Ein CI-unterstütztes Rahmenmodell für die medizinische Bildanalyse

Dissertation
zur Erlangung des Grades eines
Doktors der Naturwissenschaften
der Universität Dortmund
am Fachbereich Informatik

Jens Hiltner

Dortmund 2001

Tag der mündlichen Prüfung: 30. August 2001
Dekan: Prof. Dr. Thomas Hermann
1. Gutachter: Prof. Dr. Bernd Reusch
2. Gutachter: Prof. Dr. Heinrich Müller

Inhaltsverzeichnis

Notationen

Die folgenden Abkürzungen, Variablen und Schreibweisen sind in der Arbeit konsistent verwendet worden. Allgemein übliche Variablen, Schreibweisen oder Abkürzungen wie $\mathbb{R}$, $\mathbb{N}$, z.B., usw. werden als bekannt vorausgesetzt.

Verwendung von Variablen

I	Bild (Image)
p	Bildpunkt (Pixel)
x	Vertikale Bildpunktkoordinate (vgl. Abbildung 2.1)
y	Horizontale Bildpunktkoordinate (vgl. Abbildung 2.1)
d_x	Vertikale Ausmaße eines Bildpunktes in [mm]
d_y	Horizontale Ausmaße eines Bildpunktes in [mm]
S	Segment (Bildpunktmenge, sowohl scharf als auch unscharf)
M	Zeilenanzahl eines Bildes
N	Spaltenanzahl eines Bildes
N_4	Vierernachbarschaft
N_8	Achternachbarschaft
G_{min}	Minimaler Grauwert
G_{max}	Maximaler Grauwert
$h(g)$	Histogrammfunktion
$\langle 0, 1\rangle$	Einheitsintervall $\subseteq \mathbb{R}$
$\mathcal{A}, \mathcal{B}, \ldots$	Fuzzy-Mengen
$\mathcal{A}_\diamond, \mathcal{B}_\diamond, \ldots$	Fuzzy-Variablen
θ	Typ einer Fuzzy-Menge
$a_1, \ldots, a_4$	Parameter einer Fuzzy-Menge
μ	Zugehörigkeitsfunktion
$\hat{\mu}$	Anzahl der Elternindividuen
ρ	Anzahl der Eltern in einer Rekombination
λ	Anzahl der Nachkommen einer Generation
V	Individuum
Θ	Bias eines Neurons
ω	Wissensbasis

Verwendete Abkürzungen und Schreibweisen

BA	Bildanalyse
BV	Bildverarbeitung
CI	Computational Intelligence
CNN	Zellulares neuronales Netz (engl. *cellular neural net*)
EA	Evolutionäre Algorithmen
ES	Evolutionsstrategien
FF	Vorwärtsgerichtetes Netzwerk (engl. *feed-forward*)
FL	Fuzzy-Logik (oder allgemein Fuzzy-Methoden)
FM	Fuzzy-Menge
FOM	Fuzzy-Objekt-Modell
GA	Genetische Algorithmen
GUI	Grafische Benutzeroberfläche (engl. *graphical user interface*)
KNN	Künstliches neuronales Netz (engl. *artificial neural net*)
MRT	Magnet-Resonanz-Tomographie
NN	Neuronales Netz
ROI	Interessierender Bildbereich (engl. *region of interest*)
WR	Wissensrepräsentation
C.N	Kurzschreibweise für ein Objekt N im Kontext C
$[A_1 \mid A_2 \mid *]$	Alternative Angaben A_1 oder A_2, wobei $*$ eine leere Alternative bedeutet
$\stackrel{\text{def}}{=}$	Definition oder Wertzuweisung

Die Groß-/Kleinschreibung ist in Programmteilen und auch in den Wissensbasen ohne Bedeutung.

1 Einleitung

1.1 Bildverarbeitung im medizinischen Umfeld

In den unterschiedlichsten medizinischen und technischen Bereichen treten immer wieder diverse Fragestellungen auf, die eine Verarbeitung von Bildern erfordern. Der Mensch kann Informationen, die er visuell wahrnimmt, größtenteils problemlos verarbeiten und verstehen. Ein maschinelles Verstehen von Bildern steht aber noch immer erst am Anfang der Entwicklung einer realen Nutzbarkeit.

Aufgrund der Vielzahl von Daten, die in heutigen Untersuchungen anfallen, ist es aber kaum noch denkbar, diese Menge von Informationen manuell zu verwalten und auszuwerten. Die Forderungen nach Vorsorge- und Reihenuntersuchungen (engl. *Screening-Tests*) erfordern eine schnelle und – im Sinne von Reproduzierbarkeit – objektivierte Auswertung des erhobenen Datenmaterials, um eine sowohl inter- als auch intra-individuelle Varianz in den Entscheidungen eines menschlichen Bewerters möglichst auszuschließen.

Im Rahmen dieser Arbeit steht die automatisierte Auswertung von Daten – speziell von medizinischem Bildmaterial aus der Radiologie – im Vordergrund. Für die Akquisition radiologischen Bildmaterials werden immer neue, bezüglich der Qualität und Geschwindigkeit verbesserte Methoden und Geräte entwickelt und in der täglichen Routineuntersuchung etabliert. Häufig ist die zunehmende Menge an Bildern, die bei einer Untersuchung anfallen, jedoch ohne Befund (also Bilddaten, in denen keine Pathologie festgestellt werden kann). Trotzdem ist der Mediziner mit der Auswertung des Bildmaterials einen längeren Zeitraum beschäftigt, zumindest mehrere Minuten. Hier ist es wünschenswert, daß ein automatisiertes System existiert, welches zumindest die *sicher ohne Befund* vorliegenden Bilder aussortieren kann, um einen Zeitgewinn des Arztes für die problembehafteten Fälle zu erreichen.

Jedoch weisen gerade medizinische Bilddaten eine Vielzahl von Problemen auf, die durch Unsicherheiten verschiedenster Arten bedingt sind. Unter Unsicherheit wird hier zunächst alles subsumiert, was nicht in sicherer oder exakter Form beschreibbar bzw. vorhanden ist. Im Besonderen wird hier das unscharfe Wissen, welches über das untersuchte Bildmaterial vorliegt, betrachtet. Auf den Begriff des unscharfen Wissens wird in Kapitel 3 ausführlich eingegangen, da es verschiedene Arten unscharfen Wissens gibt und auch eine unterschiedliche Mo-

Der Kopf befindet sich im mittleren Bereich des Bildes

Das Gehirn befindet sich im oberen Bereich des Kopfes

Das Gehirn ist oben rund

Das Gehirn ist groß

Das Gehirn ist mittel–hell

(a) MRT-Bild

(b) Textuelle Beschreibung

Abbildung 1.1: Beschreibung eines MRT-Datensatzes

dellierung erfolgen muß. Neben der Form des unsicheren Wissens, welches durch Wahrscheinlichkeiten ausgedrückt werden kann, existiert noch das unscharfe (vage) Wissen, dem in dieser Arbeit besondere Bedeutung zukommt, da es besonders gut mit Hilfe von Fuzzy-Methoden modelliert werden kann. Dieses unscharfe Wissen ist täglich allgegenwärtig und wird bei vielen menschlichen Entscheidungen bewußt oder unbewußt genutzt.

Auch das Wissen über Bildinhalte ist vage und unscharf. Obwohl in der Regel das dargestellte Objekt in einem Bild bekannt ist, kann eine Beschreibung, die auf eine ganze Klasse von Bildern zutreffen soll, nur mit unscharfen Begriffen erfolgen, wenn es sich um medizinisches Bildmaterial handelt. Hier sei etwa die (unscharfe) Beschreibung eines Magnetresonanztomographie-Bilddatensatzes des menschlichen Kopfes genannt, die das Gehirn in Sagittal-Ansicht (auszugsweise) darstellt (vgl. Abbildung 1.1).

Es geht hierbei nicht um die Fragestellung, *ob* ein gefundenes Objekt einer Beschreibung entspricht, sondern *wie sehr* es der Beschreibung entspricht. Auch die Fragestellung, *wie viele* von allen existierenden Objekten derselben Klasse der Beschreibung entsprechen, ist nicht Gegenstand dieser Untersuchung.

Ziel dieser Arbeit ist die

> *Entwicklung eines Rahmenmodells, welches es Anwendern insbesondere aus dem medizinischen Umfeld erlaubt, neben den bekannten Bildverarbeitungsmethoden auf umfangreiche Bibliotheken von Methoden aus dem Bereich der Computational Intelligence zurückzugreifen. Dies macht im Vorfeld eine Untersuchung erforderlich, inwieweit mit Hilfe dieser Methoden (im folgenden kurz CI-Methoden genannt) die Segmentierung und Analyse medizinischer Bilddaten überhaupt unterstützt werden kann.*

Neben der konzeptionellen Entwicklung dieses Rahmenmodells wird zudem ein Software-System entwickelt, das dieses Modell in Form einer einfach verwendbaren Programmiersprache realisiert.

Die Programmiersprache wird auf Grundoperatoren basieren, welche in direkter Beziehung zu frei parametrisierbaren Bildverarbeitungsoperatoren stehen. Zudem müssen Kontrollstrukturen wie Schleifen, bedingte Programmausführungen und lokale Funktionen definiert werden, um eine Sprache zu realisieren, die mächtig genug ist, konkrete Probleme angemessen zu beschreiben und zu lösen.

1.2 Anforderungen an eine CI-Bildverarbeitung

Eine große Neuerung, die sich aus dem Rahmen dieser Arbeit ergibt, soll darin bestehen, daß unterschiedliche CI-Methoden Teil der Sprache werden. Somit unterscheidet sich der zu wählende Ansatz in großen Teilen von bestehenden Programmierumgebungen für die medizinische Bildverarbeitung.

Dabei muß die neue Sprache folgenden Eigenschaften genügen: Sie muß einfach und intuitiv benutzbar sein, um Anwendern einen leichten Einstieg zu ermöglichen. Dazu dient vor allem der modulare Aufbau der Sprache. Eine weitere wichtige Eigenschaft muß die leichte Erweiterbarkeit um neue Sprachkonstrukte sein, so daß zu jeder Zeit neue Anforderungen einfach in die Programmierumgebung zu integrieren sind.

Der entwickelte Lösungsansatz wird anhand eines Anwendungsszenarios für die Auswertung von MRT-Datensätzen exemplarisch gezeigt, wobei die Umsetzung in dieser genannten Programmiersprache erfolgt. Der Schwerpunkt liegt in dieser Arbeit auf der Verwendung von Expertenwissen und dessen Modellierung mit Hilfe von Fuzzy-Methoden. An den Stellen, an denen eine Ergänzung mit Hilfe neuronaler Netze sinnvoll erscheint, werden diese integriert. Parallel dazu finden evolutionäre Algorithmen zur Optimierung des verwendeten Wissens ihre Anwendung.

Dabei wird auf dem Modell aus [GW93] aufgebaut, da dieses Buch eines der wenigen Standardwerke zum Thema Bildverarbeitung ist, welches die Verwendung von Wissen explizit herausstellt und die Möglichkeit zur Nutzung einer (externen) Wissensbasis erwähnt. Die dort aber nicht ausreichende Behandlung von Unsicherheiten wird mit dieser Arbeit mittels CI-Methoden ergänzt. Zudem wird eine unscharfe Variante bildhafter Modelle, die sogenannten Fuzzy-Objekt-Modelle, neu vorgestellt. Die Arbeit gipfelt in der Realisierung einer Programmiersprache zur Analyse medizinischer Bilddaten, die auf die genannten CI-Komponenten zurückgreifen kann.

Auch an die zu entwickelnde Programmierumgebung müssen verschiedene Anforderungen formuliert und erfüllt werden. Diese Anforderungen basieren auf unterschiedlichen Annahmen, welche aus zahlreichen Gesprächen mit Medizinern und der Untersuchung von Bildverarbeitungswerkzeugen resultieren.

- **Komponentisierbarkeit der Programmierumgebung**
  Der Vorteil einer modularen Programmierumgebung liegt hauptsächlich darin, verschiedene neue Funktionalitäten bei Bedarf einfach in das bestehende System integrieren zu können. Auch die Erweiterung bestehender Funktionen geschieht dann nur in der zu ändernden Komponente, ohne Einfluß auf den Rest der Umgebung. Für das in dieser Arbeit angesprochene Problemumfeld bieten sich zentrale Komponenten wie eine Programmieroberfläche, eine Bibliothek von Standard-Bildverarbeitungsoperatoren und Editoren zur Modellierung der verwendeten CI-Methoden an.

- **Ergonomische interaktive grafische Benutzeroberfläche**
  Obwohl das System Personen adressiert, die Bildverarbeitungserfahrungen haben sollten, muß es über ein möglichst einfaches Bedienparadigma verfügen. Somit wird die Programmierung weitestgehend durch grafische interaktive Werkzeuge unterstützt.

- **Dokumentation der verwendeten Komponenten**
  Um eine Erweiterbarkeit des Systems auch über den Rahmen dieser Arbeit hinaus zu gewährleisten, ist eine Dokumentation auf verschiedenen Ebenen notwendig. Zum einen wird die Ebene der zuvor genannten Kompontenten angesprochen, zum anderen sind die Einzelfunktionen zu dokumentieren, die sich hinter den einzelnen Komponenten verbergen.

1.3 Struktur der Arbeit

Entsprechend der zuvor gegebenen Motivation und Zielsetzung gliedert sich die Arbeit in folgende Bereiche:

- Das zweite Kapitel dieser Arbeit führt in die grundsätzlichen Begriffe und Probleme der digitalen Bildverarbeitung ein und gibt einen Überblick über spezielle Fragestellungen, die sich in der Medizin und besonders in der Radiologie stellen. Zudem werden Probleme präsentiert, die eine Bildanalyse medizinischen Materials erschweren.

- Die Idee, die vorher gezeigten Probleme zum Teil mit Hilfe von Methoden der Computational Intelligence lösen zu können, setzt voraus, das in dieser Arbeit verwendete Verständnis von CI zu erläutern. Dieses geschieht im dritten Kapitel.

  Da für die Verwendung von CI-Methoden umfangreiches Wissen notwendig ist, werden in diesem Kapitel auch die Grundlagen der Wissensrepräsentation und Wissensverarbeitung dargestellt. Auch der Begriff des Wissens und besonders des unscharfen und unsicheren Wissens selbst wird an dieser Stelle erläutert.

- Das vierte Kapitel gibt eine kurze Übersicht über verschiedene existierende Arbeiten, welche aber lediglich einzelne CI-Methoden in der medizinischen Bildverarbeitung einsetzen. Damit stellt es einen Bericht über den gegenwärtigen Forschungsstand dar. Neben einem zahlenmäßigen Vergleich der Forschungsschwerpunkte und gefundenen Arbeiten wird hier aus jedem Teilgebiet der Bildverarbeitung ein Ansatz aus dem Bereich der CI vorgestellt (soweit existent) und somit die Einleitung auf einen eigenen, umfassenden CI-Ansatz mit allen Methoden gegeben.

- Im fünften Kapitel wird das neue Rahmenmodell eingeführt, welches die Komposition aller CI-Methoden in verschiedenen Stufen der Bildverarbeitung vorschlägt und Schwerpunkte für diese Bereiche angibt, in denen sich eine besondere Eignung gezeigt hat. An dieser Stelle soll nochmals hervorgehoben werden, daß es sich nicht um ein monolithisches System handelt, welches für eine Aufgabe optimiert wurde, sondern daß mit Hilfe dieses Rahmenmodells verschiedene Probleme aus dem Bereich der medizinischen Bildverarbeitung durch Unterstützung von CI-Methoden bearbeitet werden können.

- Das Kapitel sechs stellt den Prototypen der Implementierung des im vorherigen Kapitel konzeptionell entwickelten Rahmenmodells vor und zeigt auf, wie mit dem implementierten System CIM^{2}BA/P gearbeitet werden kann. Dazu werden die einzelnen Teile des Systems näher vorgestellt. Zudem wird gezeigt, wie die CI-Methoden integriert werden können und welche Schnittstellen existieren.

- Das folgende siebte Kapitel zeigt einen Ansatz für die Segmentierung von MRT-Datensätzen des Kopfes. Die Umsetzung erfolgt in der entwickelten Programmiersprache von CIM^{2}BA/P und verwendet ausschließlich die vorgestellten Konzepte des Rahmenmodells, wobei vielfältig auf die CI-Methoden zurückgegriffen wird.

- In den Abschlußbemerkungen wird dargelegt, in welchem Umfang die gesetzten Ziele und geplanten Lösungen erreicht bzw. entwickelt wurden. Noch nicht erbrachte Lösungen werden im Ausblick angedacht und diskutiert.

- Der Anhang der Arbeit zeigt viele Beispiele und Tests auf, vor allem werden hier anhand von Bildern Ergebnisse für mehrere Teilprobleme präsentiert, um Einsatz und Wirkung der entwickelten Methoden zu dokumentieren. Zudem werden auch die Werkzeuge und die Implementierung der Software CIM^{2}BA/P in der entsprechenden Form dokumentiert.

Das verwendete Bildmaterial stammt aus unterschiedlichen Quellen. Aus datenschutzrechtlichen Aspekten wird keine nähere Quellenangabe bei den einzelnen Abbildungen gemacht.

2 Grundlagen der digitalen Bildverarbeitung

Die Forschungsdisziplin Bildverarbeitung nimmt in dieser Arbeit eine zentrale Rolle ein. So bilden viele bekannte und mittlerweile gut ausgearbeitete Algorithmen der Bildverarbeitung ohne CI-Methoden, die deswegen hier auch als *klassische Bildverarbeitung* bezeichnet wird, das Fundament für eine um CI-Techniken erweiterte Methodensammlung. Unter diesem Aspekt treten in den nachfolgenden Abschnitten viele Begriffe auf, die in diesem Kapitel eingeführt und erläutert werden. Dies ist notwendig, da in verschiedenen Arbeiten des Forschungsgebietes gleiche Begriffe mit unterschiedlichen Bedeutungen auftauchen. Zudem muß für das prinzipielle Verständnis der später vorgenommenen Erweiterungen um CI-Methoden auch der methodische Umgang mit Bildern auf der untersten Ebene – worunter die einzelnen Bildpunkte verstanden werden sollen – und den damit behafteten Problemen betrachtet werden. Weiterhin wird ein kurzer Einblick in die medizinische Bildgebung, deren Fragestellungen und Schwierigkeiten gegeben. Trotzdem wird der Umfang möglichst klein gehalten, da eine Vielzahl an Literatur existiert, die in diesen Themenkomplex einführt (bspw. [Wah89, Hab91, Mül92, GW93, Abm94, KZ95, Jäh97, Kop97]).

2.1 Aufgaben der digitalen Bildverarbeitung

Die Verarbeitung von digitalen Bildern umfaßt nicht eine einzige zentrale Aufgabe. Es gibt vielmehr mehrere Aufgaben, die abhängig von den Informationen sind, die aus dem Bild gewonnen werden sollen oder aber den Eigenschaften, die das Bild später haben soll.

