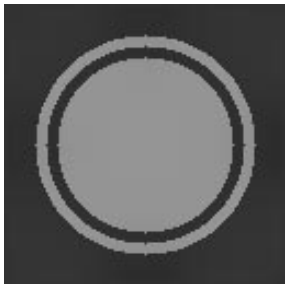
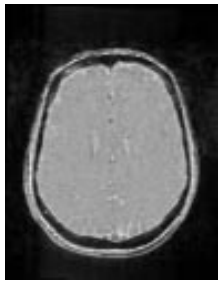
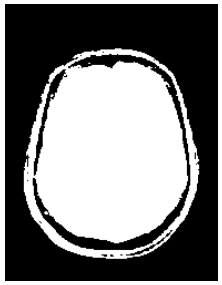
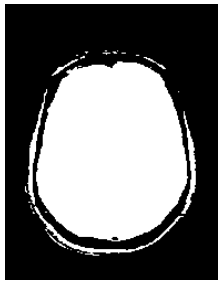
	synth. Bild	PD-gewichtet	T_1-gewichtet
Original			
korrigiertes Bild			
Segmentierung mit AFCM			
Segmentierung mit ISODATA			

Abbildung 4.14: Vergleich der Segmentierungen mit dem Isodata- und dem AFCM-Algorithmus

4.4 Segmentierung mit dem AFCM-Algorithmus

4.4.1 Segmentierung aus dem PD-gewichteten Kernspinbild

Wie der letzte Abschnitt gezeigt hat, wird mit dem AFCM-Algorithmus eine verbesserte Segmentierung im Vergleich zum Isodata-Algorithmus erreicht. Als erstes wird die Auswirkung auf die Segmentierung des Knochens aus dem PD-gewichteten Bild bei der Verwendung des AFCM-Algorithmus untersucht. Das Vorgehen orientiert sich an der Segmentierung mit dem Isodata-Algorithmus. Die Klassifizierung der Voxel wird allerdings mit Hilfe des AFCM-Algorithmus durchgeführt, wobei das Bild wieder in zwei Klassen unterteilt und jedes Voxel einer Klasse entsprechend der maximum-membership-Segmentierung zugewiesen wird. Dazu wird bei dem in Abbildung 4.3 beschriebenen Vorgehen

$$\text{isodata}(b_{PD}, 2)$$

durch

$$\text{afcm}(b_{PD}, 2)$$

ersetzt.
Zuerst erfolgt wieder die Erstellung einer Kopfmaske. Das Ergebnis für verschiedene Werte von d_1 ist in Abbildung 4.15 dargestellt. Im Vergleich zur Segmentierung mit dem Isodata-Algorithmus liefert der AFCM-Algorithmus bessere Ergebnisse. So umfassen die Kopfmasken schon für $d_1 = 4$ den gesamten Kopf, wie auch die Überlagerung des Randes der Masken mit den originalen Bildern in Abbildung 4.17 zeigt. Schließlich ist in Abbildung 4.16 der gesamte Segmentierungsvorgang dargestellt. Für die Parameter d_2 und d_3 wurden wieder die Werte $d_2 = 7$ und $d_3 = 6$ verwendet. Ein Vergleich des segmentierten Knochens, einmal mit einer Segmentierung auf Basis des Isodata-Algorithmus und einmal auf Basis des AFCM-Algorithmus zeigt Abbildung 4.18.
Zu erkennen ist, daß mit dem AFCM-Algorithmus eine bessere Segmentierung des Knochens möglich ist. Noch vorhandene Fehler, wie beispielsweise Löcher in den Knochen, können beispielsweise durch eine Veränderung der Werte d_2 und d_3 verkleinert werden. Diese Möglichkeit ist nicht untersucht worden, denn der Ansatz hat zwei Nachteile:

1. Die Segmentierung des Knochens erfolgt nur in einem kleinen Bereich um das Gehirn. Der Gesichtsschädel wird nicht segmentiert. Dadurch treten beispielsweise schon oberhalb der Augen Fehler auf. Dort wird der vorhandene Knochen unvollständig segmentiert.

2. Die Segmentierung erfolgt ausschließlich aus dem PD-gewichteten Kernspinbild. Dadurch kann es vorkommen, daß die äußere Kante des segmentierten Knochens an manchen Stellen durch kleinere Fehler in den Bildern nicht mit der äußeren Kante im T_1-gewichteten Bild übereinstimmt.

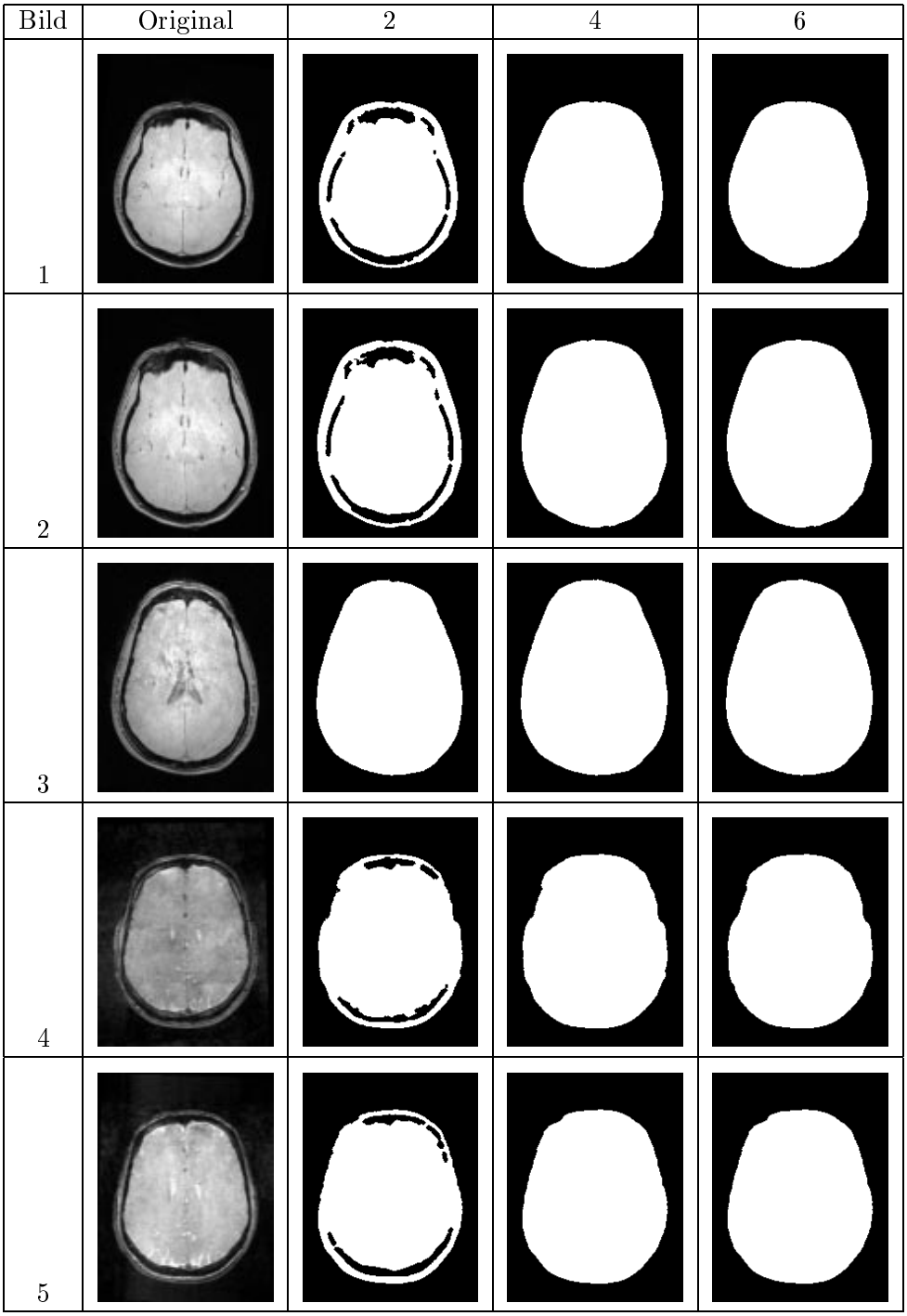


Abbildung 4.15: Auf Basis des AFCM-Algorithmus segmentierte Kopfmasken für unterschiedliche Werte des Distanzparameters

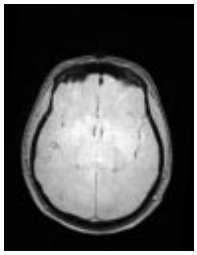
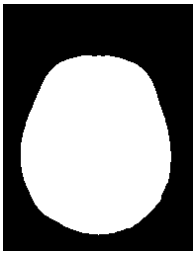
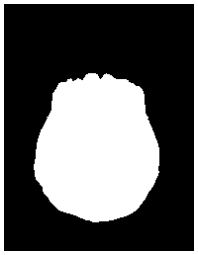
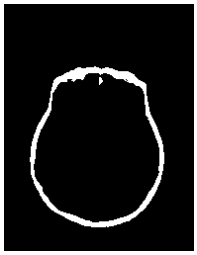
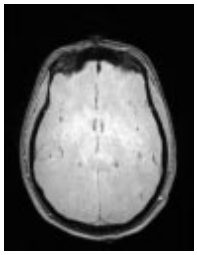
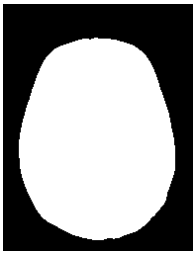
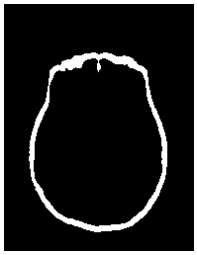
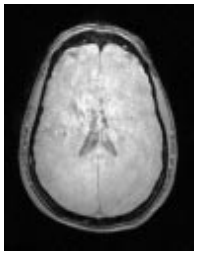
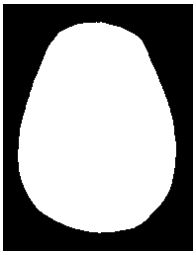
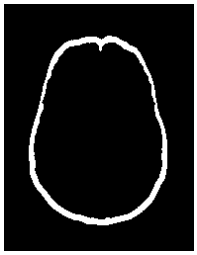
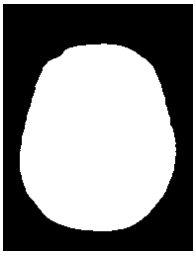
Bild	Original	Kopfmaske	Liquormaske	Knochen
1				
2				
3				
4				
5				

Abbildung 4.16: Komplette Segmentierung eines PD-gewichteten Kernspinbildes mit einem AFCM-basierten Algorithmus für fünf verschiedene Bilder, axiale Schnitte