Hier können dementsprechend zwei Anwendungsfelder angegeben werden, wie sie u. a. bei Gonzalez und Woods [GW93] zu finden sind:

- die Verbesserung der Bildinformation für die menschliche Interpretation
- die automatische Erkennung/Verarbeitung des Bildinhaltes

Ursprünge der digitalen Bildverarbeitung gehen in die frühen 20er Jahre des 20. Jahrhunderts zurück (vgl. Gonzalez und Woods [GW93]). Damals wurde nach der Installation eines Transatlantikkabels der Bildtransport zwischen den USA und Europa von ca. einer Woche auf ungefähr 3 Stunden verkürzt. Die hier notwendigen Aufgaben fielen ganz klar in den ersten Teil, der Bildverbesserung. Die automatische Bildauswertung (*Bildanalyse*) wurde in den 60er Jahren mit der Entwicklung leistungsfähiger Computer vorangetrieben, obwohl bis heute noch viele Probleme ungelöst sind. Anwendungen für die Verbesserung aufgenommener Bilder oder aber deren weiteren Verarbeitung fallen in den Bereichen *Medizin*, *Militär*, *Luft-* und *Raumfahrt* an. Auch die Industrie setzt gerade im Bereich der Qualitätskontrolle mittlerweile verstärkt auf den Einsatz von Bildverarbeitungssystemen, um eine (möglichst zerstörungsfreie) Werkstoffprüfung durchzuführen [Hab91].

2.2 Grundbegriffe der digitalen Bildverarbeitung

Um über Aufgaben, Probleme und deren Lösungsmöglichkeiten sprechen zu können, müssen zunächst einige Grundbegriffe aus dem Bereich der Bildverarbeitung definiert werden. Nicht besprochen werden hier aber die Bildwahrnehmung und die physikalischen Vorgänge der Bildentstehung, da dies den Rahmen der Arbeit sprengen würde. Zudem finden sich geeignete Beschreibungen dazu etwa bei [GW93, Jäh97, Kop97].

Definition 2.1 (Bild, Bildpunkt) Ein Bild wird durch eine Funktion I charakterisiert, die für eine Menge von Bildpunkten p definiert ist. Dabei besitzt I an jedem Bildpunkt $p = (x, y)$ einen eindeutig bestimmten Funktionswert $I(p) = I((x, y))$ für $x, y \in \mathbb{R}$. □

Festlegung 2.1 (Bild, Darstellung) Im folgenden wird *nicht* zwischen einem Bild als Funktion und seiner grafischen Darstellung (die in [KZ95] als *ikonisches Bild* bezeichnet wird) unterschieden. Auf eine Angabe des Wertebereichs der Bildfunktion wird bewußt verzichtet, da dieser abhängig von der Verwendung ist und Teilmenge von $\mathbb{Z}$, $\mathbb{R}$ oder $\mathbb{C}$ sein kann. □

Festlegung 2.2 (Klammerung, Koordinatenbezug) Der Einfachheit halber wird im folgenden auf die doppelte Klammerung verzichtet, so daß $I(x, y)$ mit $I((x, y))$ gleichbedeutend ist. Bei dem Bezug auf eine einzelne Koordinate eines Bildpunktes p mit $p = (x, y)$ wird $p.x$ für x und $p.y$ für y geschrieben. □

Definition 2.2 (Digitales Bild) Ein digitales Bild ist ein Bild (vgl. Definition 2.1), bei dem sowohl die Koordinaten der Bildpunkte (x, y) als auch die Funktionswerte diskret sind, d.h. Elemente einer abzählbaren Menge sind. In der Regel gilt $x, y, I(x, y) \in \mathbb{N}$. □

Definition 2.3 (Binäres Bild, Binärbild) Ein binäres Bild ist ein digitales Bild (vgl. Definition 2.2), für dessen Funktionswerte nur zwei Werte zugelassen sind. Häufig sind dies die Werte 0 und 1 bzw. G_{min} und G_{max}, wenn G_{max} der maximal erlaubte[1] Grauwert ist, wobei $G_{min}, G_{max} \in \mathbb{N}$ gilt. □

Definition 2.4 (Nachbarn) Jeder Bildpunkt $p = (x, y)$, der nicht am Rand des Definitionsbereiches liegt, besitzt in seiner Umgebung Nachbarn. Speziell die direkten Nachbarn, die unmittelbar an p grenzen, haben in verschiedenen Verfahren eine herausragende Bedeutung. Die jeweils senkrechten Nachbarn mit den Koordinaten

$$(x, y-1), \quad (x-1, y), \quad (x+1, y), \quad (x, y+1) \tag{2.1}$$

heißen die 4-Nachbarn bzw. N_4 von p. Alle direkten Nachbarn mit den Koordinaten

$$\begin{array}{llll} (x-1, y-1), & (x, y-1), & (x+1, y-1), & (x-1, y), \\ (x+1, y), & (x-1, y+1), & (x, y+1), & (x+1, y+1) \end{array} \tag{2.2}$$

werden als 8-Nachbarn oder mit N_8 von p bezeichnet. Um neben der verwendeten Nachbarschaft auch die Menge der benachbarten Punkte eines gegebenen Punktes p zu benennen, wird die Schreibweise $N_4(p)$ bzw. $N_8(p)$ verwendet. □

Die zuvor genannten und an Klette und Zamperoni [KZ95] angelehnten Definitionen beschreiben ein allgemeines Bild. Ein *digitales Bild* ist im allgemeinen eine Funktion, deren Funktionsparameter und Funktionswerte aus der Menge der natürlichen Zahlen stammen ($\mathbb{N}_0$) und nach oben hin begrenzt sind. Repräsentiert wird das Bild im Computer dabei in Matrix-Form. Bildpunkte werden häufig auch als *Pixel* (engl. *Picture element*) bezeichnet. Der Funktionswert an der Stelle (x, y) beschreibt die *Helligkeit* oder *Intensität* (*Grauwert* bei einkanaligen Bildern) oder die *Farbe* (bei mehrkanaligen Bildern).

Festlegung 2.3 (Bild) Wird in dieser Arbeit von einem Bild gesprochen, so ist immer ein digitales Bild gemeint, wobei die Parameter und Funktionswerte positive, ganze, nach oben beschränkte Zahlen sind, wenn nichts anderes behauptet wird oder offensichtlich aus dem Kontext erkennbar ist. Ebenso handelt es sich um ein Graustufenbild, d.h. es gibt nur einen Bildkanal.

$$0 \leq I(x, y) < \infty \text{ mit } 0 \leq x \ll \infty \text{ und } 0 \leq y \ll \infty, \text{ mit } x, y, I(x, y) \in \mathbb{N}_0 \tag{2.3}$$

□

Um von Koordinaten in einem Bild zu sprechen, muß zunächst das Koordinatensystem festgelegt werden (vgl. Abbildung 2.1). Das Koordinatensystem aus Festlegung 2.4 kann somit wie eine $(M \times N)$-Matrix[2] verstanden werden.

[1] Erlaubt im Sinne von darstellbar, in der Regel ist dies 255 (8-Bit Darstellung).

[2] Auch andere Formen, etwa ein hexagonales Gitter sind denkbar, werden aber nicht weiter betrachtet, da sie für die Arbeit nicht relevant sind.

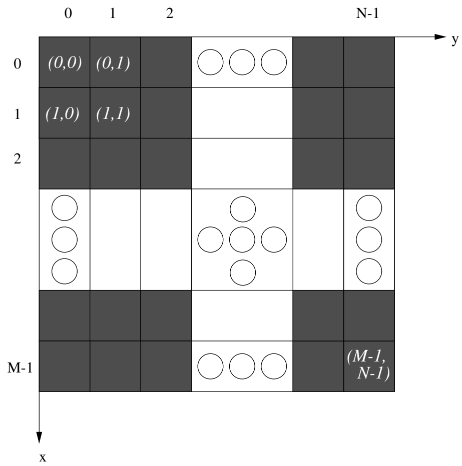


Abbildung 2.1: Schematische Darstellung der hier verwendeten Orientierung des kartesischen xy-Koordinatensystems. Für einen Bildpunkt $p = (x, y)$ beschreibt y den Spaltenindex und x den Zeilenindex. Die Zählung beginnt jeweils bei 0.

Festlegung 2.4 (Koordinatensystem) Der Nullpunkt des Koordinatensystems liegt in der linken, oberen Ecke der Bildmatrix. Die Zeilenkoordinaten $0 \leq x \leq M - 1$ verlaufen von oben nach unten, die Spaltenkoordinaten $0 \leq y \leq N - 1$ von links nach rechts. Wird nichts Gegenteiliges behauptet, so haben die Koordinatenachsen eine identische Skalierung. □

Diese Festlegung ist o.B.d.A. und beruht auf den hier hauptsächlich betrachteten Bilddaten, die diese Struktur aufweisen. Neben der so üblichen Matrizenschreibweise hat sie weiterhin den Vorteil, daß sie – um 90° negativ gedreht – dem in der Mathematik üblichen Koordinatensystem entspricht. Diese Matrizenschreibweise gilt somit für die gesamte Arbeit.

Definition 2.5 (Histogramm) Der Begriff des Histogramms eines Bildes oder Bildbereiches beschreibt die Häufigkeit des Auftretens der Grauwerte im Bild.

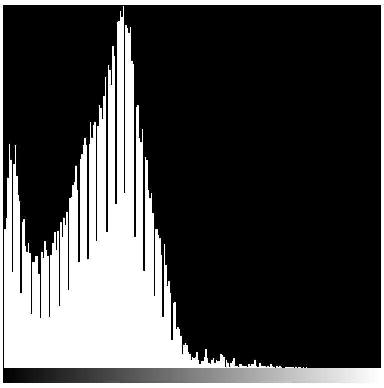
$$h(g) = \sum_{y=0}^{N-1} \sum_{x=0}^{M-1} I_g(x, y) \quad \text{mit} \tag{2.4}$$

$$I_g(x, y) = \begin{cases} 1 & \text{falls } I(x, y) = g, \\ 0 & \text{sonst} \end{cases} \tag{2.5}$$

□

Gleichung (2.4) dient zur Berechnung der *Histogrammfunktion* h eines Bildes I der Größe $M \times N$. Abbildung 2.2 zeigt ein Beispiel für ein Grauwerthistogramm.

(a) Original



(b) Histogramm

Abbildung 2.2: Ein MRT-Bild und das zugehörige Grauwert-Histogramm (mit unter dem Histogramm abgebildetem Grauwertkeil, allerdings aus Übersichtlichkeitsgründen ohne Hintergrundanteil).

Bei diesem wurde aufgrund der Übersichtlichkeit und Skalierung auf den Hintergrundgrauwert verzichtet, da dessen Anteil relativ zu den übrigen vorkommenden Grauwerten sehr hoch ist. Histogrammfunktionen dienen häufig zur *Segmentierung* (vgl. Abschnitt 2.3.2). Anhand des Histogramms ist es bei einigen Bilddaten möglich, durch Bestimmung eines oder mehrerer Schwellenwerte eine Trennung in Bildhintergrund und Bildobjekt(e) durchzuführen.

2.3 Methoden der digitalen Bildverarbeitung

Wie in Abschnitt 2.1 beschrieben, existieren verschiedene Aufgaben, die in den Bereich der *digitalen Bildverarbeitung* fallen. Abhängig von der Aufgabe ist auch die Methode, die zu ihrer Lösung verwendet wird. Die Aufgaben lassen sich entsprechend der Ziele auch in die Bereiche *Bildvorverarbeitung* und *Bildsegmentierung* einteilen. Im Rahmen eines Bildanalysesystems dienen diese Schritte als Vorbereitung zur *Beschreibung* der Bildsegmente und einer anschließenden *Objekterkennung* und *Interpretation* des Bildinhaltes. Die einzelnen Teile bzw. Schritte eines Bildanalysesystems werden nochmals in Abbildung 2.3 gezeigt und in den nächsten Abschnitten beschrieben. Diese Abbildung, die an [GW93] angelehnt ist, beinhaltet eine Wissensbasis. Erstaunlicher Weise ist dieses eines der sehr wenigen Werke aus der hier erwähnten Literatur zur Standard-Bildverarbeitung (vgl. Festlegung 2.5), welches explizit die Verwendung von Wissen erwähnt. Abmayr [Abm94] beschreibt lediglich die Möglichkeit der Wissensverwendung, in anderen Literaturquellen wird diese nur implizit angenommen und in unterschiedlich detaillierter Form nur die Prozeßkette (Abbildung 2.3) selbst beschrieben (sie-

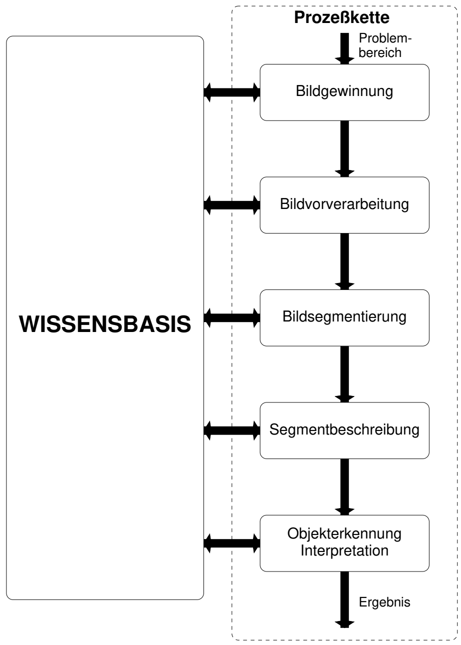


Abbildung 2.3: Grundlegende Schritte eines digitalen Bildanalysesystems.

he dazu z. B. [Köh96, Jäh97]). Daher wird das Modell aus Gonzalez und Woods [GW93] als Basis gewählt und in der vorliegenden Arbeit um CI-Methoden erweitert.

Festlegung 2.5 (Bildanalyse, Bildverarbeitung, Klassifikation)
Sofern nichts anderes behauptet wird, werden die Begriffe Bildverarbeitung und Bildanalyse synonym verwendet. Hiermit können dann alle Schritte der Bildverarbeitungskette, wie sie in Abbildung 2.3 dargestellt sind, gemeint sein. Hat die Anwendung den reinen Zweck der Bildverbesserung für den Betrachter ohne folgende Analyseschritte, wird dies explizit gesagt. Desweiteren werden mit den Bezeichnungen *klassische Bildverarbeitung* oder *Standard-Methoden*, wie sie häufiger gebraucht werden, die Bereiche der Bildverarbeitung oder Klassifikation beschrieben, die keine Methoden der Computational Intelligence verwenden. *Klassifikation* soll nicht die unüberwachte Einteilung der Objekte in Klassen beschreiben, vielmehr ist auch die Kenntnis der Klasse gemeint (*Überwachte Klassifikation*), das Segment kann also einer konkreten Klasse zugeordnet werden. □

Von der eigentlichen Bildgewinnung, Abtastung[3] der Ortskoordinaten und Quantisierung[4] der Grauwerte wird hier abgesehen und auf Abschnitt 2.4 verwiesen, wo die Techniken der Bildgewinnung kurz angesprochen werden. Es wird

[3] *Abtastung* beschreibt die Diskretisierung der Bildkoordinaten.
[4] *Quantisierung* beschreibt die Diskretisierung der Helligkeitswerte.

davon ausgegangen, daß das zu verarbeitende Bild in digitaler Form vorliegt und als Matrix – wie in Definition 2.1 beschrieben – repräsentiert wird.

2.3.1 Bildvorverarbeitung

Die Aufgabe der Bildvorverarbeitung ist es, relevante Informationen im Bild hervorzuheben und bekannte Fehler zu korrigieren. Dabei ist das Ergebnis der Vorverarbeitung wieder ein Bild [LE89]. Beispiele für die Bildvorverarbeitung sind etwa das Hervorheben von Kanten, die Reduktion von Rauschen oder das Erhöhen des Kontrastes des Bildes.

Bei der Vorverarbeitung ist die zu verarbeitende Datenmenge in der Regel groß, so daß die Effizienz hier im Vordergrund steht. Es werden – abhängig von der Arbeitsweise – drei Vorverarbeitungsmethoden unterschieden:

- Punktoperationen
- Lokale Operationen
- Globale Operationen

Definition 2.6 (Punktoperationen) Bei Punktoperationen hängt der Grauwert $I'(x', y')$ des Punktes (x', y') im Ergebnisbild I' nur von genau einem Grauwert $I(x, y)$ des Ursprungsbildes I ab. □

Hierbei können sich – z. B. bei Translationen – x von x' und y von y' durchaus voneinander unterscheiden. Andere Anwendungen sind bspw. eine *Invertierung* des Bildes, eine *Kontraständerung*[5] oder weitere Operationen mittels einer LUT[6].

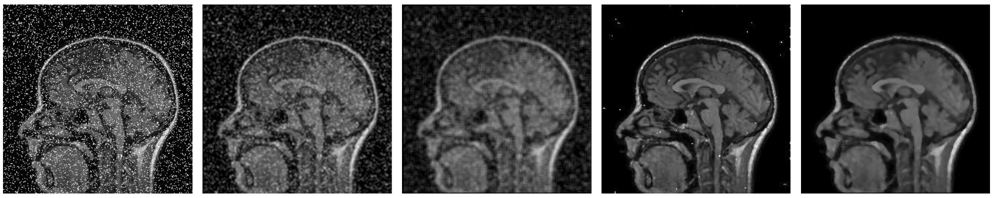
Definition 2.7 (Lokale Operationen) In lokalen Operationen ist der Grauwert $I'(x, y)$ im Ergebnisbild I' von einer wohldefinierten lokalen Umgebung um den entsprechenden Punkt (x, y) im Ursprungsbild I abhängig. □

Diese Umgebung wird meist symmetrisch zum betrachteten Punkt gewählt. Häufig ist sie quadratisch um den Punkt gelegt. Lokale Operationen finden etwa bei Faltungen (vgl. Gleichung (2.8)) zur *Kantendetektion* oder bei morphologischen Operationen Anwendung (Abschnitt 2.3.2.1).

Definition 2.8 (Globale Operationen) Bei globalen Operationen ist der Grauwert $I'(x, y)$ im Ergebnisbild O von allen Grauwerten des Ursprungsbildes I abhängig. □

[5] Der *Kontrast* eines Bildes (oder Bildsegmentes) ist definiert als die Differenz zwischen dem maximalen und dem minimalen auftretenden Grauwert [KZ95].

[6] Eine LUT (LookUpTable) ist eine Funktion von einem Grauwertbereich in einen (ggf. anderen) Grauwertbereich, das heißt $LUT : \{G_{\min}, \ldots, G_{\max}\} \to \{G'_{\min}, \ldots, G'_{\max}\}$.



(a) Verrauschtes Originalbild (b) Lineargefiltertes Bild (3×3) (c) Lineargefiltertes Bild (5×5) (d) Mediangefiltertes Bild (3×3) (e) Mediangefiltertes Bild (5×5)

Abbildung 2.4: Anwendung von Filtern zur Minderung des Rauschanteils im Bild mittels linearem Filter und Median-Filter mit jeweils einem 3×3*-Fenster und einem* 5×5*-Fenster.*

Die diskrete *Fouriertransformation* ist ein Beispiel für eine globale Operation. Hierbei wird das Bild vom *Ortsbereich* in den *Ortsfrequenzbereich* transformiert.

Als Beispiel für die Rauschminderung werden hier zwei Filter vorgestellt, die häufig eingesetzt werden. Zum einen ist das der *lineare Filter*, der eine Mittelwertbildung über ein $M_l \times N_l$ Fenster berechnet, und der nichtlineare *Median-Filter*, bei dem der Grauwert auf der mittleren Position der geordneten Folge von Umgebungsbildpunkten (ebenfalls in einem $M_l \times N_l$-Fenster) gewählt wird. Abbildung 2.4 zeigt die Anwendung beider Filter auf ein verrauschtes Bild.

Vorteil des linearen Filters ist die einfache Berechnungsmöglichkeit, allerdings wird das gefilterte Bild durch die Anwendung unschärfer, das heißt, die Glättungswirkung ist deutlicher erkennbar. Dieses Problem tritt bei dem Median-Filter nicht so deutlich bzw. erst bei größeren Filtermasken auf und es werden auch keine „neuen[7]“ Grauwerte durch Mittelwertbildung hinzugefügt. Allerdings ist der Rechenaufwand bei den Median-Filtern aufgrund der notwendigen Sortierung der Nachbarpunkte höher (in der (nicht optimierten) Implementierung (vgl. Kapitel 6) ca. ein Zeitfaktor von 2).

2.3.2 Bildsegmentierung

Der erste Schritt der *Bildanalyse* (nach der Bildvorverarbeitung) ist die *Bildsegmentierung*[8]. Hierbei wird das zu analysierende Bild in Teilbereiche zerlegt. Diese werden *Segmente* oder *Regionen*[9] genannt. Die Tiefe der Zerlegung ist pro-

[7] Hiermit sind Grauwerte gemeint, die zuvor nicht im Bildfenster enthalten sind.

[8] In der Literatur durch direkte Übernahme aus dem englischen Sprachgebrauch auch häufig als Segmentation bezeichnet.

[9] Eine Differenzierung zwischen Segment und Region wie in [KZ95] wird in dieser Arbeit nicht vorgenommen, da die Betrachtung von Löchern in Segmenten bei Bedarf explizit angegeben

blemabhängig, das heißt, daß die Zerlegung beendet werden kann, sobald das bzw. die gesuchten Objekte vom Rest des Bildes getrennt wurden [GW93]. Formal kann eine Segmentierung wie folgt definiert werden [Wah89]:

Definition 2.9 (Segmentierung) Unter Segmentierung eines diskreten Bildes $I(x, y)$ mit $\{0 \leq x \leq M-1$ und $0 \leq y \leq N-1\}$ wird die Unterteilung von I in s disjunkte, nicht-leere Teilmengen $I_1, \ldots, I_s$ verstanden, wobei ein zu definierendes Einheitlichkeitskriterium E wie folgt gilt:

1. $\bigcup_{i=1}^{s} I_i = I$

2. $\forall I_i,\ i \in \{1, \ldots, s\}$ gilt $I_i \neq \emptyset$

3. $\forall I_i, I_j,\ i, j \in \{1, \ldots, s\}, i \neq j$ gilt $I_i \cap I_j = \emptyset$

4. I_i ist zusammenhängend $\forall i \in \{1, \ldots, s\}$

5. $\forall I_i,\ i \in \{1, \ldots, s\}$ ist das Einheitlichkeitskriterium $E(I_i)$ erfüllt

6. $\forall I_i, I_j,\ i, j \in \{1, \ldots, s\},\ i \neq j$ ist das Einheitlichkeitskriterium $E(I_i \cup I_j)$ nicht erfüllt

□

Die automatische Segmentierung ist einer der schwierigsten Aufgaben in der Bildverarbeitung. Nachdem das Bild segmentiert wurde, entsprechen die Bildpunkte nicht mehr den Grauwerten des Bildes. Stattdessen ist in den Bildpunkten das Segment, zu dem dieser Bildpunkt gehört, kodiert.

Es gibt eine große Anzahl von *Segmentierungsverfahren* (vgl. [GW93, Jäh97, Hab95]), die sich neben den *Schwellenwertverfahren* jedoch generell in *kantenorientierte* und *regionenorientierte* einteilen lassen [GW93]. Welches bzw. welche der Verfahren angewendet werden können, ist abhängig von der Problemstellung und dem zu analysierenden Bildmaterial.

Die Segmentierung eines Bildes in Regionen erfolgt nach Eigenschaften, die die Regionen besitzen müssen. Die zu einer *Region* gehörenden Bildpunkte sind – bezüglich eines zuvor definierten Ähnlichkeitskriteriums – untereinander ähnlich, während sie sich von den anderen Regionen bezüglich dieses Kriteriums unterscheiden.

wird. Auch wird häufig der Begriff *Objekt* verwendet. Eine Unterscheidung zwischen Objekt und Segment wird nur dann vorgenommen, wenn dieses notwendig ist und ein Objekt aus mehreren Segmenten besteht.

2.3.2.1 Kantenorientierte Verfahren

Kantenorientierte Segmentierungsverfahren arbeiten nach dem Prinzip der Diskontinuität. Dabei wird davon ausgegangen, daß sich die Merkmalswerte von unterschiedlichen Objekten (Regionen, Segmenten) an ihrem Rand in Bezug auf das erwähnte Ähnlichkeitskriterium deutlich (was geeignet zu interpretieren und abhängig vom betrachteten Bildmaterial ist) von denen der benachbarten Regionen unterscheiden.

Häufig werden, wie bei der in Abschnitt 2.3.1 beschriebenen Bildvorverarbeitung, lokale Operationen zur *Kantendetektion* verwendet. Dazu wird für jeden Punkt (x, y) des Bildes I der Gradient (vgl. Gleichung (2.6)) bzw. dessen Betrag (vgl. Gleichung (2.7)) berechnet. Da es sich hierbei um ein diskretes Bild handelt, kann der Gradient nur approximiert werden.

$$\nabla I = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial I}{\partial x} \\ \dfrac{\partial I}{\partial y} \end{bmatrix} \tag{2.6}$$

$$|\nabla I| = \sqrt{G_x^2 + G_y^2} \tag{2.7}$$

Weil die Berechnung des Gradienten in dieser Form nur vom aktuellen Grauwert und vom Grauwert des direkten Nachbarn abhängt, ist diese Kantenberechnung sehr rauschanfällig. In der Praxis wird deshalb eine größere *Umgebung* des aktuellen Bildpunktes betrachtet, bspw. ein 3×3-Fenster. Dieses Fenster, wie es in Abbildung 2.5 dargestellt ist, wird über das gesamte Bild „geschoben“ und mit der Bildfunktion multipliziert (das heißt, es wird eine *Faltung* der Bildfunktion mit der Fensterfunktion durchgeführt).

Die *Fensterfunktion* kann unterschiedliche Werte beinhalten. Abhängig von der Größe des Fensters kann eine unterschiedliche *Glättung* erreicht werden. Der neue Funktionswert $I'(x, y)$ an der Stelle (x, y) berechnet sich nach der Faltung des Bildes I mit einem Fenster w der Größe $M_w \times N_w$ wie folgt:

$$I'(x, y) = \sum_{i=0}^{N_w - 1} \sum_{j=0}^{M_w - 1} I(x + i - \left\lfloor \frac{M_w}{2} \right\rfloor, y + j - \left\lfloor \frac{N_w}{2} \right\rfloor) \cdot w(i, j) \tag{2.8}$$

Hierbei ist zu beachten, daß am Randbereich des Bildes Probleme auftauchen können, da negative Funktionsparameter oder aber Funktionsparameter $\geq M$ bzw. $\geq N$ auftreten können. Diese Bereiche müssen besonders behandelt werden, etwa durch zyklische Erweiterung[10] des Bildes oder durch das Kopieren oder

[10] Hierbei wird die gegenüberliegende Seite des Bildes an den entsprechenden Rand kopiert. Bildlich gesprochen bedeutet dies, daß das Bild zu einem Torus geformt wird.

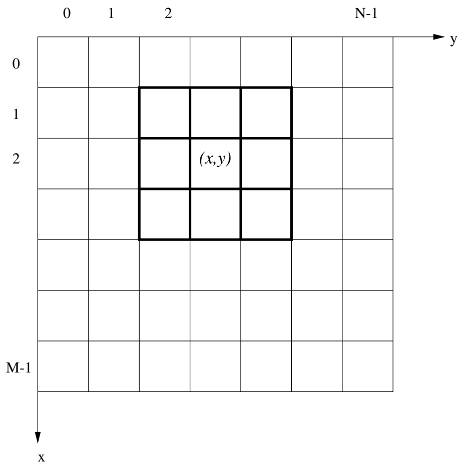
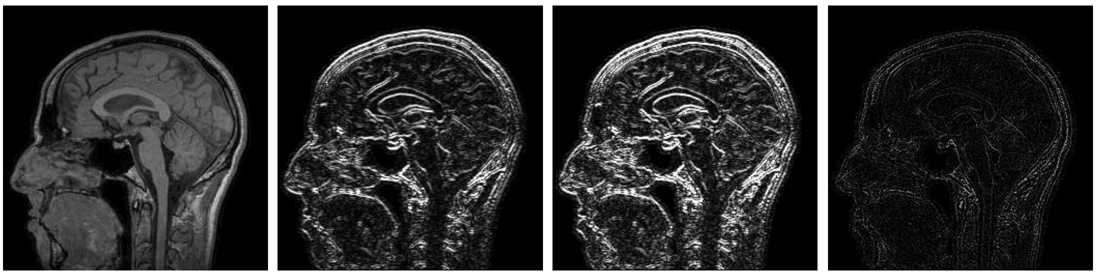


Abbildung 2.5: Schematische Darstellung eines 3×3-Fensters um den Bildpunkt (x, y), welches über das $M \times N$-Bild geschoben werden kann.



(a) Original (b) Prewitt (c) Sobel (d) Laplace

Abbildung 2.6: Einige Beispiele für die Verwendung unterschiedlicher Kantenoperatoren auf ein MRT-Bild.

Anfügen von Zeilen. Die genauere Betrachtung würde hier aber zu weit führen und ist für die weitere Arbeit nicht relevant, so daß hier lediglich auf die Literatur verwiesen wird [LE89, GW93]. Normalerweise gilt $M_w \ll M$ und $N_w \ll N$ und M_w, N_w ungerade, damit der jeweils betrachtete Punkt im Zentrum $\left(\left\lfloor \frac{M_w}{2} \right\rfloor, \left\lfloor \frac{N_w}{2} \right\rfloor\right)$ der Fenstermatrix liegt. Je nach Parameter der Maske kann die Kantenfilterung richtungsabhängig oder richtungsunabhängig sein.

Die entsprechenden Masken zur Berechnung der Kantenbilder aus Abbildung 2.6 zeigen die Abbildung 2.7 bis Abbildung 2.9.

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

(a) Horizontal

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

(b) Vertikal

Abbildung 2.7: (Richtungsabhängige) Masken für den Prewitt-*Operator.*

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

(a) Horizontal

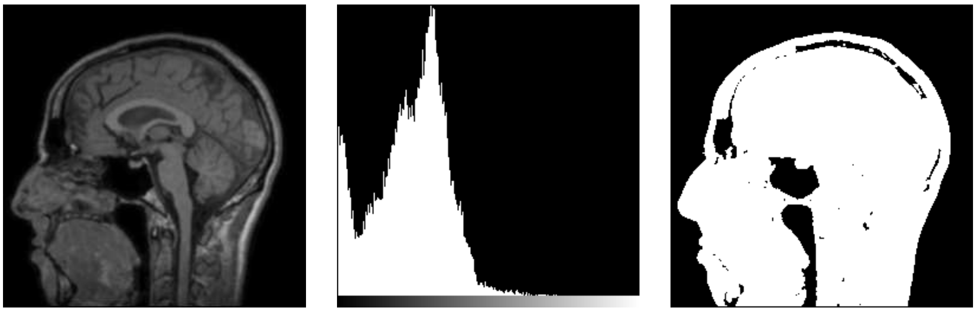
-1	0	1
-2	0	2
-1	0	1

(b) Vertikal

Abbildung 2.8: (Richtungsabhängige) Masken für den Sobel-*Operator.*

0	-1	0
-1	4	-1
0	-1	0

Abbildung 2.9: (Richtungsunabhängige) Maske für den Laplace-*Operator.*



(a) Originalbild (geglättet) (b) Histogramm (c) Binärbild

Abbildung 2.10: Bildsegmentierung mittels eines Schwellenwertes T, der im lokalen Minimum des Histogramms liegt.

2.3.2.2 Schwellenwertverfahren

Schwellenwertverfahren sind mit die wichtigsten Methoden für die Bildsegmentierung. Es gibt unterschiedliche Arten, den oder die Schwellenwerte zu bestimmen, um das gewünschte Segmentierungsergebnis zu erreichen.

$$g(x,y) = \begin{cases} 0 & \text{falls } I(x,y) \leq T, \\ 1 & \text{sonst} \end{cases} \tag{2.9}$$

Gleichung (2.9) zeigt die einfachste Variante der Binarisierung eines Bildes I mit dem Schwellenwert T[11]. Die Punkte mit dem Wert 1 werden dann dem Objekt zugeordnet, die Punkte mit dem Wert 0 gelten als Hintergrund. Ist das Objekt im Vergleich zum Hintergrund dunkel, wird die Beziehung aus Gleichung (2.9) umgekehrt. Sind mehrere Objekte mit unterschiedlichen Grauwertbereichen im Bild enthalten, können auch mehrere Schwellenwerte verwendet werden, wobei hier die Bestimmung der Schwellen T_i deutlich schwerer sein kann (vgl. [GW93]).

Abbildung 2.10 zeigt die Verwendung eines Schwellenwertes zur Segmentierung. Die Schwelle wird in einem (lokalen) Minimum des Grauwerthistogramms zwischen zwei ausgeprägten Maxima gewählt. Deren Existenz ist natürlich abhängig vom Bildinhalt.

Häufig wird aber die Schwelle T nicht aus dem globalen[12] Histogramm abgeleitet, sondern das Histogramm jeweils für eine lokale Umgebung bestimmt, woraus dann der Schwellenwert T_{lokal} abgeleitet wird.

[11] Abgeleitet von der englischen Bezeichnung *Threshold*.
[12] D.h. vom ganzen Bild abhängig (vgl. Definition 2.8).

2.3.2.3 Bereichsorientierte Verfahren

Die zuvor präsentierten Verfahren beruhten auf der Betrachtung von *Diskontinuitäten* im Bild (vgl. Abschnitt 2.3.2.1) bzw. auf der Verwendung von Schwellenwerten (vgl. Abschnitt 2.3.2.2) zur Segmentierung eines Bildes. Bereichsorientierte Verfahren dagegen basieren darauf, die gesuchten Regionen direkt zu finden.

Die Aufgabe hierbei ist es, benachbarte Punkte so zu Regionen zusammenzufassen, daß sie einem definierten Ähnlichkeitskriterium genügen und sich in diesem von den anderen Regionen unterscheiden. In dieser Form muß zunächst noch gelten, daß zwei Regionen S_1 und S_2 eines Bildes disjunkt[13] sind, d.h. $S_1 \cap S_2 = \emptyset$.

Die Ähnlichkeit kann sich durch den Grauwert, die Textur oder die Position ausdrücken. Die sogenannten *Bereichswachstumsverfahren* (auch Regionenwachstumsverfahren, Agglomerationsverfahren) benötigen zuerst *Startpunkte* (*Saatpunkte*, engl. *seed points*). Jeder Startpunkt stellt den jeweils ersten Punkt einer Region dar. Dementsprechend werden so viele Regionen erzeugt, wie Startpunkte gewählt wurden. Die Anzahl und Position der Startpunkte sind für das Ergebnis der Segmentierung des Bildes entscheidend.

Im Anschluß daran werden für jeden Startpunkt dessen Nachbarn betrachtet, wobei die Art der Nachbarschaft (vgl. Definition 2.4) entscheidend für das Segmentierungsergebnis sein kann. Die Nachbarpunkte, die das definierte Ähnlichkeitskriterium erfüllen, werden zu der aktuell betrachteten Region zugefügt und rekursiv deren Nachbarn betrachtet. Dabei ist darauf zu achten, daß das Bereichswachstum wechselnd für alle Startpunkte durchgeführt wird, denn auch davon ist das Ergebnis der Segmentierung abhängig. Das Verfahren terminiert, falls alle Bildpunkte einer Region zugeordnet wurden und eine vollständige Partitionierung des Bildes vorliegt. Dementsprechend ist das Verfahren konkurrierend, da ein einmal zu einer Region zugeordneter Bildpunkt nicht mehr anderen Regionen zugeordnet werden kann.

Die resultierenden Regionen entsprechen selten genau den Objekten im Bild, so daß durch ein *Verschmelzen* von zwei oder mehr Regionen, die zu einem einzigen Objekt gehören, das Ergebnis nachbearbeitet werden muß. Umgekehrt kann eine Region mehrere Objekte oder Teile davon enthalten, so daß die Region entsprechend aufgetrennt werden muß[14].

Zusammenfassend kann festgehalten werden, daß das Segmentierungsergebnis von folgenden Punkten abhängt:

- Anzahl der Startpunkte
- Position der Startpunkte
- Wechsel zwischen den Startpunkten

[13] Diese Forderung wird später in Abschnitt 5.2.2 aufgehoben.
[14] Diese Verfahren des Auftrennens und Verschmelzens heißen *Splitting* respektive *Merging*.

- Wahl des Ähnlichkeitskriteriums
- Wahl der Nachbarschaft

2.3.3 Segmentbeschreibung

Nach der Segmentierung liegen unterschiedliche Regionen bzw. Segmente vor. Für den anschließenden Schritt der Objekterkennung (vgl. Abschnitt 2.3.4) müssen diese Segmente beschrieben bzw. mit Eigenschaftswerten belegt werden. Dazu werden diverse Eigenschaften, die zur Erkennung des Objektes dienen können, für das Segment bestimmt, etwa die *Größe*, die *Lage*, die *Form* oder die *Textur*.

Zunächst muß dazu eine geeignete Form der *Repräsentation* der Segmente gewählt werden. Es existieren mehrere Möglichkeiten, die abhängig von der folgenden Beschreibung unterschiedliche Vor- und Nachteile besitzen.

Grundsätzlich wird unterschieden, ob externe Eigenschaften[15] des Segmentes oder interne Eigenschaften[16] betrachtet werden. Folglich eignen sich unterschiedliche Repräsentationsformen für das Segment.

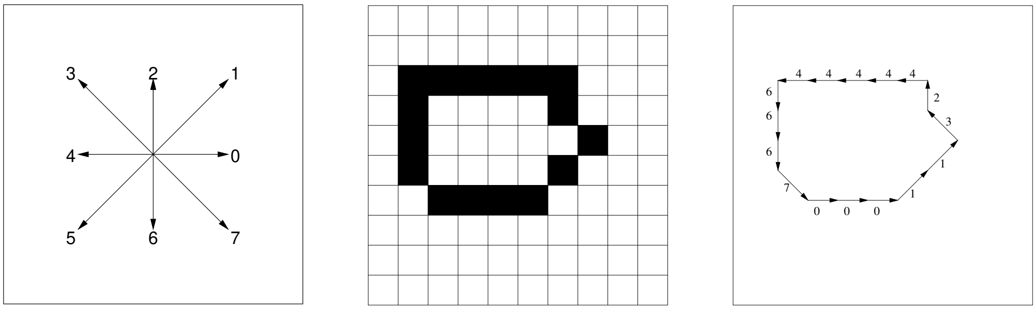
Zur Objekterkennung (vgl. Abschnitt 2.3.4) in medizinischen Bilddaten werden sowohl interne als auch externe Segmenteigenschaften benötigt, so daß hier für beide Eigenschaften geeignete Repräsentationsmöglichkeiten kurz vorgestellt werden. Für weitere Methoden sei hier wieder auf die Literatur verwiesen, etwa [GW93, Hab95]. Zudem werden weitere Eigenschaftsbeschreibungen, wie sie für Segmente zur Identifikation notwendig sind, in Abschnitt 4.2 mit den in dieser Arbeit neu entwickelten unscharfen Beschreibungsvarianten diskutiert.

2.3.3.1 Segmentdarstellung mittels Richtungsketten-Code

Externe Eigenschaften werden durch die Kontur oder Form des Segmentes festgelegt. Eine einfache und doch sehr wichtige Möglichkeit zur Konturrepräsentation stellt der *Richtungsketten-Code* (engl. *Chain code*) dar. Ausgehend von einem (beliebigen) Anfangspunkt der Kontur, die als Kette aufgefaßt werden kann, wird diese in einer definierten Richtung abgetastet. Die Richtung eines jeden folgenden Punktes (vgl. Abbildung 2.11(a)) wird in einer *Zeichenkette* (Liste) gespeichert. Falls der Startpunkt wieder erreicht wird bzw. kein neuer Randpunkt existiert, ist die gesamte Kontur erfaßt. Abbildung 2.11 zeigt ein Beispiel für die Verwendung von *Richtungsketten-Codes*.