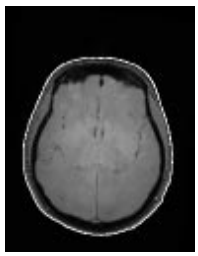
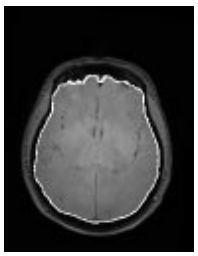
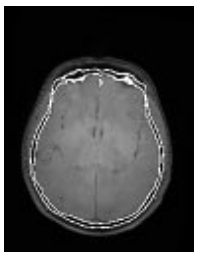
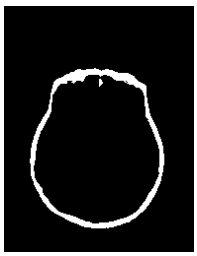
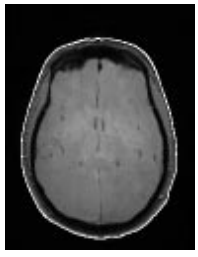
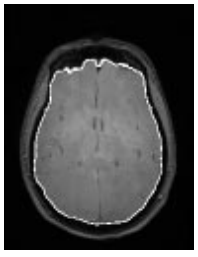
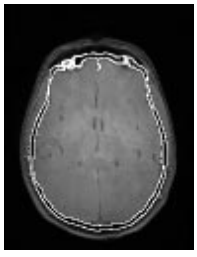
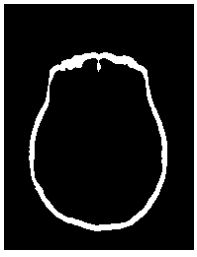
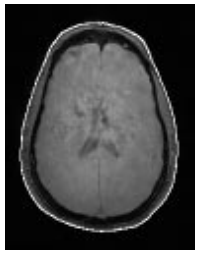
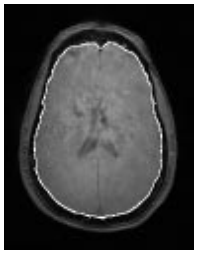
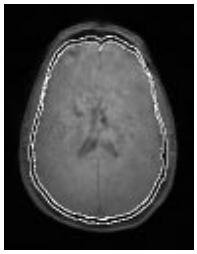
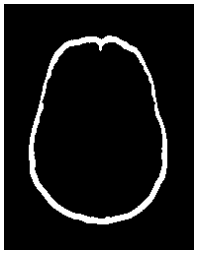
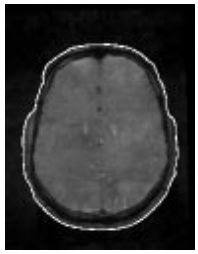
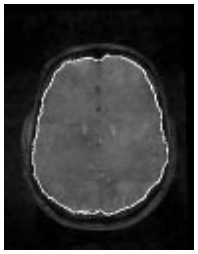
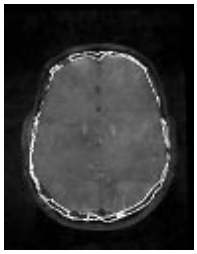
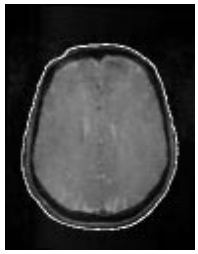
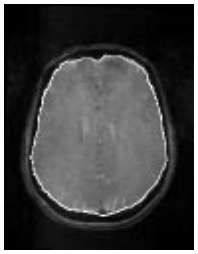
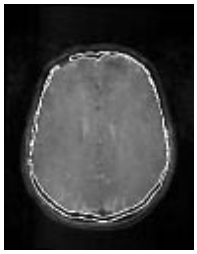
Bild	Überlagerung des Originalbildes mit dem Rand der Kopfmaske	Liquormaske	Knochenmaske	Knochenmaske
1				
2				
3				
4				
5				

Abbildung 4.17: Einblendung des Randes der Kopf-, Liquor- und Knochenmaske in das Kernspinbild für fünf verschiedene Bilder, jeweils ein axialer Schnitt

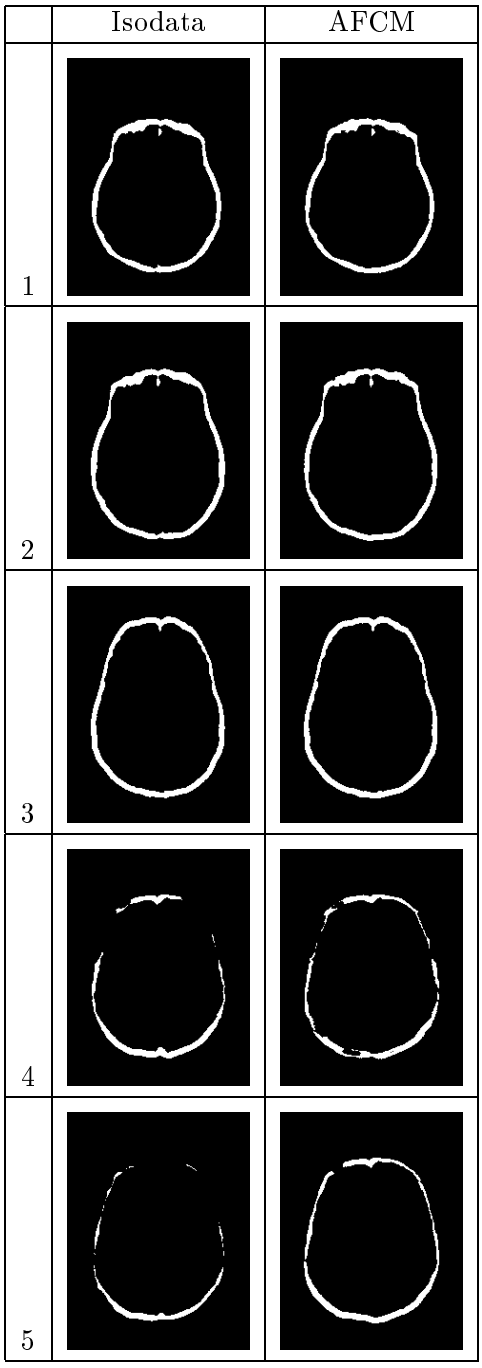


Abbildung 4.18: Gegenüberstellung der Ergebnisse der Knochensegmentierung unter Verwendung des Isodata- und des AFCM-Algorithmus mit konstanten Werten von d_2 und d_3

4.4.2 Segmentierung aus dem T_1- und PD-gewichteten Kernspinbild

Nachdem gezeigt wurde, daß sich durch die Verwendung des AFCM-Algorithmus eine bessere Segmentierung des PD-gewichteten Bildes erreichen läßt, wird jetzt ein erster Ansatz zur Segmentierung des Knochens unter Verwendung beider Bilder vorgestellt.
Das Vorgehen gliedert sich in die folgenden Schritte:

1. Erstellung der Kopfmaske,
2. Erstellung der Liquormaske,
3. Selektion aller Voxel, die innerhalb der Kopfmaske und außerhalb der Liquormaske liegen und zusätzlich bestimmte Kriterien erfüllen.

Für die Kopfmaske werden zunächst getrennte Masken aus dem T_1- und dem PD-gewichteten Bild erstellt. Für das PD-gewichtete Bild wird dazu das bereits vorgestellte Verfahren unter Ausnutzung des AFCM-Algorithmus verwendet. Um eine Kopfmaske aus dem T_1-gewichteten Bild zu erstellen, wird dieses Vorgehen leicht abgeändert. Im ersten Schritt erfolgt die Auswahl aller Voxel, die entweder zu Klasse 1 oder zu Klasse 2 zugeordnet sind, d.h. das erste Teilergebnis ergibt sich als

$$z_1 = \text{binarize}(\text{afcm}(b_{T_1}, 3), 1, 2) \ .$$

Das weitere Vorgehen ist identisch.
Die beiden entstehenden Masken werden mittels einer *und*-Operation überlagert. Auf diese Weise wird die Auswirkung des Rauschen in den Bildern kompensiert. Das Ergebnis ist die Kopfmaske. Von dieser wird eine modifizierte Maske erstellt, indem alle Voxel entfernt werden, die einen Abstand kleiner als eine Distanz d vom Rand haben. Dadurch ist eine bessere Trennung zwischen Hintergrund und Vordergrund möglich.
Die Erstellung der Liquormaske erfolgt aus dem PD-gewichteten Bild, ebenfalls durch das bereits vorgestellte Vorgehen. Durch die modifizierte Kopf- und die Liquormaske ist ein Bereich eingegrenzt, in dem die Voxel auf Zugehörigkeit zum Knochen untersucht werden. Als nächster Schritt erfolgt im T_1-gewichteten Bild die Auswahl aller Voxel innerhalb dieses Bereiches, die zur Klasse mit dem Knochen gehören. Das Ergebnis ist die Knochenmaske.
Für diese Auswahl wurden zwei Kriterien getestet. Zuerst wurden die Voxel gemäß der maximum-membership-Segmentierung ausgewählt. Das Segmentierungsergebnis ist in Abbildung 4.19 für $d = 2$ und $d = 4$ dargestellt. Es ist zu erkennen, daß der Knochen unvollständig segmentiert ist und Löcher in der Maske vorhanden sind.
Deshalb wurden als nächstes alle Voxel ausgewählt, deren Zugehörigkeitswert in der Klasse für den Knochen mindestens 0,1 ist. Die entstehenden Segmentierungsergebnisse sind in Abbildung 4.20 für $d = 2$ und $d = 4$ dargestellt. Es fällt auf, daß das Kriterium für die Zugehörigskeitswerte dazu führt, daß Teile der Kopfhaut als Knochen segmentiert werden. So ist beispielsweise am Hinterkopf

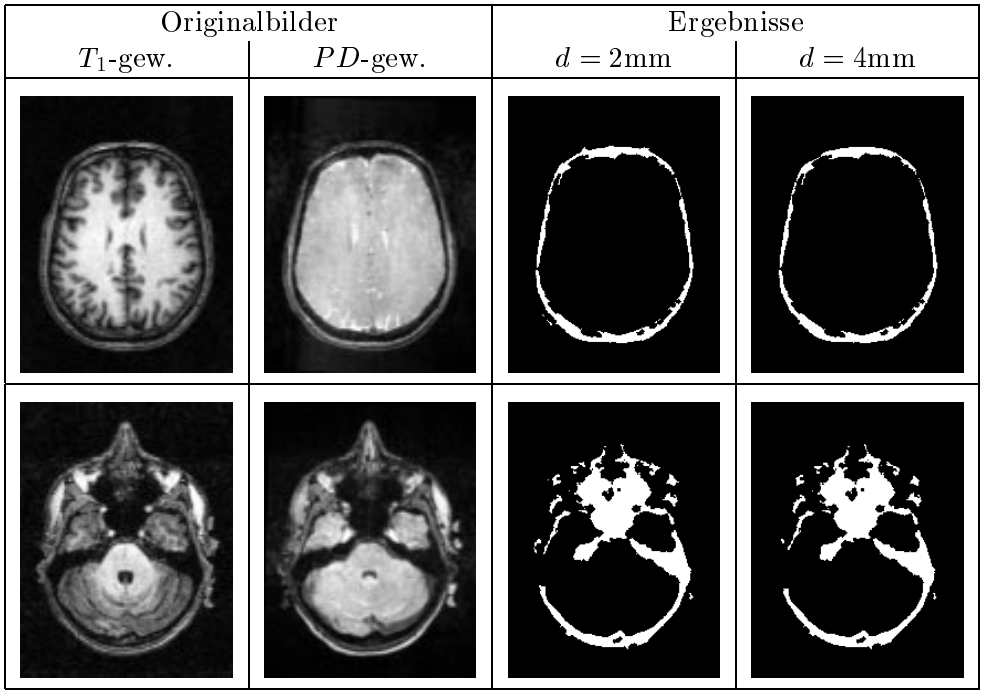


Abbildung 4.19: Segmentierung des Knochens aus einem T_1 und einem PD-gewichteten Kernspinbild, Zuordnung aller Voxel im T_1-gew. Bild zu einer Klasse mittels maximum-membership-Segmentierung: Originalbilder und Resultate, zwei axiale Schnitte

der Knochen zu dick. Würde man den Parameter d erhöhen, so kann dieser Effekt kompensiert werden. Allerdings führt das an anderen Stellen dazu, daß Teile des Knochens nicht segmentiert werden. Ansonsten ist aber an einigen Stellen im Bild der Knochen noch immer fälschlicherweise offen.

Zusammenfassend läßt sich feststellen, daß das vorgestellte Verfahren, um aus einem T_1- und einem PD-gewichteten Kernspinbild die Knochenmaske zu erstellen, teilweise zu größeren Fehlern bei der Segmentierung führt. Der Knochen ist nicht geschlossen. Es sind Stellen offen, an denen sich Knochen befindet. Außerdem ist die Trennung zwischen Kopfhaut und Knochen schwierig und nicht vollständig durchführbar. Allerdings läßt sich aus den Segmentierergebnissen intuitiv, trotz offener Stellen, die innere und äußere Kante des Knochens ablesen. Daher erscheint es sinnvoll, diese Kanten durch ein elastisches Modell zu bestimmen. Dafür ist allerdings ein initialer Vorschlag für diese notwendig. Dazu können die bislang gewonnenen Segmentierergebnisse verwendet werden.