Ein Vorteil der Richtungsketten-Codes ist die sehr kompakte Repräsentation des Segmentes. Drehungen des Segmentes um 45°-Schritte sind sehr einfach und schnell durchführbar. Ebenso leicht sind Translationen durch Neudefinition des Anfangspunktes möglich. Für eine konvexe Struktur gilt, daß ihr Ketten-Code

[15] Eigenschaften, die sich nur auf die Kontur des Segmentes beziehen.
[16] Eigenschaften, die alle Bildpunkte des Segmentes berücksichtigen.



(a) Richtungsvektoren (b) Beispielkontur (c) Richtungskette

Abbildung 2.11: Verwendung von Richtungsketten-Codes für die Segmentrepräsentation.

eine (modulo 8) monoton wachsende Folge bildet, falls die Kontur mathematisch positiv verfolgt wird, bzw. eine monoton fallende Folge in mathematisch negativer Richtung. Eine Komprimierung mittels gleicher Methoden wie bei den Lauflängen-Codes (vgl. Abschnitt 2.3.3.2) ist ebenfalls möglich. Anhand des Richtungs-Codes ist zudem eine sehr einfache Berechnung der Konturlänge unter Verwendung einer euklidischen Distanz möglich (vgl. Gleichung (2.10)).

$$\begin{aligned} Laenge &= card\{gerade\ Richtungsketten - Codes\} + \\ &\quad card\{ungerade\ Richtungsketten - Codes\} \cdot \sqrt{2} \end{aligned} \qquad (2.10)$$

Nachteilig wirkt sich sicherlich aus, daß Löcher, die in den Segmenten existieren können, gesondert behandelt werden müssen. Außerdem müssen Steuer-Codes (u. a. zur Definition des Startpunktes oder eines Loches) verwendet werden. Dies kann bspw. der (unsinnige) Richtungsketten-Code 040 oder ein zusätzliches Symbol s sein, für das $s \notin \{0, \ldots, 7\}$ gilt. Falls eine N_4-Nachbarschaft gewählt wird, reicht auch eine Betrachtung der vier Richtungsvektoren 0,2,4,6 aus Abbildung 2.11(a) aus, die dann natürlich als Richtungsvektoren 0,1,2,3 dargestellt werden können. Dann gilt allerdings die Gleichung (2.10) in dieser Form nicht mehr. Ein weiteres Problem stellen Stege mit einer Breite von Eins bzw. Verzweigungen dar.

Alternativ kann die Kontur eines Segmentes in einer Liste ihrer Randpunkte gespeichert werden. Die oben genannten Probleme von Verzweigungen oder Löchern existieren dann nicht mehr, was allerdings durch den Nachteil erhöhten Speicherplatzbedarfs und teilweise aufwendigerer Berechnung von Eigenschaften erkauft werden muß.

2.3.3.2 Segmentdarstellung mittels Lauflängen-Code

Neben der Möglichkeit, ein Segment ebenso wie das Bild in einer Matrix speichern zu können, gilt als effiziente Methode die sogenannte *Lauflängenkodierung* (engl. *Run length code*). Hierzu werden Ketten von gleichen Grauwerten abgespeichert, indem für jede Bildzeile die Startposition, der Grauwert und die zugehörige Länge der Kette gespeichert wird. Eine sehr hohe Datenreduktion wird hier bei Segmenten mit wenig wechselnden Grauwerten und homogenen Flächen erreicht. Ideal sind Binärbilder, für die jeweils nur die Startposition und Länge der zum Segment gehörenden Menge von Bildpunkten gespeichert werden muß. Weniger geeignet ist diese Repräsentation für Grauwertbilder, da Ketten gleicher Grauwerte speziell bei medizinischen Bildern meist kurz sind [Abm94].

2.3.3.3 Segmenteigenschaften

Eine Erkennung von Objekten in einem Bild, wie sie in Abschnitt 2.3.4 beschrieben wird, ist anhand ihrer Eigenschaften (Merkmale, engl. *Features*) möglich. Deshalb werden in diesem Unterabschnitt einige einfache Eigenschaften genannt, die für Bildsegmente berechenbar sind. Da die Interpretation der Eigenschaftswerte jedoch von herausragender Bedeutung ist, wird die Berechnung und Bedeutung der Zahlenwerte aber erst in Abschnitt 5.2.3 detailliert diskutiert. Betrachtete Eigenschaften sind sowohl photometrische[17] (z. B. Mittelwert, Varianz) wie geometrische Eigenschaften (Größe, Form, u. ä.). Alle Eigenschaften zusammen spannen einen Merkmalsraum auf, dessen Dimension der Anzahl der betrachteten Merkmale entspricht.

2.3.4 Objekterkennung/Interpretation

Die Objekterkennung und Bildinterpretation stehen am Schluß der Bildanalyse. *Objekterkennung* beschreibt hierbei die Bezeichnung der Bildinhalte (Segmente, Regionen) mit Namen, während die *Interpretation* die gesamte Bildszene deutet, weshalb auch häufig die Bezeichnungen *Bildverstehen* (engl. *Image understanding*) oder *Szenenanalyse* (engl. *Scene analysis*) verwendet werden. Die Objekterkennung ist somit zentraler Bestandteil für die Interpretation. Vereinfacht gesagt beruht sie auf einem Soll-/Ist-Vergleich von vorgegebenen Daten mit berechneten Daten. Die Berechnung erfolgt mit Methoden, wie sie in Abschnitt 2.3.3 beschrieben werden. Die Vorgaben stammen aus einer *Wissensbasis* (vgl. Abschnitt 3.2), in der die „*Normwerte*“ der dort beschriebenen und im Bild gesuchten Objekte enthalten sind. Hier werden einige Standardverfahren[18] beschrieben, mit denen eine Objekterkennung (*Klassifikation*) durchgeführt werden kann (vgl. [GW93, Jäh97]). Bei diesen Verfahren beruht die Zuordnung eines

[17] Auch densitometrische Eigenschaften genannt.
[18] Vergleiche hierzu Festlegung 2.5.

Segmentes zu einer Klasse auf der Positionsbestimmung des *Merkmalsvektors* im *Merkmalsraum*, die für das Segment anhand seiner Eigenschaften berechnet werden kann. Ähnliche Merkmalswerte sollten im Merkmalsraum benachbart sein, so daß sich für einzelne Klassen sogenannte *Cluster* (*Punktwolken*) bilden. In den nachfolgenden Methoden ist die Wissensbasis entsprechend direkt durch den Merkmalsraum bzw. den Clustern kodiert (unüberwachte Klassifikation). Eine Wartung durch den Domänenexperten ist somit kaum noch durchführbar.

2.3.4.1 Nachschaumethode

Diese sehr einfache Methode definiert für jeden Punkt im Merkmalsraum die Zugehörigkeit zu seiner Klasse. Dadurch bedingt muß der Definitionsbereich des Merkmalsraumes endlich sein. Ein Vorteil ist hierbei die sehr schnelle Möglichkeit zur Überprüfung, welches Segment zu welcher Klasse gehört. Dies wird durch den Nachteil erkauft, daß ein (abhängig von der Dimension des Merkmalsraums) enormer Speicherplatzbedarf vorhanden ist. Sich überlappende Merkmalswerte verschiedener Klassen sind bei dieser Klassifikationsmethode ausgeschlossen.

2.3.4.2 Quadermethode

Den Anforderungen der diskreten Wertebereiche und dem großen Speicherbedarf kann mit der Quadermethode entgegengewirkt werden, indem nicht jeder Punkt, sondern scharfe Intervalle im Merkmalsraum definiert werden. Die einzelnen Intervalle, die einen Hyperwürfel im Merkmalsraum aufspannen, entsprechen den Klassen. Nachteil ist hierbei der erhöhte Aufwand der Zugehörigkeitsbestimmung. Zudem ist die Hyperwürfelform nicht für jede Merkmalsverteilung der Klassen sinnvoll einsetzbar. Dies gilt z. B. dann, wenn die Cluster lang und schmal sind und entlang einer Diagonalen im Merkmalsraum liegen. Dann überdeckt der Hyperwürfel deutlich mehr Fläche als das Cluster. Weiterhin ist auch hier eine Überlappung der Merkmalswerte ausgeschlossen, da die Grenzen scharf gewählt werden müssen.

2.3.4.3 Methode des geringsten Abstands

Diese Methode betrachtet den Abstand des Merkmalsvektors eines zu klassifizierenden Segmentes zum Zentrum (*Schwerpunkt*) aller Cluster. Das Segment wird der Klasse zugeordnet, bei dem der euklidische Abstand zum Zentrum am geringsten ist. Ggf. kann eine Maximalentfernung definiert werden, außerhalb derer eine Klassifizierung nicht mehr durchgeführt wird. Außerdem kann eine Abstandsgewichtung abhängig von der Größe des Clusters durchgeführt werden. Hierbei muß der Abstand eines Merkmals zu einem kleinen Cluster geringer sein als bei einem großen Cluster. Der Aufwand dieses Verfahrens ist vergleichbar mit dem der Quadermethode, hat aber den Vorteil, flexibler in der Anwendung zu sein.

2.3.4.4 Methode der höchsten Wahrscheinlichkeit

Hierbei werden die Cluster als statistische Wahrscheinlichkeitsdichtefunktion modelliert. Dazu wird für jeden möglichen Merkmalsvektor die Wahrscheinlichkeit, daß er zu einer bestimmten Klasse gehört, berechnet. Entsprechend wird ein zu klassifizierender Merkmalsvektor der Klasse zugeordnet, zu der er die höchste Wahrscheinlichkeit hat. Hier wird eine (probabilistisch) unscharfe Möglichkeit gegeben, einen Merkmalsvektor unterschiedlichen Klassen zuzuweisen. Trotzdem wird von einer vollständigen Zugehörigkeit zu einer bestimmten Klasse ausgegangen.

2.4 Medizinische Bildverarbeitung

Medizinische Bildverarbeitung ist ein Thema, welches seit der Entwicklung bildgebender Verfahren immer mehr an Bedeutung gewinnt. Dies zeigt sich auch in den mittlerweile regelmäßig stattfindenden Konferenzen, Workshops und Bucherscheinungen zu diesem Thema im Inland (vgl. [LSS96, LOPR97, AMST97, LMST98, EGLM99, Han00, HL00, HHLM01]) und Ausland (vgl. bspw. [Han98, Han99, LVIF98, LVIF99]). Im folgenden werden neben den klinisch relevanten Fragestellungen auch einige bildgebende Verfahren vorgestellt.

2.4.1 Fragestellungen und bildgebende Verfahren

Bildgebende Verfahren sind aus der modernen Medizin nicht mehr wegzudenken. Dabei liefern sie unter Ausnutzung verschiedener physikalischer Eigenschaften Aussagen, die für die Diagnostik von großem Stellenwert sind. Die meisten Verfahren sind dabei in Zusammenarbeit von Ingenieuren und Medizinern entstanden.

Allerdings existieren auch unterschiedliche Aspekte bei der Beurteilung der bildgebenden Verfahren, denn nicht alles technisch Machbare steht auch für den Arzt im Vordergrund, dieser beurteilt bildgebende Verfahren nach [LOPR97]:

- der Qualität der anatomischen Darstellung von Organen und Organgrenzen
- der Detektion pathologischer Symptome
- der Differenzierung pathologischer Strukturen
- der sicheren Abgrenzung von gut- und bösartigen Prozessen
- dem Tumorstaging[19]

[19] Hierunter wird eine Malignitätsbewertung verstanden, die die Bösartigkeit des Tumors in verschiedene Klassen einordnet.

- der Belastung für den Patienten
- den Kosten der Untersuchung

Unter diesen Aspekten können die bildgebenden Verfahren deutlich voneinander unterschieden werden. Die medizinische Informatik stellt ein Bindeglied zwischen der Technik und der Medizin dar und muß als Dienstleister zwischen beiden Disziplinen fungieren. Der Informatiker muß also aus den technisch realisierbaren Bildgebungsverfahren und den für den Patienten verträglichen Methoden die für die Diagnose des Arztes optimalen Informationen aus dem Bild extrahieren.

Bei allen bildgebenden Verfahren werden physikalische Eigenschaften des aufgenommenen Materials und Gewebes ausgenutzt, um daraus für den Fachmann[20] bewertbares Bildmaterial zu erzeugen. Auch wenn diese Eigenschaften nicht bis ins Detail erklärt werden, so werden hier doch einige physikalische Grundlagen erwähnt, um später auch die Problematik der Bildgebung (vgl. Abschnitt 2.6) besser verstehen zu können.

2.4.1.1 Kernspintomographie

Die magnetische Kernspinresonanz[21] wurde im Winter 1945/46 an der Harvard Universität und zeitgleich an der Stanford Universität entdeckt. Erste Vorschläge für eine medizinische Nutzung gab es allerdings erst in den späten 50er Jahren, in den 70er Jahren wurden erste Schnittbilder an lebenden Probanden aufgenommen [Zei84].

Dabei wird folgendes Signal gemessen: Die Kernspinresonanz (engl. *Nuclear Magnetic Resonance*, *NMR*) kann durch das von einer großen Anzahl von Atomkernen kollektiv ausgesandte Hochfrequenzsignal nachgewiesen werden, wenn die mit einem *Spin*[22] und einem magnetischen Moment versehenen Atomkerne einem konstanten äußeren Magnetfeld sowie einem dazu senkrechten Hochfrequenzfeld der Kernresonanzfrequenz ausgesetzt werden. Die Signalstärke ist dabei proportional zur Anzahl der beteiligten Atomkerne und klingt nach gepulster *Hochfrequenzanregung* exponentiell mit der Zeit ab [Zei84]. Durch Variation verschiedener Meßparameter ist eine bessere Differenzierung z. B. von grauer und weißer Hirnsubstanz oder zwischen einem Tumor und gesundem Gewebe möglich. Hier zeigt sich jedoch schon ein Problem, da im Prinzip für die Aufnahme bekannt sein muß, wonach gesucht wird (vgl. Abschnitt 2.6). Eine Einführung in die Thematik der Kernspintomographie findet sich in [PJRP83, Zei84, Sie92, LOPR97].

---

[20] Diese allgemeinere Formulierung soll deutlich machen, daß auch in nicht-medizinischen Bereichen bildgebende Verfahren – etwa zur Qualitätskontrolle – eingesetzt werden.

[21] Kernspinresonanz und Magnetresonanz (MR) werden hier synonym verwendet, da die Magnetisierung durch den Kernspin verursacht wird.

[22] Eigenrotation der Teilchen.

2.4.1.2 Weitere bildgebende Verfahren

Es gibt neben der Magnetresonanztomographie viele unterschiedliche bildgebende Verfahren in der Medizin, die abhängig von verschiedenen Aspekten eingesetzt werden. Neben der eigentlichen unterschiedlichen Bildqualität bzw. generellen Anwendbarkeit müssen hier vor allem die Verfügbarkeit der Aufnahmegeräte, der Zeitbedarf, die Kosten und natürlich auch die Belastung für den Patienten berücksichtigt werden. Zu den wichtigsten bildgebenden Verfahren neben der Kernspintomographie zählen nach [LOPR97]:

- Röntgen
- Computertomographie (CT)
- Sonographie (Ultraschall)

Auf eine Erläuterung dieser weiteren Verfahren wird hier verzichtet, da dies den Rahmen der Arbeit sprengen würde und zudem das betrachtete Akquisitionsverfahren nicht relevant ist. Die Betrachtung von MRT-Bilddaten ergibt sich aus der Verfügbarkeit des vorliegenden Bildmaterials.

2.5 Existierende Bildverarbeitungswerkzeuge

Mittlerweile existieren viele Bildverarbeitungswerkzeuge, die an den Anwender unterschiedliche Anforderungen stellen. Zum einen seien hier Werkzeuge genannt, die einen Laien (in Bezug auf Bildverarbeitungsmethoden) ansprechen und eine sehr einfache Bedienung bieten. Diese sind dann häufig für genau eine spezielle Aufgabe konzipiert und müssen bei einem Aufgabenwechsel angepaßt oder neu erstellt werden. Hierbei handelt es sich eigentlich mehr um fertige Anwendungen als um Werkzeuge, wie sie hier vorgestellt werden.

Eines der bekanntesten Werkzeuge ist wohl das System *Khoros* [Kho99], welches an der Universität New Mexico entwickelt wurde und mittlerweile in der Version 2 (bzw. Khoros pro 2000) verfügbar ist. Dieses System hat einige Vorteile (aber auch Nachteile), die hier kurz vorgestellt werden.

Khoros ist eine umfangreiche Sammlung unterschiedlichster Bildverarbeitungsoperatoren, die über eine wohldefinierte Schnittstelle miteinander kombiniert werden können. Für eine einfache Bedienung existiert die Oberfläche „*Cantata*“, die eine grafische Benutzerschnittstelle bietet und somit eine leichte Entwicklung diverser Bildverarbeitungsanwendungen ermöglicht. Da die Schnittstellen offen beschrieben sind, ist dieses System auch erweiterbar. Der Vorteil der einzelnen Operatoren, die ihre Daten über Dateien austauschen, was die Erweiterung erleichtert, kann aber auch gleichzeitig als Nachteil angesehen werden, da der Kommunikationsaufwand auf diesem Wege bei großen Bilddaten, wie es etwa MRT-Datensätze darstellen (ein Datensatz ist zwischen 8 und 64 MB groß), sehr hoch ist. Ein weiterer Vorteil ist die Verfügbarkeit des Systems, d.h. es ist für fast alle UNIX-Systeme (und deren Derivate) erhältlich. Allerdings ist es nur in einer Studentenversion frei verfügbar. Der Nachteil wiederum ist hiermit auch schon genannt. Viele Anwender im medizinischen Bereich setzen aus Kostengründen und wegen der einfacheren Handhabung überwiegend Windows™-basierte Systeme ein, so daß nach erfolgreicher Erstellung einer Anwendung mit Khoros diese auf die andere Plattform[23] portiert werden muß. Daher dient die Idee des Baukastenprinzips von Khoros als Grundlage für eine eigene Entwicklung, die in Kapitel 5 konzeptionell und in Kapitel 6 als Umsetzung vorgestellt wird.

Weitere Werkzeuge sind u. a. IDL [Res98], AnalySIS [Sof98], ILabMed [SBS+99] oder CHILI [ESE+99]. Diese Werkzeuge verfolgen jedoch andere Zielsetzungen. IDL stellt bspw. eine Programmierumgebung zur Verfügung, die eine umfangreiche Bibliothek an Standardmethoden anbietet, um eine klassische Bildanalyse durchzuführen. Weiterhin liegt der Schwerpunkt auf der Visualisierung des Bildmaterials. AnalySIS ist hiermit vergleichbar. Das System CHILI ist mehr als System für Teleradiologie entwickelt worden und stellt nur einfache Analysemethoden zur Verfügung.

In keinem der Werkzeuge war es jedoch einfach möglich, eigene Datentypen zu definieren, so daß hier ein enormer Aufwand notwendig gewesen wäre, diese in der geplanten Form um CI-Methoden zu erweitern. Dadurch schließt sich eine Anwendung direkt aus und erfordert eine Eigenentwicklung.

2.6 Probleme in der medizinischen Bildverarbeitung

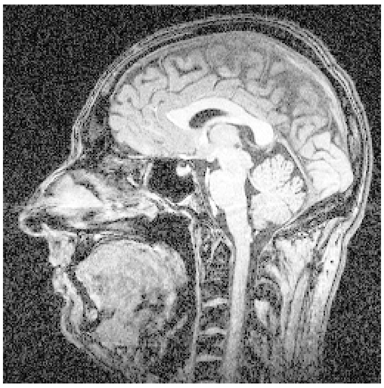
In der Bildverarbeitung existieren vielfältige Probleme, die eine *automatisierte* Analyse erschweren [LOPR97, MHM98, HFR00]. Zum einen liegt das an verschiedenen Unsicherheiten[24], zum anderen an der bei der computerbasierten Verarbeitung im Vergleich zum Menschen anderen Vorgehensweise der Bilderfassung[25]. Unter der Zielsetzung einer Diagnoseerstellung aus dem gewonnenen Bildmaterial kann auch nicht immer die technisch realisierbare Bildqualität erreicht werden, da die Belastung für den Patienten relativ zum Nutzgewinn zu hoch ist.

Die oben erwähnten Unsicherheiten können zum einen im Bildmaterial selbst begründet sein, d.h. das Bild ist z. B. aufnahmebedingt verrauscht (Abbildung 2.12(a)), beinhaltet *Bewegungsartefakte* (Abbildung 2.12(b)), Ortsunschärfen oder *Partialvolumeneffekte* (Abbildung 2.12(c)), die durch die Diskretisierung entstehen. Zumeist wird versucht, diesen Problemen schon in der Bildvorverarbeitung

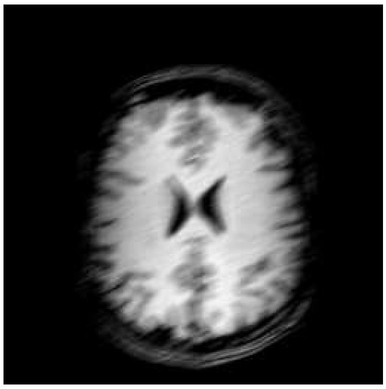
[23] Mittlerweile gibt es eine Portierung von Khoros auf Windows NT™-basierte Systeme, die allerdings nicht frei verfügbar ist.

[24] *Unsicherheit* soll an dieser Stelle ganz allgemein verstanden werden, der Begriff des unsicheren Wissens wird später noch in Abschnitt 3.2.2 erläutert.

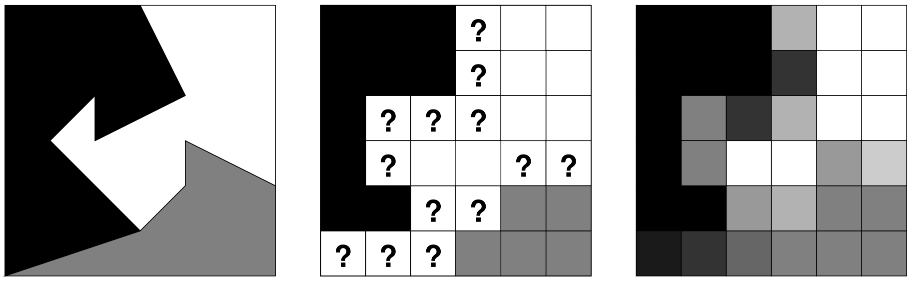
[25] "Der Mensch sieht keine Pixel" [Mei00]. Siehe dazu auch Abbildung 8.1.



(a) Bildrauschen



(b) Bewegungsartefakt



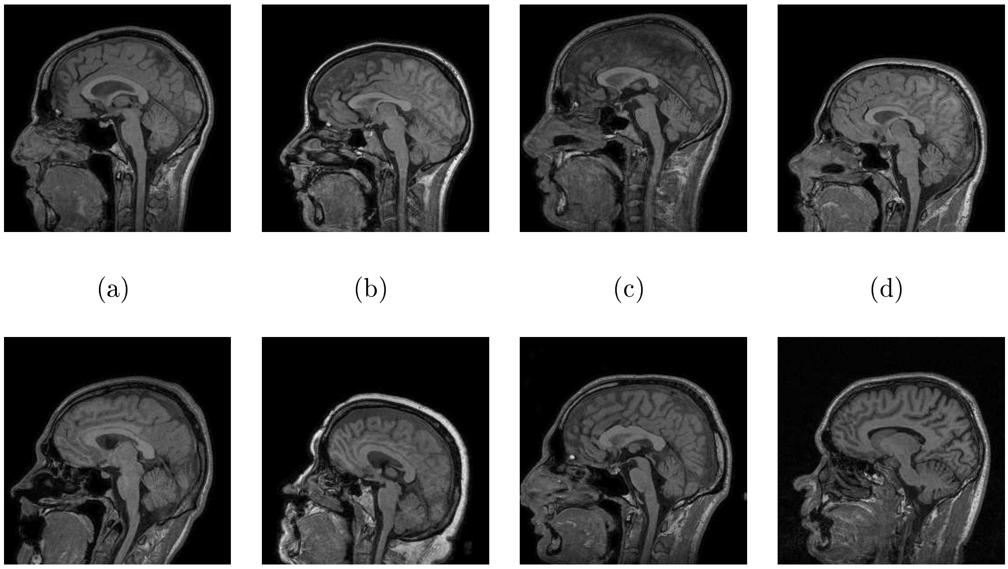
(c) Partialvolumeneffekt bedingt durch diskrete Auflösung (links reale Verteilung, mitte ungewisse Bildpunktdarstellung, rechts mögliche Darstellung

Abbildung 2.12: Darstellung einiger Probleme in der Verarbeitung digitaler medizinischer Bilddaten.

zu begegnen, allerdings nicht immer mit sehr überzeugenden Resultaten. Diese sind dadurch bedingt, das abhängig von der spezifischen Bildsequenz zusätzliches Regelwissen notwendig ist, welches in den Standardoperatoren nicht kodiert werden kann.

Prinzipiell kann auch das Wissen über die Bilddaten unsicher (im Sinne von *vage*) oder unscharf[26] sein [Spi93], bspw. über den Bildinhalt generell oder über Eigenschaften der Bildinhalte. Hier sei etwa die sogenannte *interindividuelle Variabilität* (vgl. Abbildung 2.13) genannt, d.h. die unterschiedliche Ausprägung von Objekten der gleichen Klasse, die besonders im medizinischen Bilddaten-Bereich existent ist und neben der Segmentierung auch häufig die Klassifikation der einzelnen Segmente erschwert. Allerdings lassen sich derartige Bildsequenzen recht gut umgangssprachlich beschreiben. Eine entsprechende Modellierung ist jedoch mit den Standardbildverarbeitungsoperatoren zur Bildbeschreibung und -

[26] Hierbei ist keine optische Unschärfe gemeint, sondern eine logische.



(a) (b) (c) (d)

(e) (f) (g) (h)

Abbildung 2.13: Darstellung der (interindividuellen) Variabilitäten bei medizinischen MRT-Kopfdatensätzen. Objekte gleicher Klasse unterscheiden sich deutlich in ihrem Erscheinungsbild. Dargestellt ist jeweils die mittlere sagittale Schicht.

interpretation kaum zu fassen. Sie benötigt vielmehr eine Theorie des unscharfen Wissens, anhand derer eine geeignete Basis für neue Bildverarbeitungsoperatoren geschaffen werden kann. In Abschnitt 3.2.2 wird der Begriff des unsicheren Wissens in einem ersten Schritt näher erläutert.

Die genannten und weitere Probleme werden teilweise ausführlicher in den folgenden Teilen der Arbeit auftauchen und bei Bedarf näher erläutert. Abbildung 2.14 zeigt nochmals die möglichen Unsicherheiten und ihre Position im Bildverarbeitungsprozeß auf.

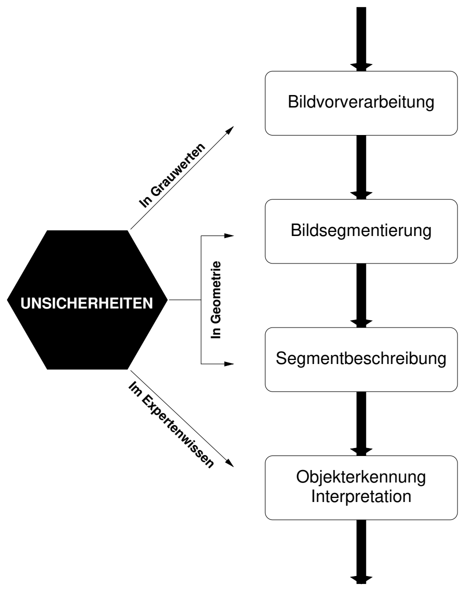


Abbildung 2.14: Arten möglicher Unsicherheiten in der digitalen Bildverarbeitung und ihre Einordnung in die Prozeßkette.

3 Computational Intelligence und Wissensrepräsentation

3.1 Grundlagen der Computational Intelligence

Im vorangegangenen Kapitel wurden grundlegende Methoden der Bildverarbeitung eingeführt und zudem skizziert, welche aktuellen Probleme diese noch in der medizinischen Bildverarbeitung aufwerfen. Um diese Probleme besser in den Griff zu bekommen, werden letztendlich Bildverarbeitungsalgorithmen um CI-Methoden ergänzt. Folglich ist es notwendig, die dafür prinzipiell in Frage kommenden Aspekte der *Computational Intelligence* einzuführen.

Zum gleichen Verständnis der verwendeten Einzelmethoden und ihrer Komposition, werden diese hier vorgestellt und zudem zentrale – in der Arbeit verwendete – Begriffe definiert. Auch dieses Kapitel ist kurz gefaßt und beinhaltet hauptsächlich Definitionen der im weiteren Verlauf der Arbeit notwendigen Terminologie. An den entsprechenden Stellen wird auf weiterführende Literatur verwiesen.

3.1.1 Der Begriff der Computational Intelligence

Der Begriff *Computational Intelligence* wurde 1994 von Bezdek [Bez94] erstmals als Zusammenfassung der hier genannten Methoden verwendet. Danach wird unter *Computational Intelligence* die Verwendung von Fuzzy-Logik, künstlichen neuronalen Netzen und evolutionären Algorithmen verstanden. Deshalb gilt in der weiteren Arbeit die folgende Festlegung 3.1, da eine exakte Abgrenzung nicht immer genau vorgenommen wird. Bspw. wird in [Bez94] die Verwendung von Expertenwissen für CI-Methoden ausgeschlossen, was hier nicht der Fall ist, sondern ausdrücklich zugelassen wird.

Häufig synonym wird die Bezeichnung *Softcomputing* verwendet. Begrifflich kann die CI vom Softcomputing dadurch abgegrenzt werden, daß sich letztere eher mit der theoretischen Untersuchung der genannten Methoden beschäftigt, während die CI die Anwendung der Methoden in den Vordergrund stellt. Zur *künstlichen Intelligenz* (KI) läßt sich die CI dadurch unterscheiden, daß Daten mehr auf Zahlenebene verarbeitet werden, während die KI eher symbolische Daten verarbeitet. Leider ist hier aber keine genaue Abgrenzung oder Definition zu finden, in mancher Literatur werden alle drei Begriffe synonym verwendet.

Festlegung 3.1 (CI-Methoden) Wenn in dieser Arbeit von CI-Methoden gesprochen wird, so sind immer die in diesem Kapitel vorgestellten Methoden in allgemeiner Form gemeint. □

Festlegung 3.2 (Verwendung von $\boldsymbol{\mu}$) Das Symbol μ wird im Kontext von Fuzzy-Methoden für die Zugehörigkeitsfunktion verwendet, während es bei evolutionären Algorithmen die Anzahl der Elternindividuen beschreibt. Um diese allgemein übliche Notation beizubehalten und trotzdem eine Verwechslung zu vermeiden, wird für Zugehörigkeitsfunktionen das Symbol μ und bei den Evolutionsstrategien das Symbol $\hat{\mu}$ verwendet. □

3.1.2 Begriffsbestimmung für Fuzzy-Methoden

Mit dem in der Informatikwelt wohl eines der am häufigsten zitierten Artikel [Zad65] hat Lotfi A. Zadeh den Grundstein für die einfache Modellierung vieler Probleme gelegt, für die eine exakte Formulierung nicht möglich oder aber zu teuer oder aufwendig ist.

Eine Einführung in die Thematik von Fuzzy-Methoden finden sich etwa in [BG93, Bie97, Thi97]. Um begriffliche Klarheit und Einheitlichkeit zu erhalten, werden die in der Arbeit verwendeten Begriffe an dieser Stelle definiert.

Definition 3.1 (Fuzzy-Menge) Eine unscharfe Menge (Fuzzy-Menge) $\mathcal{A}$ wird durch eine Funktion μ von einer Grundmenge $\mathbb{U}$ in das reelle Einheitsintervall $\langle 0, 1\rangle$ charakterisiert. Diese Funktion wird Zugehörigkeitsfunktion genannt.

$$\mu_{\mathcal{A}} : \mathbb{U} \to \langle 0, 1\rangle \tag{3.1}$$

□

Der Wert $\mu_{\mathcal{A}}(x)$ gibt den Grad an, mit dem das Element $x \in \mathbb{U}$ zur Fuzzy-Menge $\mathcal{A}$ gehört, und kann als Quasiwahrheitswert der Aussage „x ist Element von $\mathcal{A}$“ in der entsprechenden mehrwertigen Logik interpretiert werden [Pol97].

Soll eine spezielle unscharfe Menge $\mathcal{A}$ über den Grundbereich $\mathbb{U}$ beschrieben werden, dann wird ihre Zugehörigkeitsfunktion $\mu_{\mathcal{A}}$ angegeben – und zwar entweder durch einen Funktionsausdruck, eine Wertetabelle oder den Graphen von $\mu_{\mathcal{A}}$. Ansonsten wird – wie in der Literatur allgemein üblich – nicht zwischen $\mathcal{A}$ und $\mu_{\mathcal{A}}$ unterschieden. Für ein diskretes, endliches Universum $\mathbb{U}$ kann die Menge $\mathcal{A}$ auch als eine Menge von Paaren geschrieben werden:

$$\mathcal{A} = \{(u, \mu_{\mathcal{A}}(u)) \mid u \in \mathbb{U}\} \tag{3.2}$$

Eine alternative Notation[1] bei diskreten, endlichen Universen[2] ist die Bruchschreibweise, die im Zähler den Zugehörigkeitsgrad des Elementes im Nenner angibt:

$$\mathcal{A} = \frac{\mu_{\mathcal{A}}(u_1)}{u_1} + \cdots + \frac{\mu_{\mathcal{A}}(u_n)}{u_n} = \sum_{i=1}^{n} \frac{\mu_{\mathcal{A}}(u_i)}{u_i} \tag{3.3}$$

Definition 3.2 (Singleton) Ein Singleton ist eine Fuzzy-Menge $\mathcal{A}$, für die gilt:

$$\exists u_i \in \mathbb{U} : \mu_{\mathcal{A}}(u_i) > 0, \forall u_j \in \mathbb{U},\ u_j \neq u_i : \mu_{\mathcal{A}}(u_i) = 0 \tag{3.4}$$

$\square$

Definition 3.3 (Höhe) Die Höhe $height(\mathcal{A})$ einer unscharfen Menge $\mathcal{A}$ ist das Supremum der Zugehörigkeitswerte.

$$height(\mathcal{A}) \stackrel{\text{def}}{=} \sup_{u \in \mathbb{U}} \mu_{\mathcal{A}}(u) \tag{3.5}$$

$\square$

Definition 3.4 (Normale Fuzzy-Menge) Eine unscharfe Menge $\mathcal{A}$ heißt normalisiert (oder normal), wenn ihre Höhe gleich Eins ist, d.h. $height(\mathcal{A}) = 1$. $\square$

Definition 3.5 (Träger) Der Träger $support(\mathcal{A})$ einer unscharfen Menge $\mathcal{A}$ über dem Universum $\mathbb{U}$ ist die Menge aller Elemente, für welche die Zugehörigkeitsfunktion größer Null ist.

$$support(\mathcal{A}) \stackrel{\text{def}}{=} \{u \in \mathbb{U} \mid \mu_{\mathcal{A}}(u) > 0\} \tag{3.6}$$

$\square$

Definition 3.6 (Kern) Der Kern $core(\mathcal{A})$ einer unscharfen Menge $\mathcal{A}$ über dem Universum $\mathbb{U}$ ist die Menge aller Elemente, für welche die Zugehörigkeitsfunktion gleich Eins ist.

$$core(\mathcal{A}) \stackrel{\text{def}}{=} \{u \in \mathbb{U} \mid \mu_{\mathcal{A}}(u) = 1\} \tag{3.7}$$

$\square$

Definition 3.7 (Konvexität) Eine unscharfe Menge $\mathcal{A}$ heißt konvex, wenn für alle Elemente $u_1, u_2 \in \mathbb{U}$ und $\lambda \in \langle 0, 1\rangle$ gilt:

$$\mu_{\mathcal{A}}(\lambda \cdot u_1 + (1 - \lambda) \cdot u_2) \geq \min(\mu_{\mathcal{A}}(u_1), \mu_{\mathcal{A}}(u_2)) \tag{3.8}$$

$\square$

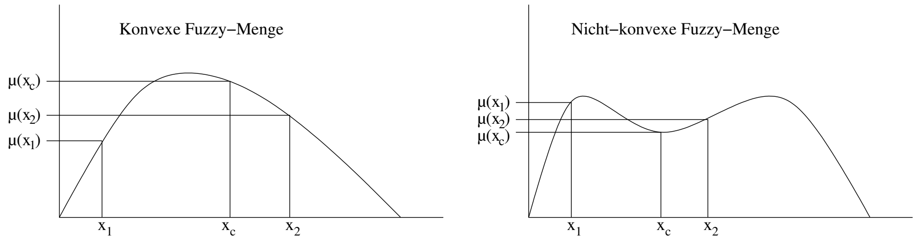


Abbildung 3.1: Beispiel für konvexe und nicht-konvexe Fuzzy-Mengen.

Abbildung 3.1 zeigt jeweils ein Beispiel für eine konvexe und eine nicht konvexe Fuzzy-Menge. Hierbei soll $x_c = \lambda \cdot u_1 + (1 - \lambda) \cdot u_2$ entsprechend Definition 3.7 gelten.

Definition 3.8 (α-Schnitt) Der α-Schnitt einer unscharfen Menge $\mathcal{A}$ über $\mathbb{U}$ mit $\alpha \in \langle 0, 1 \rangle$ ist die Menge aller Elemente aus $\mathcal{A}$, deren Zugehörigkeitsgrad größer oder gleich dem Schwellenwert α ist:

$$A^{\geq \alpha} \stackrel{\text{def}}{=} \{u \in \mathbb{U} \mid \mu_{\mathcal{A}}(u) \geq \alpha,\ \alpha \in \langle 0, 1 \rangle\} \tag{3.9}$$

Von einem *scharfen* α-Schnitt wird gesprochen, wenn folgende Schreibweise verwendet wird[3]:

$$A^{> \alpha} \stackrel{\text{def}}{=} \{u \in \mathbb{U} \mid \mu_{\mathcal{A}}(u) > \alpha,\ \alpha \in \langle 0, 1 \rangle\} \tag{3.10}$$

□

Definition 3.9 (Zylindrische Erweiterung) Eine eindimensionale unscharfe Menge $\mathcal{A}$ über dem Universum $\mathbb{U}_x$ mit der entsprechenden Zugehörigkeitsfunktion $\mu_{\mathcal{A}}$ kann wie folgt zu einer zweidimensionalen unscharfen Menge $\mathcal{A}_{2D_x}$ über dem Universum $\mathbb{U}_x \times \mathbb{U}_y$ mit $\mu_{\mathcal{A}_{2D_x}}$ erweitert werden. Analog kann eine eindimensionale unscharfe Menge $\mathcal{B}$ über dem Universum $\mathbb{U}_y$ mit der entsprechenden Zugehörigkeitsfunktion $\mu_{\mathcal{B}}$ wie folgt zu einer zweidimensionalen unscharfen Menge $\mathcal{B}_{2D_y}$ über dem Universum $\mathbb{U}_x \times \mathbb{U}_y$ mit $\mu_{\mathcal{B}_{2D_y}}$ erweitert werden.

$$\mathcal{A}_{2D_x} = \{((u_x, u_y), \mu_{\mathcal{A}}(u_x)) \mid (u_x, u_y) \in \mathbb{U}_x \times \mathbb{U}_y\} \tag{3.11}$$

$$\mathcal{B}_{2D_y} = \{((u_x, u_y), \mu_{\mathcal{B}}(u_y)) \mid (u_x, u_y) \in \mathbb{U}_x \times \mathbb{U}_y\} \tag{3.12}$$

□

[1] In dieser Arbeit wird die funktionale oder die Paar-Schreibweise verwendet, die alternative Notation gilt lediglich als Ergänzung, da sie ebenfalls häufig verwendet wird.