4.5 Elastische Modelle

Der letzte Abschnitt hat gezeigt, daß für die Segmentierung die Verwendung eines elastischen Modells [19, 35, 43, 44] sinnvoll erscheint. Diese Modelle haben den Vorteil, daß sie einen initialen Vorschlag für eine Kante selbständig anpassen können. Sind die Parameter für das Modell einmal für eine Klasse von Bildern

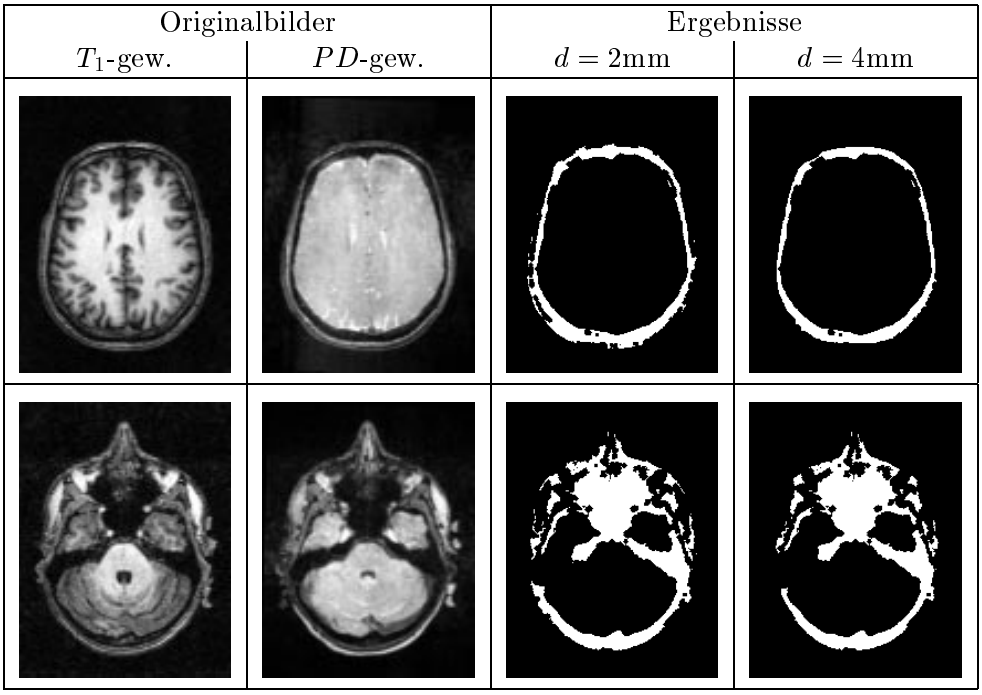


Abbildung 4.20: Segmentierung des Knochens aus einem T_1 und einem PD-gewichteten Kernspinbild, Zuordnung aller Voxel im T_1-gew. Bild zur Klasse mit Knochen, deren Zugehörigkeitswert in dieser Klasse mindestens 0,1 ist: Originalbilder und Resultate, zwei axiale Schnitte

eingestellt, kann es auf weitere Bilder dieser Klasse ohne Eingriff des Anwenders angewendet werden.
Der Knochen wird durch die innere und äußere Kante beschrieben. Durch das elastische Modell können diese Kanten mit einer Subvoxel-Genauigkeit bestimmt werden. Solche Modelle sind auch in bestimmten Grenzen nicht anfällig gegenüber Störungen in den Bildern, die bislang zu offenen Stellen geführt haben.
Das Vorgehen für die Segmentierung mit dem elastischen Modell unterteilt sich in folgende Schritte:

1. Bestimmung einer initialen Segmentierung unter Verwendung des AFCM-Algorithmus,

2. Extraktion der Oberflächen der initialen Segmentierungen als Dreiecksnetze und eventuelles Vereinfachen dieser Netze und

3. Anpassen dieser Netze an innere und äußere Kante des Knochens mit Hilfe des elastischen Modells.

Das Finden der initialen Segmentierung ist ein wichtiger Punkt für den Erfolg der Segmentierung. Nach [35] ist es erforderlich, daß sich der initiale Vorschlag in der Nähe der Kante befindet. Auf die Bestimmung dieser initialen Kanten wird später eingegangen.

4.5.1 Extraktion der Oberflächen

Nachdem eine initiale Segmentierung durchgeführt ist, wird die Oberfläche des Ergebnisses als Dreiecksnetz extrahiert. Dazu wird ein in [9, 27] vorgestelltes Verfahren zur Isoflächen-Extraktion verwendet.
Sei $b = (\mathcal{T}, \mathcal{G}, I)$ ein Bild aus dem die Oberfläche extrahiert werden soll, die entlang des Grauwertes I_0 verläuft. Jeweils acht benachbarte Punkte $(x, y, z)^T$, $(x+1, y, z)^T$, $(x, y+1, z)^T$, $(x+1, y+1, z)^T$, $(x, y, z+1)^T$, $(x+1, y, z+1)^T$, $(x, y+1, z+1)^T$, $(x+1, y+1, z+1)^T$ aus $\mathcal{T}$ bilden einen Würfel. Für einen Würfel sind zwei Zerlegungen in jeweils fünf Tetraeder möglich [27], die spiegelsymmetrisch zur $y, z - Ebene$ sind. Auf diese Weise werden die Würfel in dem Bild so angeordnet, daß zwei Würfel mit gemeinsamer Fläche jeweils spiegelsymmetrisch zueinander sind. Jedes Tetraeder läßt sich durch seine vier Eckpunkte $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$ beschreiben.
Der nächste Schritt besteht darin, zu bestimmen, ob ein Teil der Oberfläche den Tetraeder $(\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$ schneidet. Dazu wird an den vier Punkten $\mathbf{v}_i$ der Wert $I_i = I(\mathbf{v}_i) - I_0$ berechnet. Haben diese Werte unterschiedliche Vorzeichen, verläuft die Oberfläche durch diesen Tetraeder. Es müssen die Schnittpunkte mit den Kanten bestimmt werden.
Jede Kante, deren Endpunkte unterschiedliche Vorzeichen aufweisen, liefert einen Knoten für das Dreiecksnetz. Dessen exakte Position wird durch eine bilineare Interpolation bestimmt[3]. Seien (a, b, c, d) die vier Werte an den Punkten des Tetraeders, dann ist der Wert entlang der Kante (a,d) durch

$$I(u) = (a + d - b - c)u^2 + (-2a + b + c)u + a \tag{4.45}$$

bestimmt, wobei $u \in [0, 1]$. Wenn $ad < 0$ ist, das heißt, a und d haben verschiedene Vorzeichen, dann existiert genau eine Nullstelle von $I(u)$ im Bereich $0 < u < 1$. An dieser Stelle befindet sich der Schnittpunkt. Auf diese Weise liefert jedes Tetraeder, das sich mit der Oberfläche schneidet, ein Dreieck für das zu erstellende Netz.
Die Extraktion der Oberflächen erfolgt aus Masken, das heißt Bildern mit einer binären Grauwertmenge $\{0, 1\}$. Deshalb wird im weiteren stets $I_0 = 0,9$ gesetzt.

4.5.2 Vereinfachung der Dreiecksnetze

Nach der Extraktion der Oberflächen schließt sich eine Vereinfachung der erzeugten Dreiecksnetze an, die ebenfalls nach einem Algorithmus aus [9] erfolgt. Für eine Vereinfachung werden die einzelnen Kanten nacheinander überprüft. Sei $(\mathbf{v}_1, \mathbf{v}_2)$ die zu überprüfende Kante, und seien $\mathbf{v}_3$ und $\mathbf{v}_4$ der jeweils dritte Knoten der Dreiecke mit gemeinsamer Kante $(\mathbf{v}_1, \mathbf{v}_2)$. Um die Kante $(\mathbf{v}_1, \mathbf{v}_2)$ zu entfernen, muß der Abstand zwischen $\mathbf{v}_1$ und $\mathbf{v}_2$ kleiner sein als der Abstand zwischen $\mathbf{v}_3$ und $\mathbf{v}_4$. Weiterhin werden die Abstände q_1 und q_2 zwischen $\mathbf{v}_1$ und $\mathbf{v}_2$ und der neuen Oberfläche berechnet, die entsteht, wenn $\mathbf{v}_1$ und $\mathbf{v}_2$ durch $\mathbf{v}$ mit $\mathbf{v} = \frac{1}{2}(\mathbf{v}_1 + \mathbf{v}_2)$ ersetzt würden. Sind diese Abstände kleiner als ein Wert

[3] Nach [9] führt eine lineare Interpolation zu einem schlechten Ergebnis

t, erfolgt die Ersetzung. Der Test, ob zwei Knoten durch einen ersetzt werden können, wird für jede Kante durchgeführt. Im folgenden wird $t = 2$ verwendet. Die ganze Vereinfachung wird solange durchgeführt, bis das Ergebnis ein Dreiecksnetz mit der gewünschten Anzahl von Knoten ist.

4.5.3 Anpassung der Dreiecksnetze

Nach obiger Vereinfachung erfolgt die Anpassung der Netze. Dazu werden die mit Hilfe des AFCM-Algorithmus erstellen, intensitätskorrigierten Kernspinbilder verwendet. Zum Anpassen eines Netzes an eine Kante im Bild $b = (\mathcal{T}, \mathcal{G}, I)$ wird das Modell nach [19] benutzt.
Auf jeden Knoten $\mathbf{v}_0$ im Netz wirken interne und externe Kräfte. Die interne Kraft versucht, den Knoten zwischen seinen mit Kanten verbundenen Nachbarn $\mathbf{v}_i, i = 1, \ldots, N$ zu zentrieren:

$$\mathbf{F}_{int} = \frac{1}{N} \sum_{i=1}^{N} (\mathbf{v}_i - \mathbf{v}_0) \tag{4.46}$$

Die externe Kraft verschiebt die Knoten entlang ihrer Oberflächennormalen $\mathbf{n}_0$. Sie ergibt sich aus dem Bild b und setzt sich aus zwei Teilen zusammen. Der erste Teil $F_{ext,1}$ ergibt sich aus einem Gradientenfeld, das aus dem mit einem Gaußfilter gefilterten Bild

$$b_g = \text{gauss}(b, \sigma) = (\mathcal{T}, \mathcal{G}, I_g) \tag{4.47}$$

erzeugt wird. Das führt auf:

$$F_{ext,1} = \langle \nabla I_g(\mathbf{v}_0), \mathbf{n}_0 \rangle \; . \tag{4.48}$$

Der zweite Teil $F_{ext,2}$ der externen Kraft fixiert die Oberfläche in einem kleinen Bereich um einen Grauwert I_{lim}. Er ergibt sich als:

$$F_{ext,2} = \tanh\left(\kappa(I(\mathbf{v}_0) - I_{lim})\right) \; . \tag{4.49}$$

Mit Hilfe von κ kann die Unempfindlichkeit gegenüber dem Rauschen im Bild gesteuert werden. Es wird $\kappa = 3$ verwendet. Ausgehend von dem initialen Netz erfolgt eine Anpassung an die Oberfläche durch

$$\mathbf{v}_0(t+1) = \mathbf{v}_0(t) + \omega_{int}\mathbf{F}_{int} + \omega_{ext}\left(F_{ext,1}F_{ext,2} + F_{ext,2}\right)\mathbf{n}_0 \; . \tag{4.50}$$

in mehreren Iterationsschritten. Der Teil $F_{ext,1}$ der externen Kraft, der sich aus dem Gradienten ergibt, wird durch $F_{ext,2}$ gewichtet, um die wirkende Kraft zu reduzieren, wenn ein bestimmter Grauwertbereich erreicht ist. Die Oberflächennormalen sind so gerichtet, daß sie aus dem vom Netz umschlossenen Bereich

herauszeigen. Durch diese Richtung der Oberflächennormalen bewegt sich bei $\omega_{ext} > 0$ ein Knoten $\mathbf{v}_0$ in den Kopf hinein, wenn $I(\mathbf{v}_0) < I_{lim}$ ist. Ist dagegen $\omega_{ext} < 0$, dann bewegt sich bei $I(\mathbf{v}_0) < I_{lim}$ der Knoten $\mathbf{v}_0$ aus dem Kopf heraus. Die Konvergenzeigenschaften lassen sich durch die Wichtung der Kräfte mittels ω_{int} und ω_{ext} steuern.