[2] Auch bei unendlichen Universen ist diese Schreibweise möglich, wobei das Summenzeichen durch ein Integralzeichen ersetzt wird.

[3] Hier ist die Literatur nicht einheitlich, es finden sich auch umgekehrte Benennungen (vgl. [BG93, Bie97])

In [Zad75a, Zad75b, Zad75c] führt Zadeh das Konzept der linguistischen Variablen ein, die dort durch ein 5-Tupel $(X, T(X), \mathbb{U}, G, M)$ charakterisiert wird. Dabei ist X der Bezeichner (Name) der Variablen, $T(X)$ ist die Menge der linguistischen Terme, die wiederum Bezeichner für Fuzzy-Mengen über dem Universum $\mathbb{U}$ sind. G ist eine Menge syntaktischer Regeln zur Erzeugung der Termnamen und M eine Menge semantischer Regeln, die jedem Term eine Fuzzy-Menge über $\mathbb{U}$ als Bedeutung zuweist. Die Regeln für die Erzeugung der Termnamen und die Zuweisung der Namen zu den Mengen werden hier nicht gebraucht, da die Termnamen immer mit den Mengen identifiziert werden. Deshalb wird folgende verkürzte Definition verwendet.

Definition 3.10 (Linguistische Variable, Linguistischer Term) Ein linguistischer Term ist ein Bezeichner für eine Fuzzy-Menge. Eine linguistische Variable wird durch ein Tupel $(X, T, \mathbb{U})$ charakterisiert, wobei X der Bezeichner (Name) der Variablen ist und T die Menge der linguistischen Terme, deren Fuzzy-Mengen über dem Universum $\mathbb{U}$ definiert sind. □

Definition 3.11 (T-Norm) Eine T-Norm ist eine Funktion $\tau : \langle 0,1\rangle^2 \to \langle 0,1\rangle$, die für alle $x, y, z, x_1, x_2, y_1, y_2 \in \langle 0,1\rangle$ folgende Bedingungen erfüllt:

1. $\tau(x, 1) = x$ (neutrales Element)
2. Wenn $x_1 \leq x_2$ und $y_1 \leq y_2$, dann $\tau(x_1, y_1) \leq \tau(x_2, y_2)$ (Monotonie)
3. $\tau(x, y) = \tau(y, x)$ (Kommutativität)
4. $\tau(x, \tau(y, z)) = \tau(\tau(x, y), z)$ (Assoziativität) □

Wegen 1 und 2 gilt auch: $\tau(0, x) = 0$, da $\tau(0, x) \leq \tau(0, 1) = 0 \ \forall x \in \langle 0,1\rangle$.

Definition 3.12 (S-Norm) Eine S-Norm (auch T-Conorm) ist eine Funktion $\sigma : \langle 0,1\rangle^2 \to \langle 0,1\rangle$, die für alle $x, y, z, x_1, x_2, y_1, y_2 \in \langle 0,1\rangle$ folgende Bedingungen erfüllt:

1. $\sigma(x, 0) = x$ (neutrales Element)
2. Wenn $x_1 \leq x_2$ und $y_1 \leq y_2$, dann $\sigma(x_1, y_1) \leq \sigma(x_2, y_2)$ (Monotonie)
3. $\sigma(x, y) = \sigma(y, x)$ (Kommutativität)
4. $\sigma(x, \sigma(y, z)) = \sigma(\sigma(x, y), z)$ (Assoziativität) □

Wegen 1 und 2 gilt auch: $\sigma(1, x) = 1$, da $1 = \sigma(1, 0) \leq \sigma(1, x) \ \forall x \in \langle 0,1\rangle$.

Definition 3.13 (Negation) Eine Negation ist eine Funktion $\nu : \langle 0,1\rangle \to \langle 0,1\rangle$, die für alle $x, y \in \langle 0,1\rangle$ folgende Bedingungen erfüllt:

1. $\nu(0) = 1$ und $\nu(1) = 0$
2. Wenn $x \leq y$, dann $\nu(x) \geq \nu(y)$ (Comonotonie)

Gilt auch $\nu(\nu(x)) = x$, so heißt die Negation *stark*. □

Definition 3.14 (T-Quantor) Ein T-Quantor TQ_τ ist eine durch die Assoziativität und Kommutativität erlaubte Erweiterung einer T-Norm τ auf eine n-stellige Eingabe mit $n \in \mathbb{N}^{>0}$:

$$\tau_1(x_1) = x_1 \tag{3.13}$$
$$\tau_{n+1}(x_1, \ldots, x_{n+1}) = \tau(\tau_n(x_1, \ldots, x_n), x_{n+1}) \tag{3.14}$$
$$TQ_\tau(x_1, \ldots, x_n) = \tau_n(x_1, \ldots, x_n) \tag{3.15}$$
$$TQ_\tau(X) = TQ_\tau(x_1, \ldots, x_n) \text{ mit } X = \{x_1, \ldots, x_n\} \tag{3.16}$$

□

Definition 3.15 (S-Quantor) Ein S-Quantor SQ_σ ist eine durch die Assoziativität und Kommutativität erlaubte Erweiterung einer S-Norm σ auf eine n-stellige Eingabe mit $n \in \mathbb{N}^{>0}$:

$$\sigma_1(x_1) = x_1 \tag{3.17}$$
$$\sigma_{n+1}(x_1, \ldots, x_{n+1}) = \sigma(\sigma_n(x_1, \ldots, x_n), x_{n+1}) \tag{3.18}$$
$$SQ_\sigma(x_1, \ldots, x_n) = \sigma_n(x_1, \ldots, x_n) \tag{3.19}$$
$$SQ_\sigma(X) = SQ_\sigma(x_1, \ldots, x_n) \text{ mit } X = \{x_1, \ldots, x_n\} \tag{3.20}$$

□

Beispiele für linguitische Variablen sind Eigenschaften betrachteter Objekte. Z. B. ist die Eigenschaft *Größe* eine linguistische Variable, deren Werte linguistische Terme sind, bspw. *klein*, *mittel*, *groß*. Die Interpretation der Terme erfolgt durch unscharfe Mengen. Eine sehr gebräuchliche T-Norm ist die min-Funktion, die S-Norm die max-Funktion, als Negation wird häufig die Funktion $\nu(x) = 1-x$ verwendet. Diese und weitere Beispiele finden sich etwa in [Bie97, Thi97].

Definition 3.16 (Unscharfe Relation) Eine unscharfe Relation (oder Fuzzy-Relation) $\mathcal{S}$ wird charakterisiert durch eine Funktion μ von den Grundmengen $\mathbb{U}_1, \ldots, \mathbb{U}_n$ in das reelle Einheitsintervall $\langle 0, 1\rangle$.

$$\mu_{\mathcal{S}} : \mathbb{U}_1 \times \cdots \times \mathbb{U}_n \to \langle 0, 1\rangle \tag{3.21}$$

□

Definition 3.17 (Ikonische Fuzzy-Menge) Ikonische Fuzzy-Mengen sind zweidimensionale Fuzzy-Mengen, die über der Menge der Bildpunktkoordinaten definiert sind. Funktionswerte werden aber nicht als Grauwerte[4] sondern als Zugehörigkeitswerte interpretiert.

$$\mu : \{0, \ldots, M-1\} \times \{0, \ldots, N-1\} \to \langle 0, 1 \rangle \tag{3.22}$$

□

Im Gegensatz zur unscharfen Interpretation von Bildern, wie sie in Abschnitt 4.2 verwendet wird und wo zum Grauwert $I(x, y)$ die Zugehörigkeit $\mu_I(x, y)$ angegeben wird, werden bei ikonischen Fuzzy-Mengen die Grauwerte selbst als Zugehörigkeitswerte interpretiert. Eine Beispiel für eine ikonischen Fuzzy-Menge ist die zylindrische Erweiterung, mathematisch handelt es hierbei um 2-stellige Fuzzy-Relationen.

3.1.3 Grundlagen künstlicher neuronaler Netze

Künstliche neuronale Netze (*KNN*), im folgenden kurz neuronale Netze genannt[5], wurden in den letzten Jahren in vielfältigster Weise eingesetzt und haben u. a. im Bereich der Mustererkennung und Klassifikation sehr gute Ergebnisse erreicht. Eine weitere Anwendung ist etwa die Repräsentation von mathematischen Funktionen [DG97].

Der große Nutzen neuronaler Netze liegt darin, daß sie lernfähig[6] sind. In der Regel kann zwischen einer Trainingsphase und einer Anwendungsphase differenziert werden. Während in der Trainingsphase das Netz lernt, findet in der Anwendungsphase keine Änderung des Netzes mehr statt. In der Trainingsphase werden mehrere Formen des Lernens unterschieden [Zel94]:

- **Überwachtes Lernen**: Hierbei gibt ein externer „Lehrer" zu jedem Eingabevektor die gewünschte Sollausgabe vor. Ziel bei wiederholter Präsentation der Trainingsdaten ist die Anpassung des Netzes derart, daß es die Assoziation zwischen Eingabemuster und Ausgabemuster selbständig vornehmen kann und auch für unbekannte, ähnliche Eingabemuster eine (gewünschte) Ausgabe liefert. Diese Fähigkeit des Netzes wird *Generalisierung* genannt. Die bekannteste Lernregel ist das Gradientenabstiegsverfahren, welches in der Literatur mit Backpropagation bezeichnet wird [Zel94].

[4] Unabhängig davon sind Bildoperationen auch auf ikonische Fuzzy-Mengen anwendbar (vgl. z. B. Abschnitt 5.2.3).

[5] Auf die Spezifizierung „künstlich" kann hier problemlos verzichtet werden, da auf „biologische" neuronale Netze aus Platzgründen nicht eingegangen wird. Die Motivation aus der Biologie ist in [RMS91, Zel94] ausführlich beschrieben. Ebenso werden die Begriffe „Netz" und „Netzwerk" synonym verwendet.

[6] Lernfähig im Sinne von selbstadaptiv.

- **Verstärkendes Lernen**: Ähnlich wie beim überwachten Lernen existiert auch hier ein externer Lehrer. Dieser gibt jedoch nicht die Sollausgabe vor, sondern teilt dem Netz nur mit, ob es richtig oder falsch klassifiziert hat. Nachteil ist, daß diese Form deutlich langsamer ist. Allerdings ist es biologisch plausibler.

- **Unüberwachtes Lernen**: Hierbei existiert kein Lehrer, das Lernen geschieht durch Selbstorganisation, indem ähnliche Eingaben in ähnliche Kategorien klassifiziert werden. Das bekannteste Beispiel für unüberwachtes Lernen sind die selbstorganisierenden Karten von Kohonen, die bekannteste Lernregel die Hebbsche Lernregel.

Neuronale Netze bestehen im allgemeinen aus folgenden Komponenten:

1. **Zellen**: Zellen oder auch Neuronen (vgl. Abbildung 3.2(a)) stellen die kleinste Einheit eines neuronalen Netzes dar. Sie besitzen eine Aktivierungsfunktion f_{act}, die die Berechnung eines Aktivierungswertes (oder einfach auch nur *Aktivierung*) a_j aus der Netzeingabe net_j (Gleichung (3.25)) berechnet:

$$a_j = f_{act}\left(net_j\right) \tag{3.23}$$

   Schließlich besitzt die Zelle noch eine Ausgabefunktion f_{out}, welche die Ausgabe o_j der Zelle j aus der Aktivierung a_j berechnet[7]:

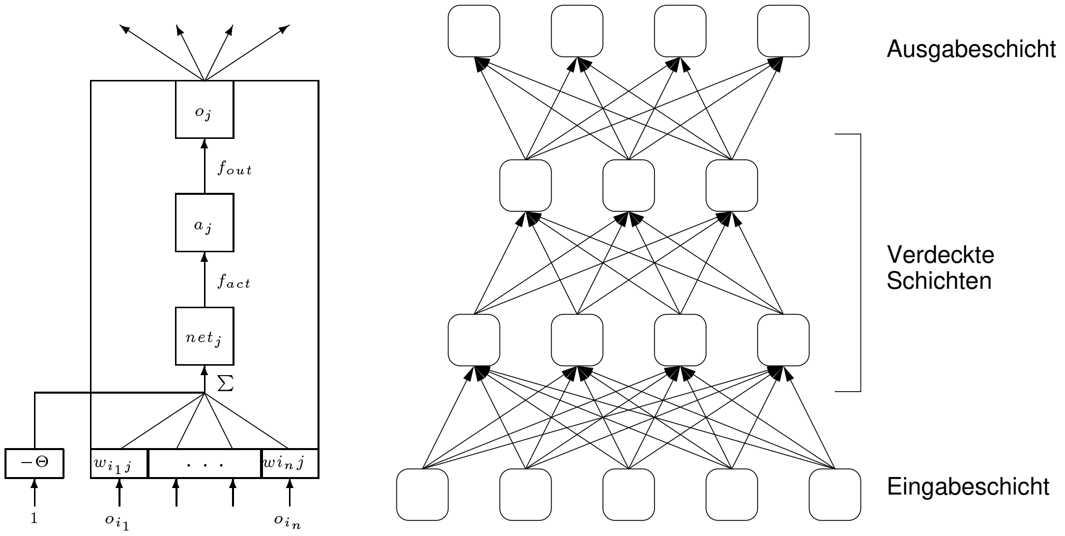
$$o_j = f_{out}\left(a_j\right) \tag{3.24}$$

2. **Verbindungsnetzwerk**: Ein neuronales Netz kann als gerichteter, gewichteter Graph angesehen werden. Hierbei stellen die Kanten die gewichteten Verbindungen zwischen den Neuronen dar. Das Gewicht der Verbindung von Zelle i nach Zelle j wird mit w_{ij}, die Matrix der Verbindungen aller Zellen (Gewichtsmatrix) mit W bezeichnet.

3. **Propagierungsfunktion**: Sie gibt an, wie sich die Netzeingabe eines Neurons aus den Ausgaben der anderen Neuronen und den Verbindungsgewichten berechnet. Die Netzeingabe net_j berechnet sich aus den n_j Eingaben und dem *Bias* Θ_j wie folgt:

$$net_j = \left(\sum_{i=1}^{n_j} o_i \cdot w_{ij}\right) - \Theta_j \tag{3.25}$$

[7]Für diese wird in der Regel die Identitätsfunktion gewählt, das heißt $f_{out}(x) = x$.



(a) Einzelnes Neuron (b) Neuronales Netz

Abbildung 3.2: Darstellung eines einzelnen Neurons [NKK94] und eines dreischichtigen, vorwärtsgerichteten Netzes, das vollständig schichtweise verbunden ist.

4. **Lernregel**: Eine Lernregel ist ein Algorithmus, gemäß dem ein neuronales Netz lernt, für eine Eingabe eine gewünschte Ausgabe zu berechnen. Dabei erfolgt das Lernen häufig durch Modifikation der Gewichte. Falls eine Sollausgabe für die Eingabedaten vorliegt, wird versucht, den Fehler zwischen Sollausgabe und Istausgabe zu minimieren.

Die Zellen können nach ihrer Position im neuronalen Netz in Eingabeneuronen, verdeckte Neuronen und Ausgabeneuronen differenziert werden. Entsprechend befinden sich die Eingabeneuronen auf der Eingabeschicht, verdeckte Neuronen auf der bzw. den 0 bis n verdeckten Schichten und die Ausgabeneuronen auf der Ausgabeschicht (vgl. Abbildung 3.2(b)). Zellen der Eingabeschicht leiten die Eingabe in das Netz weiter, Zellen der mittleren Schichten dienen ebenso wie die Ausgabeschicht der Informationsverarbeitung, wobei die Ausgabeschicht schließlich die Ausgabe des Netzes nach außen gibt. Die Anzahl der Schichten, die ein Netz aufweist, wird in der Literatur nicht einheitlich verwendet. Angelehnt an [Zel94] wird hier die Zahl trainierbarer Schichten verwendet, d.h. die Eingabeschicht wird nicht mitgezählt, so daß ein Netz mit $n+1$ Schichten (einschließlich Eingabeschicht) als n-schichtiges Netz bezeichnet wird.

Seit der Entwicklung der ersten Netz-Modelle, den sogenannten Perzeptrons, im Jahre 1958 durch Rosenblatt [Ros58], wurden viele verschiedene Arten von Netzwerktypen bezüglich der Topologien, Lernverfahren und Aktivierungs- bzw.

Ausgabefunktionen vorgestellt. Eine umfangreiche Übersicht findet sich etwa in [Zel94, Pat97]. Genau dabei liegt auch das Problem neuronaler Netze. Es gibt keine allgemeingültige Methode, die für eine gegebene Problemstellung Aussagen über die Topologie des zu verwendenden Netzes, der Anzahl von Neuronen je Schicht, der Art der Aktivierungsfunktion oder über die zu verwendenden Lernregeln macht. Für die Qualität des Netzes ist somit viel Erfahrung des „Netz-Designers“ gefragt. Allerdings gibt es auch Ansätze, diese Design-Arbeit mittels evolutionärer Algorithmen zu lösen [SHF94, Hei97].

Hier werden nach dieser grundsätzlichen Beschreibung eines neuronales Netzes die Typen vorgestellt, die in der weiteren Arbeit verwendet werden. Die Auswahl der Typen beruht dabei auf der Tatsache, daß sie einfach anzuwenden sind und gute Ergebnisse für die jeweiligen Aufgaben geliefert haben.

3.1.3.1 Vorwärtsgerichtete Backpropagation-Netze

Unter vorwärtsgerichteten Backpropagation-Netzen sind Netze zu verstehen, deren Graphen zyklenfrei sind, d.h., daß keine Verbindung von einer Zelle i zu einer Zelle j mit $i \geq j$ existiert. *Backpropagation* beschreibt eine Lernregel für das überwachte Lernen. Es handelt sich hierbei um ein Gradientenabstiegsverfahren, welches sich in zwei Stufen gliedert [NKK94]:

1. In der ersten Stufe wird dem neuronalen Netzwerk eine Eingabe präsentiert und vorwärts durch das Netz propagiert, um die Ausgabe jeder Einheit zu berechnen.

2. In der zweiten Stufe werden zunächst die Fehlersignale für die Ausgabeneuronen durch Vergleich der berechneten mit der erwünschten Ausgabe bestimmt und die Gewichtsänderungen der zur Ausgabeschicht führenden Verbindungen ermittelt. Dies wird rückwärts für die Ausgabeschicht und alle verdeckten Schichten durchgeführt[8].

3.1.3.2 Zellulare neuronale Netze

Zellulare neuronale Netze (engl. *Cellular Neural Networks*, kurz *CNN*), entwickelt von Chua und Yang [CY88a, CY88b], stellen eine besondere Form neuronaler Netze dar. Sie bestehen aus einer Matrix von $M \times N$ Zellen, die den Neuronen entsprechen. Jede Zelle ist im Netzwerk gleich. Eine weitere herausragende Eigenschaft ist, daß jede Zelle genau mit ihren Nachbarzellen verbunden ist, wobei eine geeignete Nachbarschaft definiert wird. Diese kann eine rechteckige, hexagonale oder andere Nachbarschaft sein, die durch die Aufgabenstellung definiert wird. Jede Zelle besitzt einen internen Zustand, das Netz kann sowohl in kontinuierlicher als auch in diskreter Zeit betrieben werden.

[8] Dadurch wird auch der Name *Backpropagation* motiviert.

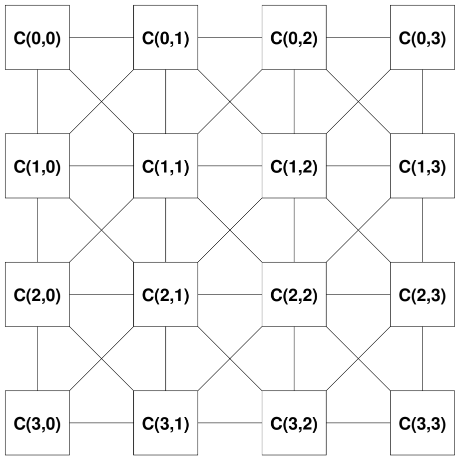


Abbildung 3.3: Schematische Darstellung eines zellularen neuronalen Netzes der Größe 4×4 *und einer* 1*-Nachbarschaft [CY88b].*

In [CY88b] wird eine sogenannte r-Nachbarschaft definiert, die eine quadratische Umgebung im Abstand r von der jeweiligen Zelle betrachtet:

Definition 3.18 (r-Nachbarschaft) Eine r-Nachbarschaft N_r^{CNN} mit $r \in \mathbb{N}$ einer Zelle $C(i,j)$ in einem zellularen neuronalen Netz der Größe $M \times N$ ist definiert durch

$$N_r^{CNN} = \{C(k,l) \mid \max\{\mid k-i \mid, \mid l-j \mid\} \leq r,\ 1 \leq k \leq M,\ 1 \leq l \leq N\} \quad (3.26)$$

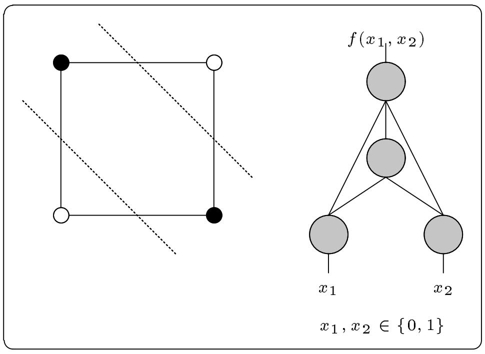
□

Offensichtlich entspricht die 1-Nachbarschaft N_1^{CNN} dem in Definition 2.4 definierten N_8. Abbildung 3.3 zeigt ein zellulares neuronales Netz der Größe 4×4 mit einer 1-Nachbarschaft.

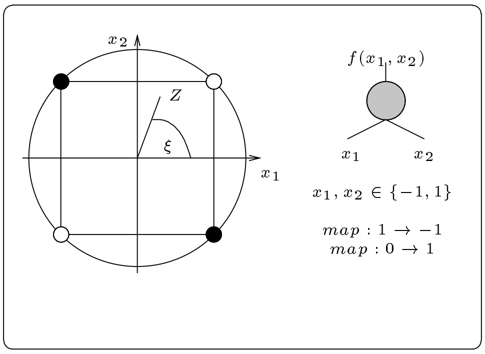
Aufgrund dieser Architektur eignen sich zellulare neuronale Netze besonders für die Bearbeitung von Bilddaten auf Bildpunktebene, da jede Zelle einen internen Zustand hat, der mit dem Grauwert des Bildes belegt werden kann, so daß das Netz genau die Bildmatrix beinhaltet. Schon in Definition 3.18 wird anhand des Definitionsbereiches $M \times N$ verdeutlichet.

Eine ausführliche Beschreibung der zellularen neuronalen Netze und deren Anwendung finden sich in den zum Teil schon genannten Arbeiten [CY88b, CY88a, CR93], vor allem ist hier eine Beschreibung einer in Hardware realisierbaren Zelle angegeben.

An dieser Stelle wird jetzt die von Igor und Naum Aizenberg entwickelte spezielle Form zellularer neuronaler Netze [Aiz92, AA94] vorgestellt, die eine besondere Art von Zellen verwenden. Dabei werden von Aizenberg zwei Arten von Zellen unterschieden: die mehrwertigen (engl. *Multi-valued Neurons*, kurz MVN)



(a) „Klassische" Lösung zur Berechnung der XOR-Funktion



(b) CNN nach Aizenberg

Abbildung 3.4: Problem der linearen Separierbarkeit. Im Gegensatz zu klassischen Netzen (Abbildung 3.4(a)) kann die XOR-Funktion mit CNN nach Aizenberg (Abbildung 3.4(b)) mit nur einer Zelle gelöst werden.

und die binärwertigen (engl. *Universal Binary Neurons*, kurz UBN) Neuronen. Im Gegensatz zu den „klassischen" CNN von Chua und Yang sind die Verbindungsgewichte bei Aizenberg komplex-wertig, die internen Zustände des Netzes sind diskret. Es existiert ein schneller Lernalgorithmus für beide Arten (MVN und UBN) von Netzen, der in [Aiz99] nachgelesen werden kann. Aufgrund der komplexen Zahlenarithmetik können beliebige Boolesche (UBN) bzw. mehrwertige (MVN) Funktionen berechnet werden und nicht nur Schwellenwertfunktionen. Bspw. kann die XOR-Funktion, die aufgrund der nicht linearen Separierbarkeit mit einem klassischen neuronalen Netz (vgl. Abschnitt 3.1.3.1) mehr als ein Neuron benötigt, mit CNN von Aizenberg mit einer einzelnen Zelle realisiert werden (Abbildung 3.4).

Dabei gilt folgende Darstellung und Verwendung von Variablen zur Berechnung der XOR-Funktion mit Hilfe von CNN nach Aizenberg. Die Grundidee liegt darin, die beiden Eingaben als eine komplexe Zahl Z (Gleichung (3.27)) aufzufassen und den Quadranten, in dem Z liegt, als Ergebnis $P(Z)$ zu verwenden (Gleichung (3.28)). m gibt die Anzahl der betrachteten Quadranten an und ist deshalb hier mit $m = 4$ definiert.

$$Z = x_1 + i \cdot x_2 = \mid Z \mid \cdot e^{i \cdot \xi} \text{ mit } \xi = \tan^{-1} \frac{x_2}{x_1} \tag{3.27}$$

$$P(Z) = (-1)^j, \text{ falls } 2\pi \cdot \frac{j+1}{m} > \underbrace{\arg(Z)}_{\xi} \geq 2\pi \cdot \frac{j}{m} \tag{3.28}$$

$$\text{mit } m = 4, \ 0 \leq j < m$$

Eine mögliche Anwendung von CNN in der medizinischen Bildverarbeitung wird in Abschnitt 4.3.1 beschrieben. Eine Übersicht der Anwendung von CNN nach Aizenberg findet sich bspw. in [AAMM98, Aiz99], ein Vergleich der CNN nach Chua und Yang und nach Aizenberg in [FW99].

Festlegung 3.3 (CNN) Wird im folgenden von CNN gesprochen, so ist immer der Ansatz nach Aizenberg gemeint. Sind CNN nach Chua und Yang gemeint, so wird dieses explizit gesagt. □

3.1.4 Grundlagen evolutionärer Algorithmen

Evolutionäre Algorithmen umfassen die Teilgebiete Evolutionsstrategien, genetische Algorithmen, evolutionäres Programmieren und genetisches Programmieren. Besprochen werden hier nur die ersten beiden Methoden, da sowohl das evolutionäre Programmieren als auch das genetische Programmieren in der weiteren Arbeit keine Anwendung finden.

Es handelt sich um Optimierungsmethoden, die Probleme nicht auf herkömmlichem, algorithmischem Weg lösen, sondern nach dem Vorbild der biologischen Evolution. Ausgehend von einer (zufällig generierten) Menge von Lösungskandidaten wird mit Hilfe einer Bewertungsfunktion eine Teilmenge daraus als beste Lösungen ausgewählt, um daraus neue Lösungsmöglichkeiten zu generieren. Eine Einführung und Übersicht findet sich etwa in [SHF94, Poh00].

3.1.4.1 Evolutionsstrategien

Evolutionsstrategien, entwickelt in den 60er und 70er Jahren von Rechenberg [Rec73] und Schwefel [Sch77], beruhen auf der Idee, aus einer Menge von möglichen Lösungen eines Problems neue (bessere) Lösungen zu erzeugen, indem nach dem Prinzip „survival of the fittest" gute Lösungen miteinander rekombiniert und durch Mutation neue Lösungen erreicht werden (vgl. Abbildung 3.5). Der Schwerpunkt wird dabei auf die Mutationskomponente gelegt. Der gesamte Prozeß wird solange wiederholt, bis ein vorher definiertes Abbruchkriterium erfüllt wird.

In einem ersten Schritt muß eine geeignete Kodierung der Lösungen erfolgen. Diese geschieht bei Evolutionsstrategien mit Realzahlen. Eine einzelne Lösung wird als *Individuum* bezeichnet, eine Menge von Individuen der gleichen Generation als *Population*. Die Bewertung der Güte eines Individuums erfolgt mit Hilfe einer *Fitneßfunktion*, die problemabhängig definiert werden muß. Abhängig von dieser Fitneßfunktion wird die optimale Lösung von einem globalen Minimum oder Maximum dieser Funktion repräsentiert.

Dann wird zunächst eine zufällige *Startpopulation* erzeugt, die aber auf Basis einer Vorgabe beruhen kann. Anschließend beginnen die Rekombinations- und Mutationsphasen. Entsprechend der Anzahl $\hat{\mu}$ der *Eltern*, also der Individuen, die für die Rekombination zur Verfügung stehen, und der Nachkommen λ, werden die Evolutionsstrategien als $(\hat{\mu}\#\lambda)$-Evolutionsstrategien bezeichnet, wobei

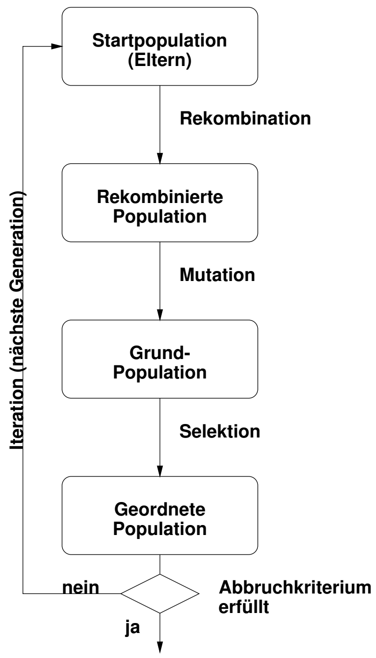


Abbildung 3.5: Grundlegende Schritte einer Evolutionsstrategie.

$1 \leq \hat{\mu} \leq \lambda$ gilt. Das # steht hierbei für „+“ (Plus) oder „,“ (Komma). Wird die sogenannte *Plus-Strategie* verwendet, werden für die spätere Auswahl der nächsten Generation (also der $\hat{\mu}$ besten Individuen) die Eltern-Individuen weiter mit berücksichtigt, während bei der *Komma-Strategie* die Elterngeneration „vergessen“ wird („stirbt“).

Für die *Rekombination* werden zufällig (gleichverteilt) ρ, $\rho \in \{1, \ldots, \hat{\mu}\}$, Eltern[9] ausgewählt. Die Rekombination erfolgt entweder so, daß einzelne Gene (Elemente eines Individuums, Realzahlen) zufällig ausgetauscht werden, oder aber der Mittelwert aus den Elementen der einzelnen Eltern gebildet wird. Das Ergebnis ist ein neues Individuum. In dieser Form werden λ Nachkommen erzeugt.

In der anschließenden *Mutationsphase* werden die λ Nachkommen mutiert, das heißt, jedes Gen eines Individuums wird zufällig verändert, wobei hier von einer normalverteilten Änderung ausgegangen wird.

Schließlich wird in der *Selektionsphase* die Fitneß jedes Individuums bestimmt. Wie erwähnt tritt hier die unterschiedliche Plus- bzw. Komma-Strategie auf. Hierzu wird bei der Komma-Strategie aus der aktuellen Population von λ Individuen bzw. bei der Plus-Strategie aus der aktuellen Population von λ Individuen plus den $\hat{\mu}$ Eltern die $\hat{\mu}$ besten Individuen als nächste *Elterngeneration* ausgewählt.

Falls das definierte Abbruchkriterium noch nicht erreicht ist, wird der Prozeß mit der neu berechneten Generation als Eltern erneut gestartet. Als Abbruchkri-

[9] Falls $\rho > 1$ ist, wird auch $(\hat{\mu}/\rho\#\lambda)$-Evolutionsstrategie geschrieben.

terium kann die Fitneß des besten Individuums herangezogen werden, oder aber auch die Zeit oder die Anzahl der erzeugten Generationen.

Es existieren viele Untersuchungen, die Aussagen über die Wahl von $\hat{\mu}$, λ, ρ und das Verhältnis untereinander machen. Auch die Schrittweite der Änderung, die über die Standardabweichung σ festgelegt wird, wird adaptiert. Dies kann durch zusätzliche Parameter, die dem Individuum hinzugefügt werden, erreicht werden. Neben der Schrittweite können noch Suchrichtungen im Parameterraum vorgegeben werden [Sch95] oder Richtungsausprägungen (eine sogenannte Schiefe, (vgl. [HRF99])). Diese Modifikationen des Grundprinzips führen zu einer Steigerung der Performanz des Systems.

3.1.4.2 Genetische Algorithmen und ein Vergleich zu Evolutionsstrategien

Genetische Algorithmen, ebenfalls in den 60er Jahren, aber vollkommen unabhängig von den Evolutionsstrategien von Holland [Hol75] entwickelt, funktionieren ähnlich wie die Evolutionsstrategien. Ein wesentlicher Unterschied besteht allerdings darin, daß bei den genetischen Algorithmen die Kodierung der Individuen *binär* erfolgt. Zudem wird bei den Genetischen Algorithmen der Schwerpunkt auf die Rekombinationsphase gelegt, wo viele unterschiedliche Methoden entwickelt wurden [SHF94]. Mutationen erfolgen nur durch das zufällige Setzen (d.h. unabhängig vom aktuellen Wert) oder Ändern (d.h. durch „Kippen") eines Bits (*Gens*). Hingegen ist gerade die Mutation ein zentraler Bestandteil bei Evolutionsstrategien.

3.2 Grundlagen der Wissensrepräsentation

Ein Hauptaspekt dieser Arbeit liegt in der Verwendung von verschiedenen Wissensarten zur Analyse medizinischer Bilddaten. Daher werden an dieser Stelle einige Begriffe, die für das Verständnis wichtig sind, erläutert.

3.2.1 Wissensarten

Wissen kann in Bezug auf seine Repräsentation in prozedurales Wissen (auch Handlungswissen) oder deklaratives Wissen (auch Faktenwissen) unterschieden werden.

Eine deklarative Wissensrepräsentation ist dadurch gekennzeichnet, daß die Aussagen über den Problemkreis passive Bestandteile der Wissensbasis eines Systems darstellen und die Abarbeitung durch einen Interpreter erfordern. Im Gegensatz dazu ist das prozedurale Wissen aktiver Natur, d.h. direkt ausführbar. Eine andere Charakterisierung lautet, daß bei der deklarativen Methode eher das „was", bei der prozeduralen Methode eher das „wie" der Eigenschaften von Elementen eines Problems im Vordergrund steht [NB87].

3.2.2 Unsicheres Wissen

Der Begriff *Unsicherheit* wird in dieser Arbeit allgemein verstanden und nicht nur probabilistisch. Neben der wahrscheinlichkeitsbedingten Unsicherheit muß deshalb auch der unscharfe Aspekt berücksichtigt werden, zudem ist die Unvollständigkeit des Wissens über einen Sachverhalt mit dem Begriff *Unsicherheit* abgedeckt.

Unsicherheit im probabilistischem Sinne gibt eine Wahrscheinlichkeit für das Eintreten eines Ereignisses an, welches dann vollständig eintritt oder nicht eintritt. Bspw. kann hier die Wahrscheinlichkeit der Existenz eines Tumors in einem MRT-Kopfbild angegeben werden[10], wenn gewisse Informationen (Anamnesedaten) über den Patienten vorliegen. Der Patient hat einen Tumor oder nicht. Unsicherheit im Kontext der Unschärfe gibt an, *wie sehr* (zu welchem Grad) ein Ereignis eintritt. Die Aussage, der Patient hat einen *großen* Tumor, ist unter anderem davon abhängig, wie der Term *groß* interpretiert wird, dementsprechend kann ein Zugehörigkeitsgrad des gemessenen Tumors zur Klasse der *großen Tumoren* angegeben werden. Schließlich kann noch eine Unsicherheit durch mangelndes oder unvollständiges[11] Wissen existieren, etwa darüber, welche Krankheiten der Patient zuvor gehabt hat.

3.2.3 Wissenserwerb

Der Wissenserwerb (Wissensakquisition) wird hier nicht näher erläutert, es wird davon ausgegangen, daß das benötigte Wissen in geeigneter Form vorliegt. Dieses Wissen wird in Wissensakquisitionssitzung u. a. in Form von Interviews, Beobachtungen und durch Literaturstudium gewonnen. Einen Überblick über Akquisitionsmethoden findet sich z. B. in [Gör93, Fat91, Fat92].

3.2.4 Wissensrepräsentation

Nachdem verschiedene Arten von Wissen bzw. die dort inhärenten Unsicherheiten erläutert wurden, werden hier die in der Arbeit verwendeten Methoden zur Repräsentation des verwendeten Wissens vorgestellt. Die genannten Wissensrepräsentationsmethoden werden in [Rei91] ausführlich behandelt und anhand von Beispielen erläutert.

3.2.4.1 Regelbasierte Wissensrepräsentation

Eine Regel besteht aus einer Vorbedingung (Prämisse) und einer Schlußfolgerung bzw. Aktion (Konklusion). Die Konklusion wird ausgeführt, wenn die Prämisse

[10] Von der Wahrscheinlichkeit bzw. Möglichkeit, daß der Datensatz falsche Informationen zeigt, wird hier abgesehen.

[11] Wissen ist unvollständig, wenn eine Menge von Aussagen existieren, von denen mindestens eine wahr ist, aber nicht bekannt ist, welche (vgl. [Rei91]).

erfüllt ist. Diese Art der Repräsentation eignet sich besonders gut für prozedurales (Handlungs-)Wissen, in der die Konklusionen aus Aktionen bestehen, die ausgeführt werden, sobald die Vorbedingung erfüllt ist. In diesem Fall wird häufig die Bezeichnung „die Regel feuert“ verwendet.