4.6 Implementierung

In dieser Arbeit wird die innere Kante im PD-gewichteten Bild angepaßt, die äußere Kante im T_1-gewichteten. Zur Bestimmung des Wertes von I_{lim} werden, ähnlich wie in [19], die Klassenzentren aus dem AFCM-Algorithmus verwendet. Die Segmentierung des PD-gewichteten Bildes führt auf zwei Klassenzentren v_1 und v_2. Für dieses Bild wird I_{lim} als

$$I_{lim} = \frac{v_1 + v_2}{2} \tag{4.51}$$

gesetzt. Für das T_1-gewichtete Bild existieren drei Klassenzentren $v_1 < v_2 < v_3$. Der Wert von I_{lim} wird ebenfalls nach Gleichung 4.51 bestimmt.
Für eine Verwendung des beschriebenen elastischen Modells muß eine initiale Segmentierung für die innere und äußere Kante des Knochens generiert werden. Wie im vorletzten Abschnitt zu sehen ist, läßt sich die Kante zwischen Knochen und Gehirnflüssigkeit bei Verwendung des AFCM-Algorithmus einfach segmentieren. Das dort vorgestellte Verfahren zur Erstellung der Liquormaske wird für die initiale Segmentierung verwendet. Schwieriger gestaltet sich der initiale Vorschlag für die äußere Kante.
Da eine Segmentierung dieser Kante mit Hilfe des AFCM-Algorithmus sehr schwierig ist, wird versucht, diese aus der initialen inneren Kante zu gewinnen. Dazu wird auf der Liquormaske eine Distanztransformation durchgeführt und alle Voxel mit einem Abstand bis zu 7mm selektiert. Die so entstehende Maske repräsentiert die verwendete initiale äußere Kante. In Abbildung 4.21 sind die so entstehenden initialen Kanten in die originalen Bilder eingezeichnet.
Ausgehend von diesen initialen Kanten erfolgt die Anpassung unter Verwendung des vorgestellten elastischen Modells. Die angepaßten Kanten finden sich in Abbildung 4.22. Die Parameter für das elastische Modell sind $\omega_{int} = 0,05$ und $\omega_{ext} = 0,01$ für die innere und $\omega_{ext} = -0,02$ für die äußere Kante.
Aus Abbildung 4.22 ist zu erkennen, daß die innere Kante korrekt an die vorhandene innere Kante angepaßt ist. Bei der äußeren Kante dagegen gibt es noch Schwierigkeiten. Sie verläuft teilweise innerhalb des Knochens, wo das elastische Modell die flüssigkeitsreiche Spongiosa als Kante zwischen Knochen und Kopfhaut ansieht, die in den Kernspinbildern ähnliche Intensitäten wie die Kopfhaut aufweist. In einem weiteren axialen Schnitt (unten) ist außerdem zu erkennen, daß die Kante durch den Schädelknochen verläuft. Dadurch befinden sich Teile des Knochens nicht zwischen den angepaßten Kanten. Diese Probleme sind auf eine schlecht gewählte initiale Kante zurückzuführen. Daher wird diese als nächstes so gewählt, daß sie vollständig außerhalb des Knochens verläuft.
Bei der Auswahl einer entsprechenden äußeren Kante ist eine Eigenschaft des verwendeten elastischen Modells zu berücksichtigen. Durch die Wahl von $\omega_{ext} <$

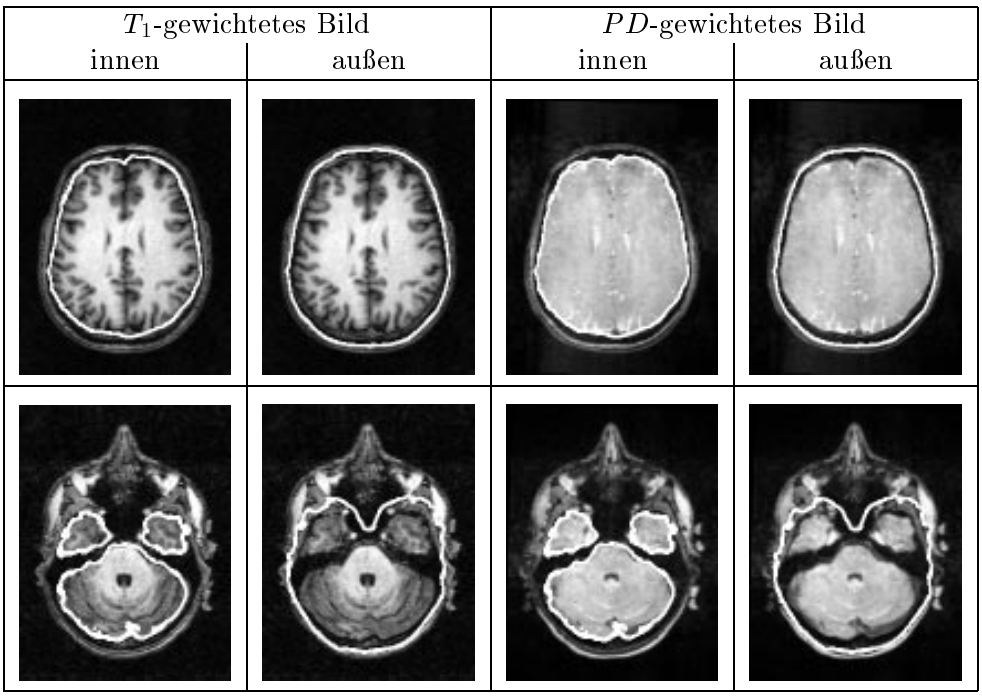


Abbildung 4.21: Initiale äußere und innere Kante des Knochens in einem Bild, eingeblendet jeweils in das T_1- und PD-gewichtete Bild, zwei axiale Schnitte

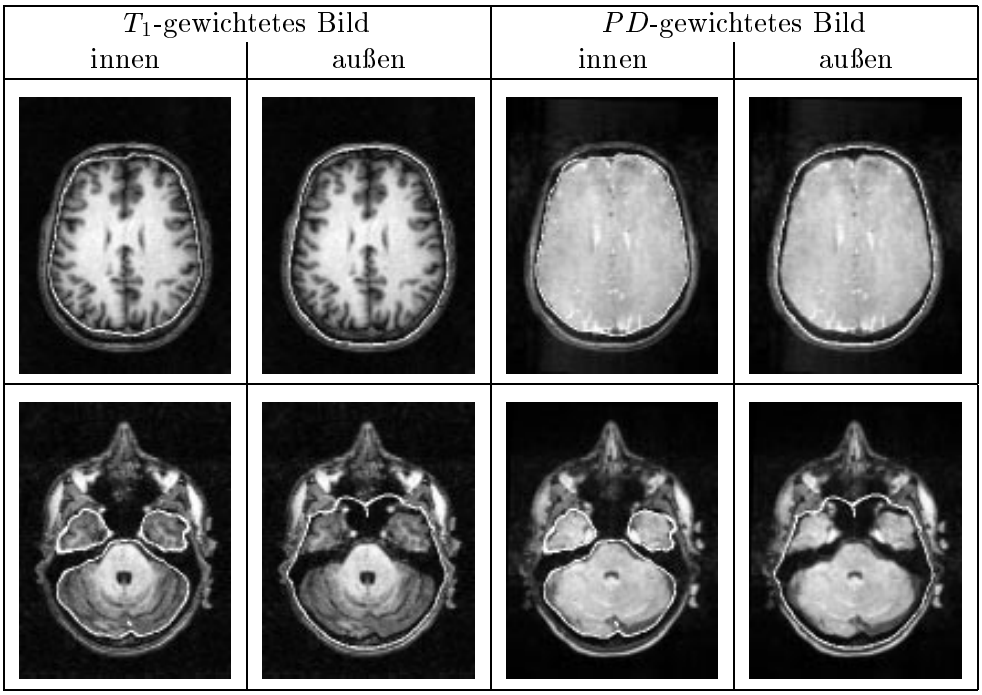


Abbildung 4.22: Angepaßte äußere und innere Kante des Knochens zu den initialen Kanten aus Abbildung 4.21 in einem Bild, eingeblendet jeweils in das T_1- und PD-gewichtete Bild, zwei axiale Schnitte

0 wird erreicht, daß ein Knoten, der sich in einem dunklen Bereich befindet, nach außen in Richtung der Kopfhaut bewegt wird. Befindet er sich in einem hellen Bereich, so wird angenommen, er sei in der Kopfhaut, und er wird nach innen bewegt. Befindet er sich dagegen außerhalb des Kopfes, so ist er auch in einem dunklen Bereich des Bildes und wird deshalb weiter nach außen bewegt. Unter Berücksichtigung dieses Verhaltens muß die initiale äußere Kante gewählt werden.
Die Wahl dieser initialen Kante erfolgt mit Hilfe der Kopfmaske. Entsprechend dem Vorgehen in Kapitel 4.4.2 werden wieder separat aus dem T_1- und dem PD-gewichteten Bild die Masken erstellt. Die fertige Kopfmaske ergibt sich durch eine Überlagerung mittels und-Operation. Aus der so entstehenden Kopfmaske wird die initiale Kante generiert. Um ein Abdriften eines Knotens in Bereiche außerhalb des Kopfes zu vermeiden, wird das T_1-gewichtete Bild modifiziert und die Anpassung in diesem modifizierten Bild vorgenommen. Die Modifikation erfolgt in folgenden Schritten (Abbildung 4.23):

1. Invertierung der Kopfmaske,
2. Distanztransformation und Selektion aller Voxel mit einem Abstand bis 7mm,
3. Überlagerung mit dem T_1-gewichteten Bild.

So wird erreicht, daß der Grauwert von Voxeln, die nach der Kopfmaske zum Hintergrund gehören, auf den maximalen Wert gesetzt wird. Das führt dazu, daß sich der Hintergrund im Bild wie die Kopfhaut verhält. Um kleinere Fehler in der Maske zu beheben, erfolgt eine Verkleinerung der Maske in den Schritten eins und zwei.
Zuletzt müssen noch die notwendige Anzahl von Knoten in den Netzen sowie die erforderlichen Parameter ω_{int} und ω_{ext} bestimmt werden. Für die Bestimmung der Knotenanzahl sind folgende Bedingungen zu berücksichtigen:

- Das Netz sollte möglichst wenig Knoten enthalten. Dadurch wird der Anpassungsprozess beschleunigt.
- Das Netz sollte aber so viele Knoten enthalten, daß eine gute Anpassung an die Kante erfolgt.

Aus diesem Grund stellten sich für beide Dreiecksnetze 30000 Knoten als sinnvoll heraus. Als Parametereinstellung werden $\omega_{int} = 0,06$ und $\omega_{ext} = 0,01$ für die Anpassung an die innere Kante und $\omega_{int} = 0,05$ und $\omega_{ext} = -0,01$ für die Anpassung an die äußere Kante verwendet.
Eine Segmentierung mit diesen Parametern ist in den Abbildungen 4.24 (initiale Kanten) und 4.25 (angepaßte Kanten) dargestellt. Dort ist zu sehen, daß in diesem Bild die Dreiecksnetze korrekt an die vorhandenen Kanten angepaßt worden sind. Das Netz für die innere verläuft auf der Kante zwischen Knochen und Gehirnflüssigkeit, das Netz für die äußere auf der Kante zwischen Knochen und Kopfhaut. Dieser Datensatz ist korrekt segmentiert worden. Deshalb wird dieses Verfahren jetzt auf alle fünf vorhandenen Datensätze angewendet.

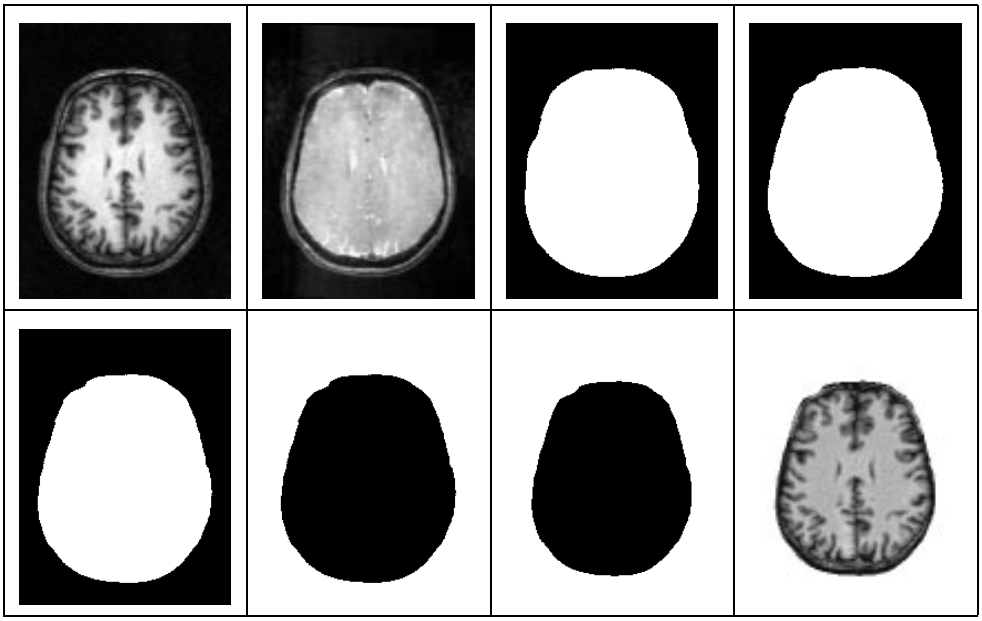


Abbildung 4.23: Erstellung des modifizierten T_1-gewichteten Bildes (axialer Schnitt), von links nach rechts: oben: T_1- und PD-gew. Bild, Kopfmaske aus dem T_1- und PD-gewichteten Bild, unten: Kopfmaske, invertierte Kopfmaske, nach Distanztransformation, modifiziertes Bild

4.7 Bewertung

In diesem Abschnitt wird der zuletzt untersuchte Segmentierungsalgorithmus angewendet, um damit die fünf registrierten Datensätze (Abbildung 3.5) zu segmentieren. Dieser sieht das folgende Vorgehen für eine Segmentierung vor:

1. Erstellung der Liquormaske und der Kopfmaske mit dem in Kapitel 4.4.2 vorgestellten Verfahren,

2. Extraktion der Oberflächen aus den Masken und Vereinfachen der Netze auf 30000 Knoten mit den in Kapitel 4.5 vorgestellten Algorithmen und

3. Anpassung der Dreiecksnetze an die Kanten. Die Anpassung der inneren Kante erfolgt im PD-gewichteten Kernspinbild, die Anpassung der äußeren im T_1-gewichteten. Es werden die Bilder mit korrigierten Intensitäten verwendet, die ein Ergebnis des AFCM-Algorithmus sind. Die Parameter für das elastische Modell sind: $\omega_{int} = 0,06$, $\omega_{ext} = 0,01$ für die Anpassung an die innere Kante und $\omega_{int} = 0,05$, $\omega_{ext} = -0,01$ für die Anpassung an die äußere. I_{lim} wird gemäß Gleichung 4.51 gesetzt.

Das Ergebnis der Segmentierung ist in den Abbildungen 4.26 bis 4.30 dargestellt. Es sind jeweils die innere und äußere Kante des Knochens in das T_1- und das PD-gewichtete Bild eingeblendet. Zusätzlich dazu findet sich die Knochenmaske, die durch Auffüllen des Bereiches zwischen äußerer und innerer Kante entsteht. Dargestellt sind jeweils zwei axiale Schnitte, ein coronarer und ein sagitaler Schnitt.

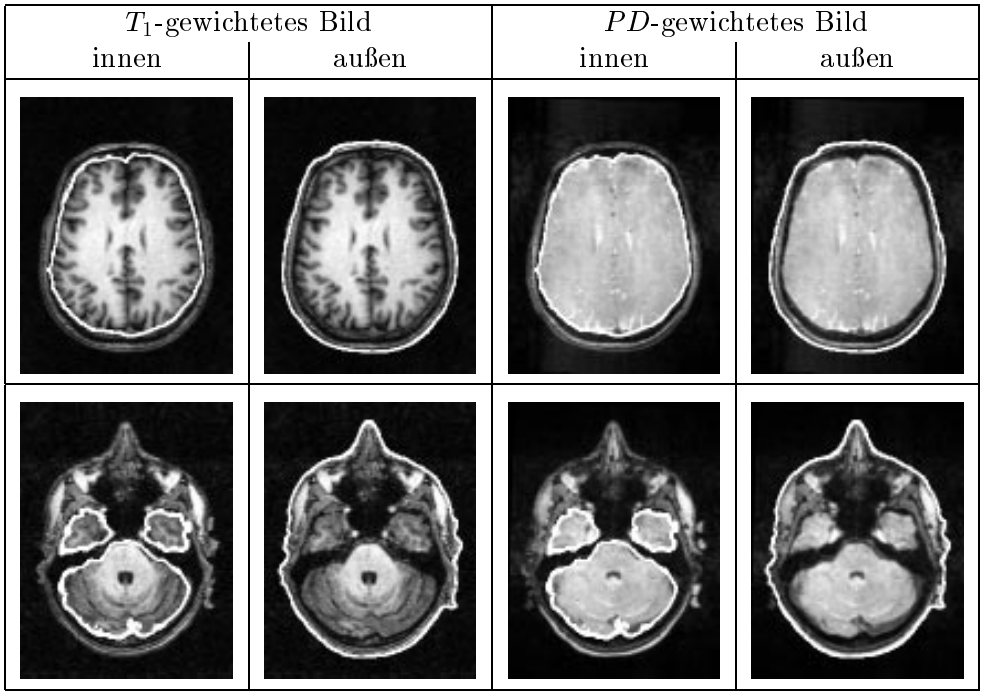


Abbildung 4.24: Verbesserte initiale äußere und initiale innere Kante des Knochens in einem Bild, eingeblendet jeweils in das T_1- und PD-gewichtete Bild, zwei axiale Schnitte

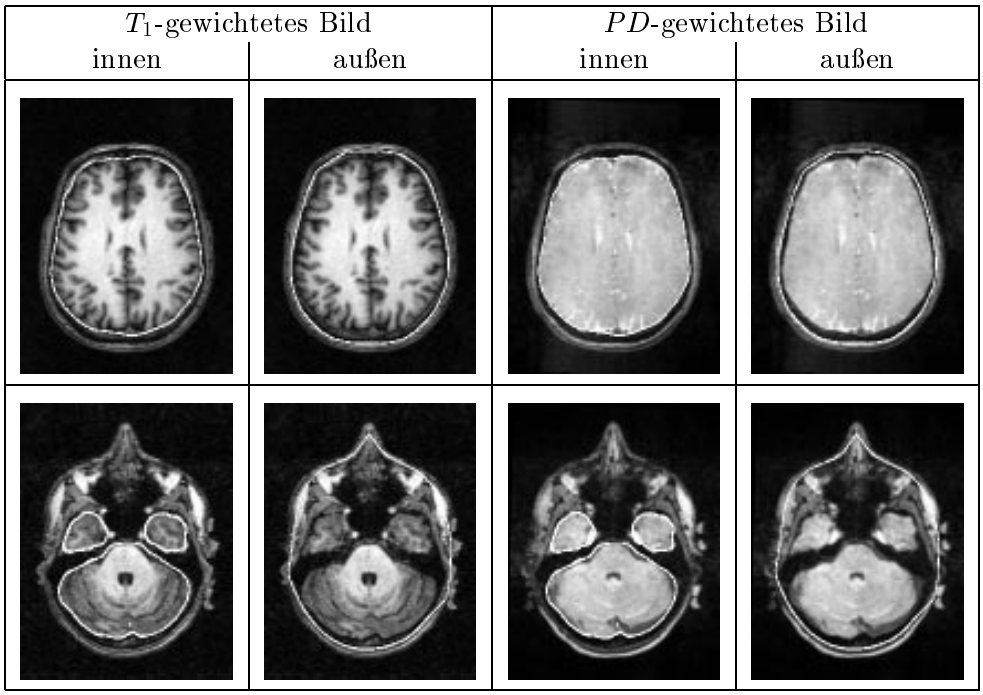


Abbildung 4.25: Angepaßte äußere und innere Kante des Knochens zu den initialen Kanten aus Abbildung 4.24 in einem Bild, eingeblendet jeweils in das T_1- und PD-gewichtete Bild, zwei axiale Schnitte

Zur Qualität der Segmentierung läßt sich folgendes feststellen:

- Im Datensatz 1 (Abbildung 4.26) erfolgt eine korrekte Anpassung der inneren und äußeren Kante. Es sind keine Fehler zu erkennen.
- Im Datensatz 2 (Abbildung 4.27) erfolgt ebenfalls eine korrekte Anpassung der Dreiecksnetze an die Kanten. In der sagitalen Schnittdarstellung fällt lediglich am Hirnstamm auf, daß dort das Netz für die innere Kante das Netz für äußere etwas durchdringt. Allerdings ist dieser Bereich des Bildes weit außerhalb der Region, in der später eine korrekte Segmentierung benötigt wird. Im ersten axialen Schnitt im T_1-gewichteten Bild ist zudem ein größerer Bereich mit Hirnflüssigkeit (zwischen Gehirn und innerer Kante des Knochens) zu erkennen. An dieser Stelle tritt eine erhebliche Verbesserung gegenüber dem in Kapitel 1 angesprochenen Schätzverfahren auf.
- Im Datensatz 3 (Abbildung 4.28) werden die Netze ebenfalls korrekt angepaßt. Die massive Störung in dem PD-gewichteten Bild, die in der coronaren Schnittdarstellung oben am Kopf zu erkennen ist, führt lediglich dazu, daß das Netz für die innere Kante ca. ein bis zwei Voxel oberhalb der eigentlichen Kante verläuft.
- In den Schnittdarstellungen von Datensatz 4 (Abbildung 4.29) ist eine korrekte Anpassung der Netze zu erkennen. Auf der zweiten axialen Schnittdarstellung vom PD-gewichteten Bild ist links vom Kopf ein heller Kreis zu erkennen. Dieser beeinflußt das Histogramm des Bildes nachhaltig. Dadurch ist die Segmentierung mit dem Isodata- und auch mit dem AFCM-Algorithmus erschwert und führt zu einem fehlerhaft segmentierten Knochen. Durch das elastische Modell ist dennoch eine korrekte Segmentierung möglich.
- Im Datensatz 5 (Abbildung 4.30) erfolgt die Anpassung der Netze ebenfalls korrekt. Auffallend in der coronaren Schnittdarstellung ist, daß der Knochen oben sehr dünn ist. Das ist allerdings durch den im PD-gewichteten Kernspinbild erkennbaren Bereich mit Gehirnflüssigkeit bedingt und kein Segmentierungsfehler.

In allen verwendeten Datensätzen erfolgt demnach eine korrekte Segmentierung des Knochens.
Abschließend werden in Tabelle 4.1 die für eine Segmentierung benötigten Zeiten dargestellt. Als Engstelle ist dabei der AFCM-Algorithmus zu betrachten, der durch seine Ausführungszeit von ca. vier Stunden pro Bild 90% der Gesamtzeit benötigt und diese damit wesentlich dominiert.
Zusammenfassend läßt sich feststellen, daß der verwendete Ansatz den Knochen aus einem T_1- und einem PD-gewichteten Kernspinbild korrekt segmentiert. Vorteilhaft ist, daß durch die Verwendung dieser beiden Datensätze gegenüber dem bislang gebräuchlichen Verfahren eine genaue Segmentierung der Kante zwischen Knochen und Gehirnflüssigkeit und damit des gesamten Knochens erreicht wird. Als weiterer Vorteil erweist sich dabei, daß das Verfahren

Aktion	Zeit / min.
AFCM-Algorithmus	2×240
Erstellung der Kopfmaske	2×2
Erstellung der initialen Liquormaske aus dem PD-gew. Bild	2
Oberflächen-Extraktion aus den initialen Masken und Vereinfachung des Netzes	2×8
Erstellung des modifizierten T_1-gewichteten Bildes	4
Anpassung der Dreiecksnetze an die Kanten	2×12
Gesamtzeit	530

Tabelle 4.1: Benötigte Zeiten (in Minuten) für die Segmentierung des Knochens aus einem T_1- und einem PD-gewichteten Kernspinbild

bei den vorhandenen Datensätzen die Segmentierung ohne jeglichen manuellen Eingriff durchführt. Nachteilig ist, daß der Knochen als homogen zwischen der inneren und der äußeren Kante betrachtet wird und dadurch beispielsweise die Stirnhöhle als Knochen segmentiert wird. Ein weiterer Nachteil ist der Zeitbedarf von ca. 530 Minuten (8,8 Stunden) für eine komplette Segmentierung.

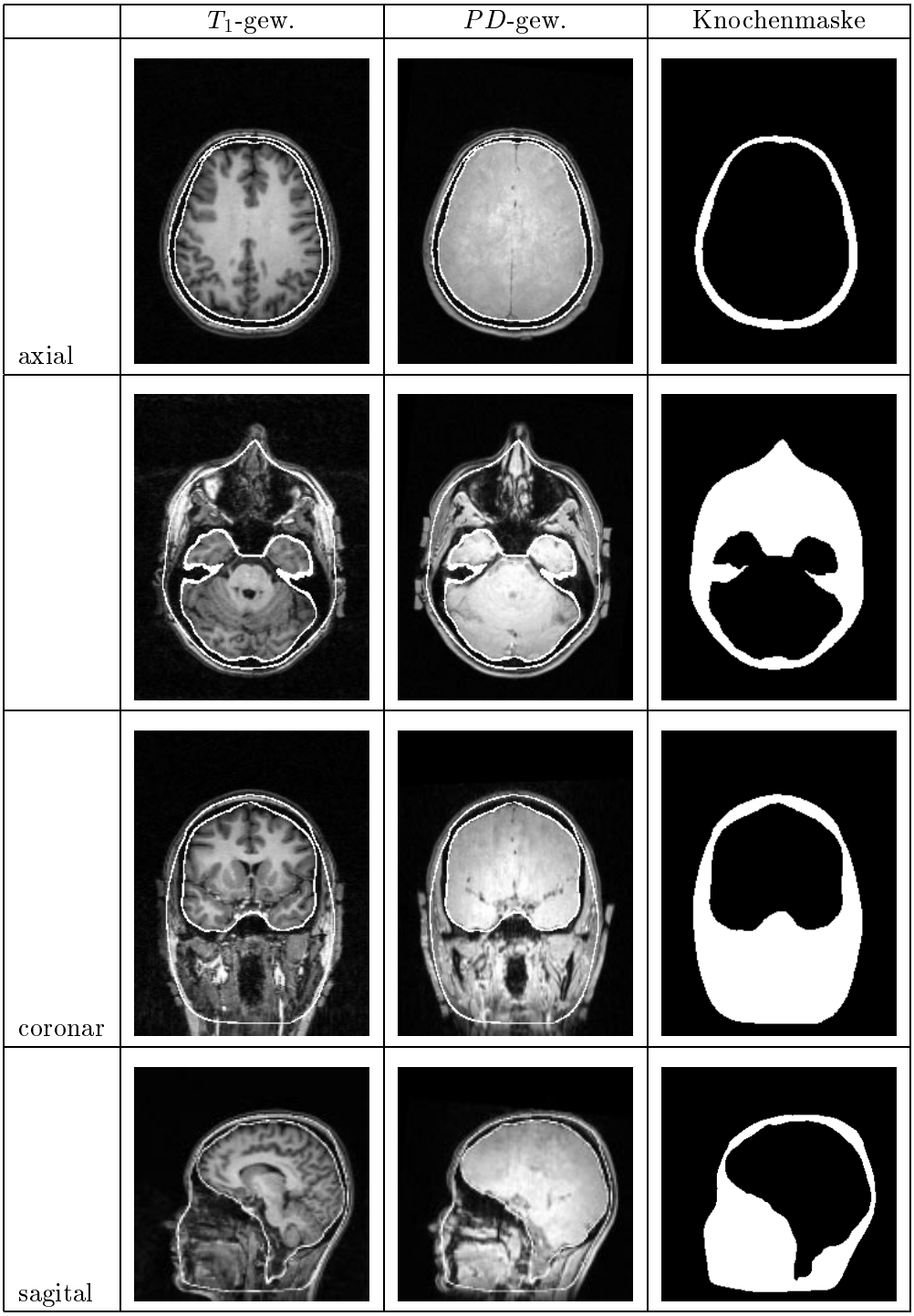


Abbildung 4.26: Innere und äußere Kante des Knochens im Datensatz 1, zwei axiale, ein coronarer und ein sagitaler Schnitt

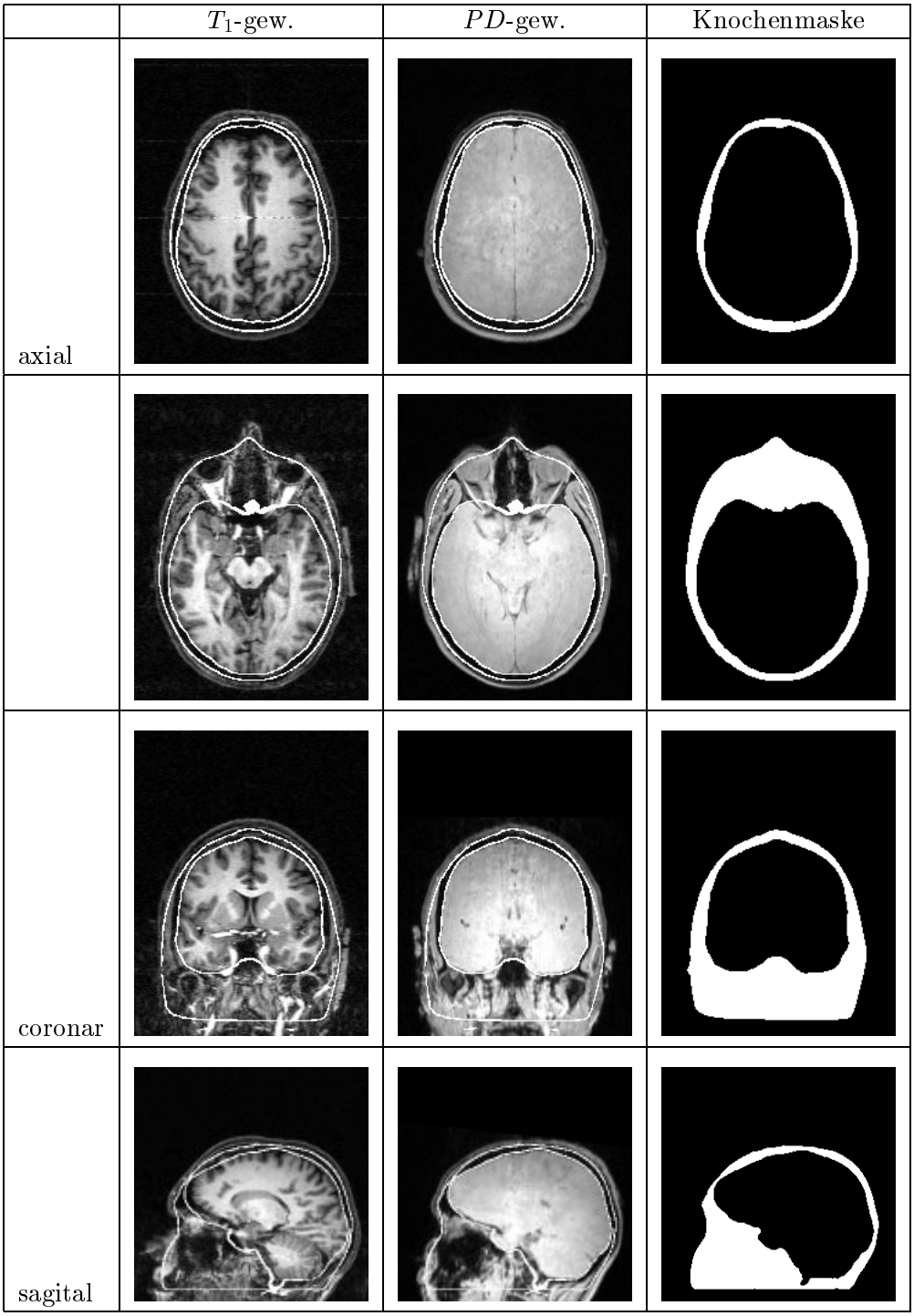


Abbildung 4.27: Innere und äußere Kante des Knochens im Datensatz 2, zwei axiale, ein coronarer und ein sagitaler Schnitt

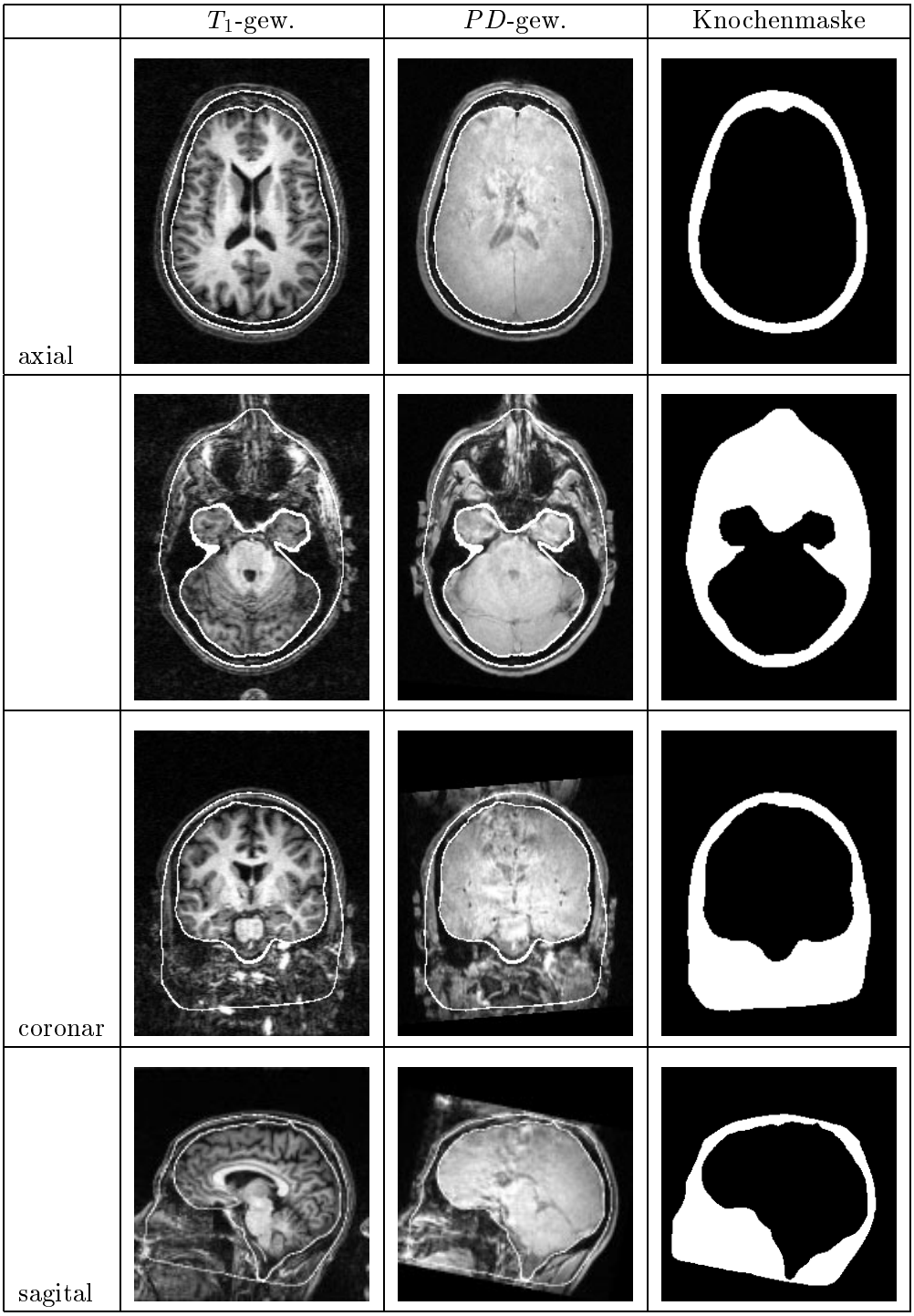


Abbildung 4.28: Innere und äußere Kante des Knochens im Datensatz 3, zwei axiale, ein coronarer und ein sagitaler Schnitt

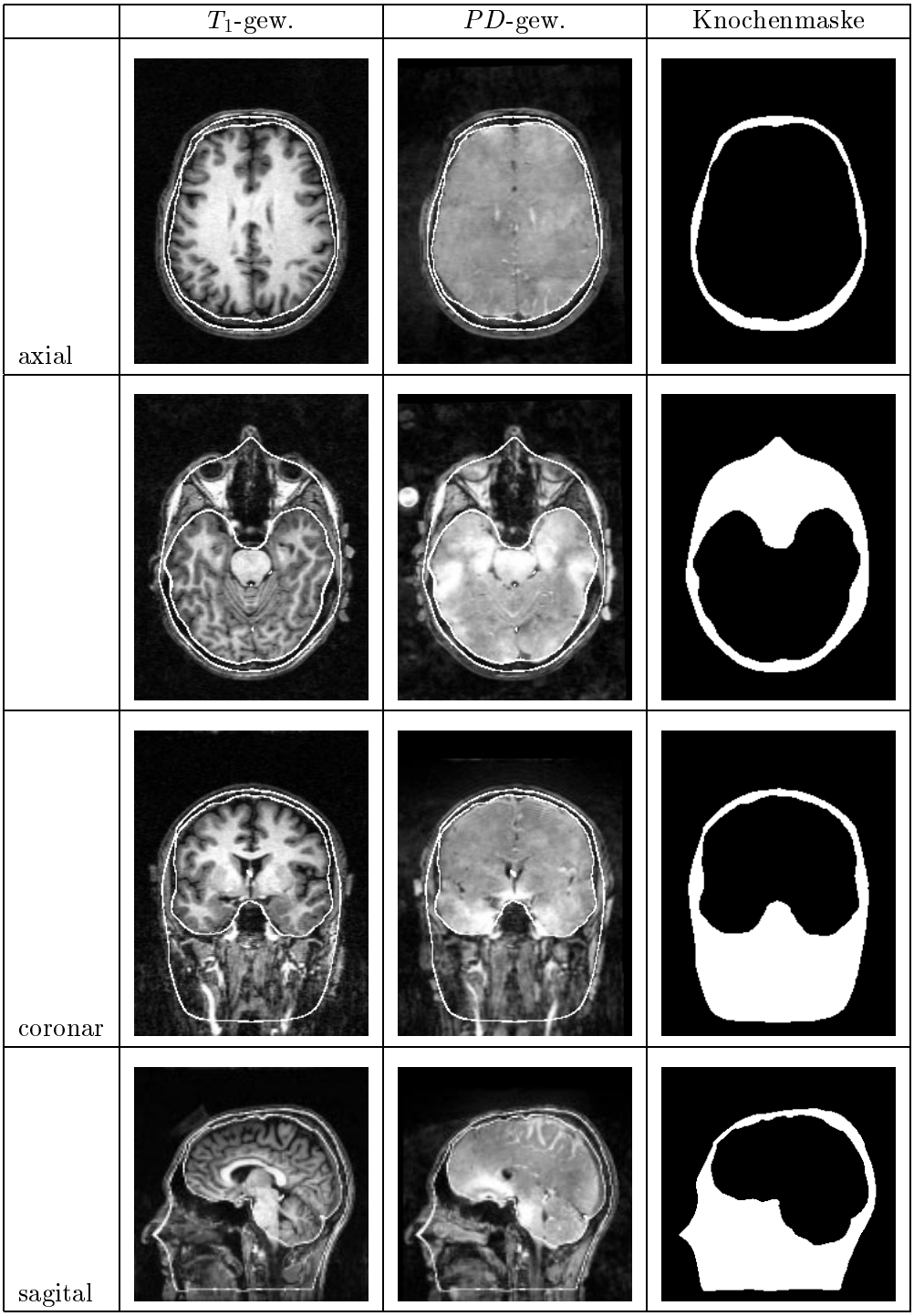


Abbildung 4.29: Innere und äußere Kante des Knochens im Datensatz 4, zwei axiale, ein coronarer und ein sagitaler Schnitt

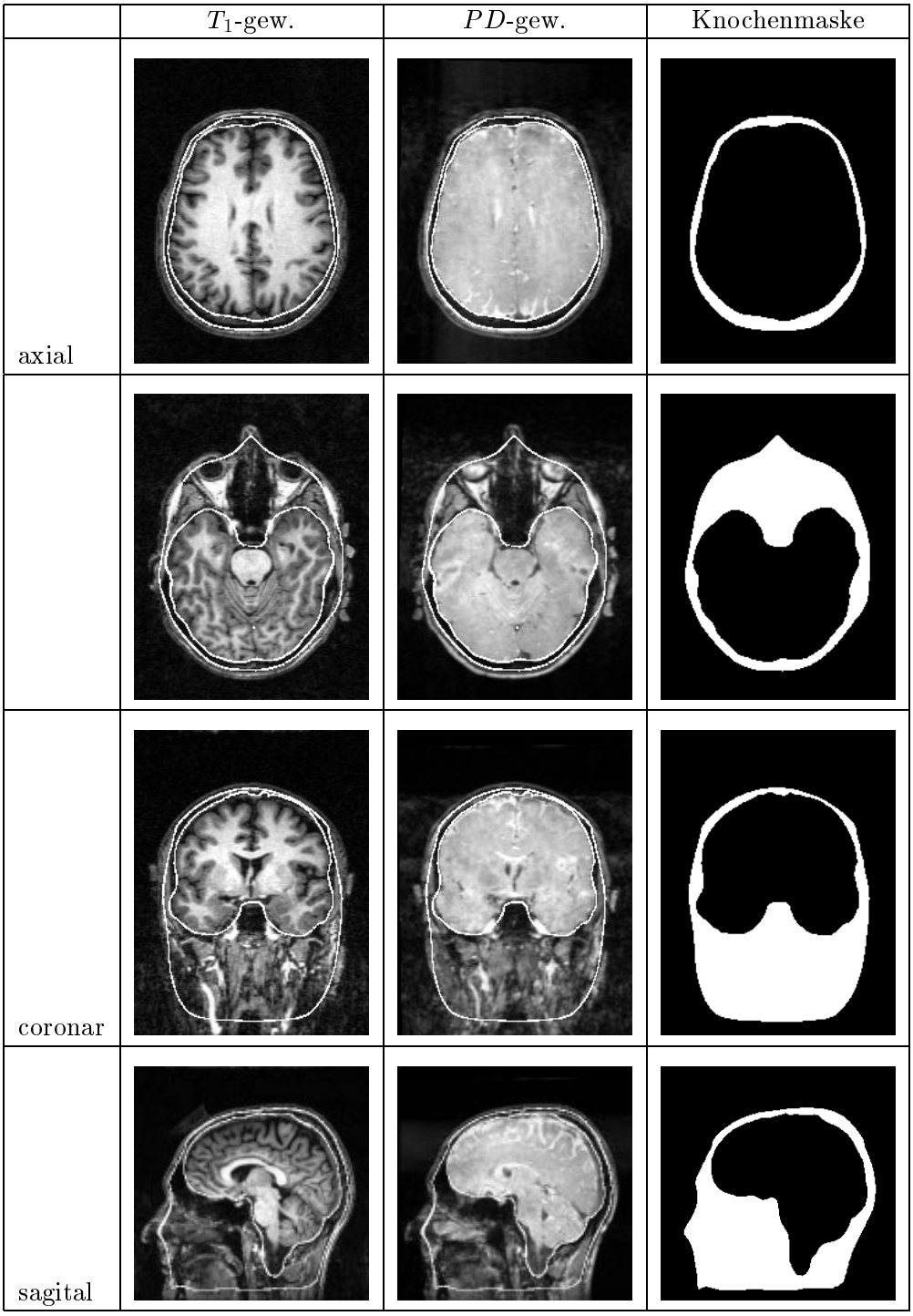


Abbildung 4.30: Innere und äußere Kante des Knochens im Datensatz 5, zwei axiale, ein coronarer und ein sagitaler Schnitt

Kapitel 5

Zusammenfassung

In dieser Arbeit wird ein Verfahren vorgestellt, um eine Segmentierung des Knochens aus Kernspinbildern zu erhalten.
Eine genaue Modellierung des Knochens ist Voraussetzung für Anwendungen in der Quellokalisation [11] oder bei der Simulation biomechanischer Eigenschaften des Kopfes. Bislang erfolgt lediglich eine recht ungenaue Schätzung der inneren Kante der Knochens.
Ausgehend von dem dual-echo-Datensatz, bestehend aus einem T_1- und einem PD-gewichteten Kernspinbild, ist das gesamte Vorgehen in die Schritte Registrierung und Segmentierung unterteilt.
Die Registrierung transformiert beide Bilder in ein gemeinsames Koordinatensystem. Dafür wird ein Verfahren mit einer Kostenfunktion auf Basis von Mutual Information verwendet. Die benötigte Abbildung setzt sich aus Skalierung, Rotation und Translation zusammen. Eine Optimierung der Kostenfunktion erfolgt mit einem Mehrgitterverfahren auf Basis eines Downhill-Simplex-Algorithmus.
Für die Segmentierung des Knochens wird ein elastisches Modell verwendet. Ausgehend von zwei initialen Segmentierungen für die innere und äußere Kante des Knochens erfolgt die Anpassung mit diesem Modell. Um die initiale Segmentierung zu erhalten, werden die Voxel in beiden Bildern mittels des AFCM-Algorithmus klassifiziert. Daraus werden zwei Masken erstellt, die die initialen Kanten repräsentieren. Von jeder Maske wird die Oberfläche als Dreiecksnetz extrahiert. Diese Netze werden danach mit dem elastischen Modell angepaßt.
Das gesamte Verfahren, bestehend aus Registrierung und Segmentierung, wird auf fünf Datensätze angewendet. Die Registrierung der Bilder erforderte bei Verwendung eines Parallelrechners maximal eine Zeit von 60 Minuten. Bei der Überprüfung der Ergebnisse haben sich nur Fehler erkennen lassen, die unter der Diskretisierung im Voxelgitter lagen.
Die Segmentierung auf den registrierten Datensätzen lieferte korrekt segmentierte innere und äußere Kanten des Knochens. Es wird dadurch eine bessere Segmentierung des Knochens im Vergleich zu bislang existierenden Verfahren erreicht. Die Verbesserung ist durch eine korrekt segmentierte innere Kante, das heißt, der Kante zwischen Knochen und Gehirnflüssigkeit, möglich. Als weiterer Vorteil dieses Segmentierungsverfahrens hat sich herausgestellt, daß bei allen

verwendeten Datensätzen die Kanten ohne manuellen Eingriff korrekt gefunden wurden. Das Segmentierungsverfahren hat aber auch zwei Nachteile. Zum einen ist das die benötigte Zeit von ca. 530 Minuten für eine komplette Segmentierung, die durch das sehr aufwendige Lösen großer Gleichungssysteme im AFCM-Algorithmus bedingt ist. Hier könnte allerdings durch den Einsatz eines Multigrid-Verfahrens eventuell eine starke Beschleunigung erzielt werden. Außerdem wird der Knochen, wie auch in derzeit verfügbaren Anwendungen, als homogenes, festes Gewebe zwischen der inneren und der äußeren Kante angenommen. In der Realität befinden sich im Knochen jedoch belüftete Kammern, wie beispielsweise die Stirnhöhlen.
Zusammenfassend läßt sich feststellen, daß es durch diese Arbeit möglich ist, eine verbesserte Segmentierung des Knochens aus Kernspinbildern zu erhalten. Die Verfahren verwenden dual-echo-Datensätze, bestehend aus einem T_1- und einem PD-gewichteten Kernspinbild, und liefen auf den Testdaten ohne Eingriff des Anwenders.

Anhang A

Verwendete Operationen

An dieser Stelle sind die in der Arbeit verwendeten Operationen auf den Bildern zusammengestellt.
Die Operationen werden auf ein Bild $b = (\mathcal{T}, \mathcal{G}, I)$ angewendet. Das Ergebnis ist ein Bild $b_{erg} = (\mathcal{T}, \mathcal{G}_{erg}, I_{erg})$ mit derselben Trägermenge $\mathcal{T}$.
In der folgenden Tabelle sind die verwendeten Operationen aufgeführt. Angegeben sind jeweils die Grauwertmenge $\mathcal{G}$ des Bildes, auf das die Operation angewendet werden kann. Weiterhin sind die Grauwertmenge des Ergebnisses $\mathcal{G}_{erg}$ und eine kurze Beschreibung der Operation vorhanden.

Operation	$\mathcal{G}$	$\mathcal{G}_{erg}$	Beschreibung
afcm	$\subseteq \mathbb{R}$	$\mathbb{N}_0$	Beschreibung siehe Kapitel 4.3, Seite 44 ff.
and	{0,1}	{0,1}	and verknüpft zwei Bilder $(\mathcal{T}, \{0,1\}, I_k), k \in \{1,2\}$ mit logischem *und*. I_{erg} ist als $I_{erg}(\mathbf{x}) = \begin{cases} 1 & \text{falls} \quad I_1(\mathbf{x}) = I_2(\mathbf{x}) = 1 \\ 0 & \text{sonst} \end{cases}$ definiert.
biggestcomp	{0,1}	{0,1}	biggestcomp wählt die größte zusammenhängende Komponente im Vordergrund bezüglich der 26-er Nachbarschaft aus. Gehört ein Voxel $\mathbf{x}$ zu dieser, dann ist $I_{erg}(\mathbf{x}) = 1$. Ansonsten ist $I_{erg}(\mathbf{x}) = 0$. Ein Voxel $\mathbf{x}$ gehört zum Vordergrund, falls $I(\mathbf{x}) = 1$ ist.
binarize	$\subseteq \mathbb{R}$	$\{0,1\}$	binarize wird durch zwei Parameter g_{min} und g_{max} gesteuert. I_{erg} ist als $I_{erg}(\mathbf{x}) = \begin{cases} 1 & \text{falls} \quad g_{min} \leq I(\mathbf{x}) \leq g_{max} \\ 0 & \text{sonst} \end{cases}$ definiert.
dist	$\{0,1\}$	$\mathbb{R}$	dist bezeichnet eine Distanztransformation. $I_{erg}(\mathbf{x})$ gibt den Abstand des Voxels $\mathbf{x}$ vom Vordergrund im Ausgangsbild an. Ein Voxel $\mathbf{x}$ gehört zum Vordergrund, wenn $I(\mathbf{x}) = 1$ ist. Die Distanztransformation erfolgt nach [31].
gauss	$\subseteq \mathbb{R}$	$\mathcal{G}$	gauss wendet einen Gaußfilter auf ein Bild an. Das Ergebnis ist eine gefilterte Version dieses Bildes.
invert	{0,1}	{0,1}	invert invertiert ein Bild. I_{erg} ist durch $I_{erg}(\mathbf{x}) = \begin{cases} 1 & \text{falls} \quad I(\mathbf{x}) = 0 \\ 0 & \text{falls} \quad I(\mathbf{x}) = 1 \end{cases}$ definiert.
isodata	$\subseteq \mathbb{R}$	$\mathbb{N}_0$	Beschreibung siehe Kapitel 4.1, Seite 33 ff.

Literaturverzeichnis

[1] Allgower, E. L. und K. Georg: *Numerical Continuation Methods - An Introduction*. Springer, 1990.

[2] Ardekani, B. A., J. Kershaw, M. Braun und I. Kanno: *Automatic detection of the mid-sagittal plane in 3-D brain images*. IEEE Transactions on Medical Imaging, 16(6):947 – 952, 1997.

[3] Bedell, B. J., P. A. Narayana und D. A. Johnston: *Three-dimensional MR image registration of the human brain*. Magnetic Resonance in Medicine, 35:384 – 390, 1996.

[4] Bezdek, J. C., L. O. Hall und L. P. Clarke: *Review of MR image segmentation techniques using pattern recognition*. Medical Physics, 20(4):1033 – 1048, 1993.

[5] Clarke, L. P., R. P. Velthuizen, M. A. Camacho, J. J. Heine, M. Vaidyanathan, L. O. Hall, R. W. Thatcher und M. L. Silbiger: *MRI Segmentation: Methods and applications*. Magnetic Resonance Imaging, 13(3):343 – 368, 1995.

[6] Collignon, A., F. Maes, D. Delaere, D. Vandermeulen, P. Suetens und G. Marchal: *Automated multimodality medical image registration using information theory*. In: Bizais, Y. und C. Barillot (Hrsg.): *Information Processing in Medical Imaging*, S. 263 – 274. Kluwer Academic Publishers, Dordrecht, 1995.

[7] Gee, J. C., C. Barillot, L. le Briquer, D. R. Haynor und R. Bajcsy: *Matching structural images of the human brain using statistical and geometrical image features*. In: *Visualization in Biomedical Computing 1994*, Bd. 2359, S. 191 – 204. SPIE Press, Bellingham, WA, 1994.

[8] Gerig, G., J. Martin, R. Kikinis, O. Kübler, M. Shenton und F. A. Jolesz: *Automating segmentation of dual-echo MR head data*. In: Colchester, A. C. F. und D. J. Hawkes (Hrsg.): *Information Processing in Medical Imaging - Proceedings of the 12th IPMI*, S. 175 – 187. Springer, Heidelberg, 1991.

[9] Gueziec, A. und R. Hummel: *The wrapper algorithm: surface extraction and simplification*. In: *Workshop on Biomedical Image Analysis*, S. 204 – 213. IEEE Computer Press, Los Alamitos, 1994.

[10] Hill, D., C. Studholme und D. Hawkes: *Voxel similarity measures for automated image registration.* In: Robb, A. R. (Hrsg.): *Visualization in Biomedical Computing 1994*, Bd. 2359, S. 205 – 216. SPIE Press, Bellingham, WA, 1994.

[11] Huiskamp, G., M. Vroeijenstijn, R. van Dijk, G. Wieneke und A. C. van Huffelen: *The need for correct realistic geometry in the inverse EEG problem.* IEEE Transactions on Biomedical Engineering, 46(11):1281 – 1287, 1999.

[12] Jähne, B.: *Digitale Bildverarbeitung.* Springer, Berlin, 4. Aufl., 1997.

[13] Kao, Y.-H., J. A. Sorenson, M. M. Bahn und S. S. Winkler: *Dual-Echo MRI segmentation using vector decomposition and probability techniques: A two-tissue model.* Magnetic Resonance in Medicine, 32:342 – 357, 1994.

[14] Kao, Y.-H., J. A. Sorenson und S. S. Winkler: *MR image segmentation using vector decomposition and probability techniques: A general model and its application to dual-echo images.* Magnetic Resonance in Medicine, 35:114 – 125, 1996.

[15] Kass, M., A. Witkin und D. Terzopoulos: *Snakes: Active contour models.* International Journal of Computer Vision, 1:321–331, 1987.

[16] Kruggel, F.: *Multimodale Registrierung - Algorithmen und Applikationen.* In: Arnolds, B., H. Müller, D. Saupe und T. Tolxdorff (Hrsg.): *5. Workshop Digitale Bildverarbeitung in der Medizin*, S. 139 – 145. Zentralstelle für Forschungsförderung und Technologietransfer, Albert-Ludwigs-Universität, Freiburg, 1997.

[17] Kruggel, F. und G. Lohmann: *BRIAN (Brain Image Analysis) - A tool for the analysis of multimodal brain datasets.* In: *Computer Assisted Radiology.* Elsevier, 1996.

[18] Kruggel, F. und D. Y. von Cramon: *Alignment of MR brain data sets with the stereotactical coordinate system.* Medical Image Analysis, 3(2):1 – 11, 1999.

[19] Kruggel, F. und D. Y. von Cramon: *Measuring the cortical thickness.* In: *IEEE Workshop on Mathematical Methods in Biomedical Image Analysis, Hilton Head Island, SC, USA*, S. 154–161. IEEE Computer Society Press, Los Alamitos, 2000.

[20] Liang, Z., J. R. MacFall und D. P. Harrington: *Parameter estimation and tissue segmentation from multispectral MR images.* IEEE Transactions on Medical Imaging, 13(3):441–449, 1994.

[21] Lohmann, G.: *Volumetric Image Analysis.* John Wiley & Sons, Chicester, 1998.

[22] Maes, F., A. Collignon, D. Vandermeulen, G. Marchal und P. Suetens: *Multimodality image registration by maximization of mutual information*. IEEE Transactions on Medical Imaging, 16(2):187 – 198, 1997.

[23] Maintz, J. B. A. und M. A. Viergever: *A survey of medical image registration*. Medical Image Analysis, 2(1):1 – 36, 1998.

[24] Mittelhäusser, G. und F. Kruggel: *Fast segmentation of brain magnetic resonance tomograms*. In: *Lecture Notes in Computer Science*, Bd. 905, S. 237 – 241. Springer, Heidelberg, 1995.

[25] Otte, M.: *Registrierung funktionaler und anatomischer MR-Bilddaten*. In: Arnolds, B., H. Müller, D. Saupe und T. Tolxdorff (Hrsg.): *5. Workshop Digitale Bildverarbeitung in der Medizin*, S. 152 – 157. Zentralstelle für Forschungsförderung und Technologietransfer, Albert-Ludwigs-Universität, Freiburg, 1997.

[26] Palubinskas, G.: *Routinen zur Segmentierung des Knochens aus PD-gewichteten Kernspinbildern*. Max-Planck-Institut für neuropsychologische Forschung, Leipzig, UNIX-Shellskripte, unveröffentlicht, 1997.

[27] Payne, B. A. und A. W. Toga: *Surface mapping of brain function on 3D models*. IEEE Computer Graphics and Applications, 10:33 – 41, 1990.

[28] Pham, D. L. und J. L. Prince: *An adaptive fuzzy C-means algorithm for image segmentation in the presence of intensity inhomogenities*. Pattern Recognition Letters, 20(1):57–68, 1999.

[29] Press, W. H., S. A. Teukolsky, W. T. Vetterling und B. P. Flannery: *Numerical Recipes in C*. Cambridge University Press, 2. Aufl., 1992.

[30] Roche, A., G. Malandain, N. Ayache und S. Prima: *Towards a better comprehension of similarity measures used in medical image registration*. In: Taylor, C. und A. Colchester (Hrsg.): *Medical Image Computing and Computer-Assisted Intervention-MICCAI'99*, S. 555–566. Springer, Berlin, 1999.

[31] Saito, T. und J.-I. Toriwaki: *New algorithms for euclidean distance transformation of an n-dimensional digitized picture with applications*. Pattern Recognition, 27(11):1551 – 1565, 1994.

[32] Schwarz, H. R.: *Methode der finiten Elemente*. Teubner, Stuttgart, 3 Aufl., 1991.

[33] Staib, L. H. und X. Lei: *Intermodality 3D medical image registration with global search*. In: *IEEE Workshop on Biomedical Image Analysis*, S. 225 – 234. IEEE Computer Society Press, Los Alamitos, 1994.

[34] Studholme, C., D. L. G. Hill und D. J. Hawkes: *Automated 3-D registration registration of MR and CT images of the head*. Medical Image Analysis, 1(2):163 – 175, 1996.

[35] Terzopoulos, D. und K. Fleischer: *Deformable models*. The Visual Computer, 4:306–331, 1988.

[36] Thirion, J.-P.: *Fast non-rigid matching of 3D medical images*. Techn. Ber. 2547, Institut National de Recherche en Informatique et en Automatique, 1995.

[37] Weickert, J.: *Scale-space properties of nonlinear diffusion filtering with a diffusion tensor*. Techn. Ber. 104, Laboratory of Technomathematics, University of Kaiserslautern, P.O. Box 3049, D-67653 Kaiserslautern, 1994.

[38] Weickert, J.: *Fast segmentation methods based on partial differential equations and the watershed transformation*. In: *Mustererkennung 1998*, S. 93 – 100. Springer, Berlin, 1998.

[39] Weikert, J.: *A review of nonlinear diffusion filtering*. In: ter Haar Romeny, B., L. Florack, J. Koenderink und M. Viergever (Hrsg.): *Scale-Space Theory in Computer Vision, Lecture Notes in Comp. Science*, Bd. 1252, S. 3 – 28. Springer, Berlin, 1997.

[40] Wells, W. M., P. Viola, H. Atsumi, S. Nakajima und R. Kikinis: *Multi-modal volume registration by maximization of mutual information*. Medical Image Analysis, 1(1):35 – 51, 1996.

[41] Wolberg, G.: *Digital Image Warping*. IEEE Computer Society Press, Washington, 1990.

[42] Wolters, C., S. Reitzinger, A. Basermann, S. Burkhardt, U. Hartmann, F. Kruggel und A. Anwander: *Improved tissue modelling and fast solver methods for high resolution FE-modelling in EEG/MEG-source localization*. In: *Internatinal Conference on Biomagnetism, BIOMAG 2000, Helsinki*, 2000.
http://www.ccrl-nece.technopark.gmd.de/simbio/publications.html.

[43] Xu, C., D. L. Pham, M. E. Rettmann, D. N. Yu und J. L. Prince: *Reconstruction of the human cerebral cortex from magnetic resonance image*. IEEE Transactions on Medical Imaging, 18(6), 1999.

[44] Xu, C. und J. L. Prince: *Snakes, shapes and gradient vector flow*. IEEE Transactions on Medical Imaging, 7(3):359 – 369, 1998.

Erklärung

Ich erkläre hiermit, daß ich die vorliegende Arbeit selbständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel angefertigt habe.

Leipzig, den 5. Oktober 2000 Stefan Burkhardt