3.2.4.2 Semantische Netze

Semantische Netze sind eine graphartige Repräsentation des Wissens, in denen Konzepte in den Knoten gespeichert werden, während die Kanten (gerichtet oder ungerichtet) assoziative Beziehungen zwischen den Konzepten widerspiegeln. Ein Beispiel wird in Abbildung 3.6 gezeigt. Es wird zwischen Netzen mit untypisierten Kanten (es gibt nur eine Art von Kante) oder typisierten Kanten (es gibt verschiedene Arten von Kanten mit unterschiedlicher Semantik) unterschieden, wobei erstere kaum mehr eine Rolle spielen. Unter einem Konzept wird dabei folgendes verstanden:

Definition 3.19 (Konzept) Ein Konzept wird durch ein 3-Tupel *(N, E, I)* definiert. Die *Extension E* ist die Menge aller Objekte, die zum Konzept gehören. Die *Intension I* gibt die Eigenschaften an, die ein Objekt haben muß, um zu diesem Konzept zu gehören. *N* gibt den *Namen* des Konzeptes an. □

In semantischen Netzen werden die Eigenschaften eines Konzeptes ebenfalls durch ein Konzept repräsentiert, wobei dann die Kante, die einen Konzeptknoten mit einem solchen *Eigenschaftsknoten* verbindet, die *Eigenschaftsklasse* definiert.

3.2.4.3 Frames

Ähnlich wie semantische Netze repräsentieren auch Frames [Min74] das Wissen graphartig, wobei hier der Graph durch einen Baum dargestellt wird und die Knoten den Frames entsprechen. Dadurch wird die besondere Eigenschaft der Frames, die Vererbung, sehr anschaulich dargestellt. Jedoch haben die Kanten des Graphen, welche die Vererbung repräsentieren, lediglich die mögliche Typisierung *is-a* oder *instance-of*. Eigenschaften werden direkt in den Frames in sogenannten *Slots* gespeichert, wobei zwischen terminalen und nicht-terminalen Slots (diese beinhalten wiederum einen Frame[12]) unterschieden wird. Frames, die lediglich terminale Slots enthalten, bilden die Blätter des Baums (vgl. Abbildung 3.7).

Definition 3.20 (Frame) Ein Frame wird durch ein 3-Tupel *(Name, NT-Slots, T-Slots)* definiert, wobei NT-Slots die nicht-terminalen und T-Slots die terminalen Slots darstellen. Nicht-terminale Slots beinhalten wiederum Frames, während terminale Slots lediglich eine Zeichenkette enthalten. □

[12] Das Beinhalten wird durch eine Referenz über den Namen realisiert.

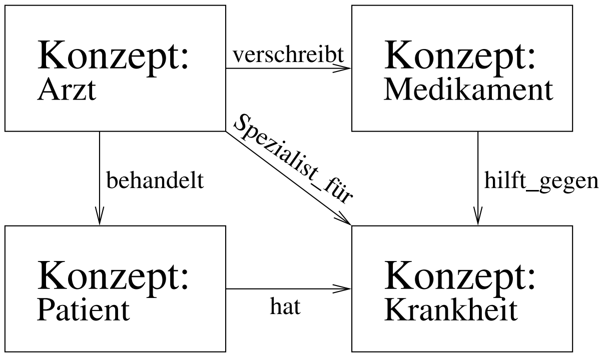


Abbildung 3.6: Einfaches Beispiel für ein semantisches Netz.

Abbildung 3.7: Einfaches Beispiel für Frames (ohne Vererbung).

Durch die Möglichkeit zur Vererbung werden Eigenschaften möglichst nah an der Wurzel des Baumes definiert, die dann als Voreinstellungswerte (Default-Werte) in allen davon abgeleiteten Frames solange gelten, bis sie überschrieben werden. Dadurch wird der Grundidee, Wissen werde in stereotypischen Erinnerungsmustern gespeichert, besonders Rechnung getragen.

Weiterhin bieten Frames die Möglichkeit, prozedurale Komponenten zu speichern. Dies erfolgt in sogenannten Dämonen, die etwa die Berechnung eines Eigenschaftswertes anstoßen, sofern der Wert dieser Eigenschaft abgefragt wird. Auch ist es mit diesen Dämonen möglich, eine Gültigkeitsprüfung für Eigenschaftswerte durchzuführen, falls diese geändert werden. Dementsprechend sind die Dämonen immer an bestimmte Slots gebunden.

3.2.4.4 Vergleich semantischer Netze und Frames

Auf den ersten Blick scheinen semantische Netze und Frames gleich zu sein. Unterschiede finden sich jedoch in der Struktur der Repräsentation. Während Frames nur eine Baumstruktur erlauben und somit das Konzept der Vererbung betonen, erlauben semantische Netze beliebige Relationen zwischen zwei Konzepten. Während bei Frames die Eigenschaften im Frame selbst in den Slots gespeichert werden können, werden diese bei semantischen Netzen in eigenen Eigenschaftskonzepten gespeichert (und durch die Intensionen referenziert). Schließlich erlauben Frames noch die Möglichkeit zur Verwendung von Dämonen, also prozeduralen Komponenten. Das in dieser Arbeit realisierte System verwendet deshalb ein hybrides Repräsentationsmodell[13], wie in Abschnitt 5.2.1.1 gezeigt wird.

3.2.4.5 Weiteres Vorgehen

An dieser Stelle sind alle Grundlagenbegriffe insofern eingeführt, daß im nachfolgenden eine Analyse der aktuellen CI-Methoden auf dem Gebiet der Bildverarbeitung vorgenommen werden kann. Diese Analyse liefert dann wiederum den Ausgangspunkt für die eigenen Ansätze. Kommen dabei Erweiterungen bzw. Abwandlungen der bis dato vorgenommenen Bezeichner und Definitionen vor, so werden diese mit einem entsprechenden Verweis versehen.

[13] Ähnlich wie in [NB87], der allerdings nicht zwischen Frames und Konzepten unterscheidet.

4 CI-Methoden in der digitalen Bildverarbeitung

Der Einsatz einzelner CI-Methoden in der Bildverarbeitung ist Thema der Forschung schon seit den 70er Jahren. Vorrangig wurde die Verwendung unscharfer Methoden untersucht (vgl. etwa [Ros79, BP92b, Tiz98]). In diesem Kapitel wird eine Übersicht über verschiedene Arbeiten gegeben, die eine oder mehrere der CI-Methoden im Bereich der (medizinischen) Bildverarbeitung verwenden, wie sie in Abschnitt 3.1 vorgestellt wurden. Dabei erfolgt die Gliederung der Abschnitte nach den einzelnen Bildverarbeitungsstufen aus Abbildung 2.3.

Festlegung 4.1 (CI-Bildverarbeitung) Unter CI-Bildverarbeitung werden alle Ansätze in der digitalen Bildverarbeitung verstanden, bei denen Bilder auch mit Hilfe von CI-Methoden, wie sie in Abschnitt 3.1 beschrieben sind, verarbeitet werden. In welcher Art und Weise dies erfolgt, hängt von der jeweiligen Problemstellung ab. □

Diese Festlegung erweitert die von Tizhoosh [Tiz98] gegebene Definition der Fuzzy-Bildverarbeitung um den Ansatz neuronaler Netze und evolutionärer Algorithmen. Vor allem ist herauszustellen, daß die Verarbeitung nicht nur auf der untersten Ebene der Bildpunktverarbeitung (der sogenannten Low-Level-Bildverarbeitung) erfolgt, sondern auch die Verwendung von Expertenwissen in der High-Level-Bildverarbeitung ausdrücklich erlaubt ist. Im Prinzip wird die klassische Bildverarbeitung um die genannten Methoden an den Stellen ergänzt, wo dadurch Vorteile erzielt werden.

4.1 Übersicht bestehender CI-Methoden für die medizinische Bildverarbeitung

Abbildung 4.1 zeigt eine Übersicht einer (sicherlich nicht vollständigen) Literaturliste, wie sich dort die CI-Methoden in den einzelnen Teilgebieten der Bildverarbeitung verteilen. Bei dieser Liste handelt es sich um eine bezüglich des

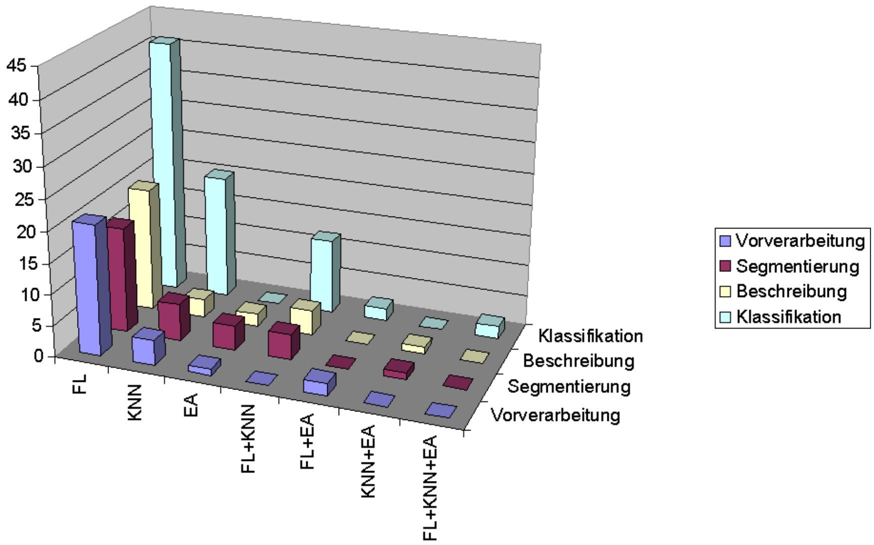


Abbildung 4.1: Verteilung der Verwendung von CI-Methoden in der medizinischen Bildverarbeitung gegliedert nach Stufen der Prozeßkette und den eingesetzten Methoden (insgesamt 167 Arbeiten).

Themengebietes repräsentativ[1] ausgewählte Sammlung von Arbeiten. Die dazugehörige zahlenmäßige Verteilung zeigt Tabelle 4.1. Nicht alle Arbeiten, die hier genannt werden, beziehen sich auf medizinische Bilder, sondern teils auch auf andere Bildmaterialien. Trotzdem wird deutlich, daß gerade Fuzzy-Methoden schon verstärkt genutzt werden, während Kombinationen der CI-Methoden nicht oder nur selten zu finden sind. Wenn keine eindeutige Zuordnung einer Arbeit zu einem bestimmten BV-Teilgebiet der Prozeßkette möglich war, wurde diese Arbeit zur höchsten verwendeten Abstraktionsebene gezählt (was die relativ hohe Anzahl von Fuzzy-Arbeiten in der Klassifikationstufe erklärt).

Im folgenden werden einige Arbeiten etwas näher diskutiert. Gegliedert werden die Abschnitte nach ihrem Bezug zu der in Abbildung 2.3 vorgestellten Prozeßkette in Vorverarbeitung, Segmentierung, Beschreibung und Klassifikation, sofern eine genaue Zuordnung möglich ist.

[1] Repräsentativ in dem Sinne, daß die Summe aller betrachteten Arbeiten alle CI-Methoden und alle Stufen der Bildverarbeitungsprozeßkette abdecken, explizit vorgestellt wird jeweils ein Beispiel einer Anwendung.

Methode	Vorverarbeitung	Segmentierung	Beschreibung	Klassifikation	$\sum$
FL	21	17	20	41	99
KNN	4	6	3	20	33
EA	1	4	2	0	7
FL+KNN	0	4	4	12	20
FL+EA	2	0	0	2	4
KNN+EA	0	1	1	0	2
FL+KNN+EA	0	0	0	2	2
$\sum$	28	32	30	77	167

Tabelle 4.1: Zahlenmäßige Verteilung der CI-Methoden zu den Prozeßstufen (vgl. Abbildung 4.1).

4.2 Anwendung unscharfer Methoden in der medizinischen Bildverarbeitung

4.2.1 Bildvorverarbeitung

Die Bildvorverarbeitung findet auf der Grauwertebene statt. Für die Anwendung unscharfer Methoden werden die Grauwerte unscharf interpretiert, das heißt, jedem Grauwert wird ein Zugehörigkeitswert zugeordnet, der abhängig von der Zielstellung ist. Ausgehend von diesem Zugehörigkeitswert wird der Grauwert in einen anderen Grauwert transformiert, so daß das Bild für eine weitere Verarbeitung besser geeignet ist bzw. eine visuelle Verbesserung stattgefunden hat. Die Zugehörigkeitswerte sind Singletons zu den einzelnen Bildpunkten und geben bspw. an, wie sehr ein Bildpunkt bzw. dessen Grauwert *hell* ist.

[Tiz98] beschreibt z. B. eine regelbasierte Methode zur Kontrastverbesserung eines Bildes. Dabei werden drei Regeln implementiert:

- Wenn der Grauwert *dunkel* ist, dann mache ihn *schwarz*.
- Wenn der Grauwert *grau* ist, dann mache ihn *grau*.
- Wenn der Grauwert *hell* ist, dann mache ihn *weiß*.

Das Regelsystem hat den Vorteil, sehr einfach zu sein. So kann das gewünschte Ziel ohne Komplikationen beschrieben werden. Es läßt sich auch mit Hilfe einer Sigmoidfunktion beschreiben, die in der klassischen Bildverarbeitung zur Kontrastverbesserung verwendet wird. Eine ähnliche Regelmenge zur Bestimmung eines Schwellenwertes wird in [KTL+98] beschrieben. Aber gerade die sehr einfachen Regeln erlauben auch eine erweiterbare Methode zur Kontrastanhebung durch zusätzliche Regeln zur Rauschminderung, etwa mit Regelparametern wie:

- Wenn der Grauwert *dunkel* ist und die benachbarten Bildpunkte *hell* sind, dann mache den Grauwert *hell*.
- Wenn der Grauwert *hell* ist und die benachbarten Bildpunkte *dunkel* sind, dann mache den Grauwert *dunkel*.

Mit derartigen Regeln können etwa verrauschte Bildpunkte entfernt werden. Allerdings liegt hier wie bei allen Fuzzy-Systemen das Problem in der Definition der verwendeten Fuzzy-Mengen und der Regeln, die entscheidend für die Qualität des Ergebnisses sind. Auch [LH95] verwendet solche Regeln, um das Rauschen im Bild zu vermindern, wobei als Zugehörigkeitswert, der den Grad der Rauschfreiheit angibt, die absolute Differenz zum mittleren Grauwert der Umgebung betrachtet wird.

Einen ähnlichen Ansatz verfolgt Arakawa in [Ara96]. Er erweitert einen Medianfilter (vgl. Abbildung 2.4) um eine Regelkomponente. Neben der normalen Berechnung des Median-Wertes in einem Grauwertfenster wird anhand von vier Regeln ein Vertrauensmaß $\mu(p)$ berechnet. Dieses gibt an, wie sehr der Grauwert des aktuell betrachteten Punktes p als verrauscht gilt[2]. Mit diesem Maß wird der neue Grauwert an der Stelle p berechnet:

$$I'(p) = m(p) + \mu(p) \cdot (I(p) - m(p)) \tag{4.1}$$

Hier stellt I das Bild und m den Medianwert bezüglich eines Fensters um p dar. Ist das Vertrauensmaß μ in den Punkt p gleich Null, ergibt sich aus Gleichung (4.1), daß der Medianwert verwendet wird. Ist das Maß gleich Eins (Punkt ist nicht verrauscht), wird der Funktionswert an der Stelle beibehalten. Das Problem ist wiederum die Berechnung des Vertrauensmaßes μ, welches Arakawa mit folgenden Regeln löst:

- Wenn der Wert $u(p)$ *klein* ist und $v(p)$ *klein* ist, ist das Vertrauen *hoch*.
- Wenn der Wert $u(p)$ *klein* ist und $v(p)$ *groß* ist, ist das Vertrauen *gering*.
- Wenn der Wert $u(p)$ *groß* ist und $v(p)$ *klein* ist, ist das Vertrauen *gering*.
- Wenn der Wert $u(p)$ *groß* ist und $v(p)$ *groß* ist, ist das Vertrauen *sehr gering*.

Hierbei sind $u(p)$ und $v(p)$ Hilfsvariablen. $u(p)$ gibt die absolute Differenz vom Medianwert $m(p)$ zum Funktionswert $I(p)$ an. $v(p)$ gibt den Durchschnitt der absoluten Differenzen von $I(p)$ zu den beiden nächsten Funktionswerten an, die im betrachteten Fenster liegen. $u(p)$ wird dann groß, wenn zwischen dem Medianwert und dem Funktionswert ebenfalls eine große Differenz ist. Der Wert $v(p)$ wird dann groß, wenn auch die nächsten Nachbarn eine große Differenz haben. In diesen Fällen gilt der Punkt als sehr stark verrauscht. Die Festlegung der unscharfen Mengen für die Interpretation der Terme muß wieder kontextabhängig erfolgen.

[2] Es wird nicht berechnet, *ob* der Punkt verrauscht ist oder nicht!

4.2.2 Segmentierung

Die Segmentierung mit Hilfe unscharfer Methoden kann etwa durch ein unscharfes Bereichswachstumsverfahren, wie es in Abschnitt 5.2.2 vorgestellt wird, erfolgen. Interessant ist auch die Erweiterung der Hough-Transformation[3] um eine unscharfe Komponente. Entsprechende Arbeiten finden sich etwa bei [HKP94]. Die generalisierte Hough-Transformation von Ballard [Bal81] wird von Philip et al. [PDMP$^+$94] in einer unscharfen Variante vorgestellt. Die scharfe Variante hat einen Zeitaufwand von $O(M \cdot N \cdot K^2)$, wenn das untersuchte Bild die Größe $M \times N$ und das im Bild gesuchte Modell eine Konturlänge von K hat. Die Grundidee bei beiden und auch bei anderen Arbeiten über eine unscharfe Hough-Transformation liegt darin, den Rand des gesuchten Objektes unscharf zu betrachten, wie es etwa auch in Abschnitt 5.2.3.4 vorgestellt wird. Die klassische Hough-Transformation [Hou62] verwendet ein Akkumulatorfeld, welches inkrementiert wird, wenn ein Punkt im Bild und ein Punkt im gesuchten Modell übereinstimmen[4]. Bei der unscharfen Variante wird der Akkumulator aber nicht um den Wert 1 erhöht, sondern um einen Wert $\mu \in \langle 0, 1 \rangle$, der vom Abstand des Punktes zum Modell abhängig ist. Das generelle Problem der Hough-Transformation ist der Zeitaufwand für die Berechnung. Dieser erhöht sich damit nochmals deutlich, da nicht nur für jeden Punkt des Modells eine Zugehörigkeit zum Bild getestet werden muß, sondern für jeden Punkt des unscharfen Randes des Modells. Die Anzahl zusätzlicher Punkte ist abhängig vom Grad der gewählten Unschärfe des Modells, der Mehraufwand ist direkt davon abhängig (wird eine $m \times n$ Umgebung für jeden Bildpunkt betrachtet, erhöht sich der Zeitaufwand auch um einen Faktor $m \cdot n$ auf $O(M \cdot N \cdot K^2 \cdot m \cdot n)$.

Die unscharfe Berechnung eines Schwellenwertes für die Segmentierung von Bildern wird etwa in [HW95] beschrieben. Für das Maß der Unschärfe (engl. *Measure of Fuzziness*) kann die von De Luca und Termini [DLT72] definierte Fuzzy-Entropie $E(\mathcal{A})$ einer diskreten Fuzzy-Menge $\mathcal{A}$ der Kardinalität des Trägers n verwendet werden, die wie folgt definiert ist:

$$E(\mathcal{A}) = \frac{1}{n \cdot \ln 2} \cdot \sum_{i=1}^{n} S(\mu_{\mathcal{A}}(a_i)) \tag{4.2}$$

mit der Shannon-Funktion

$$S(x) = -x \cdot \ln x - (1 - x) \cdot \ln(1 - x), \quad x \in \langle 0, 1 \rangle. \tag{4.3}$$

[3] Die Hough-Transformation wurde ursprünglich zur Detektion von kollinearen Bildpunkten definiert [Hou62] und schließlich zur Findung beliebiger Segmente, die durch ihre Kontur gegeben sind, erweitert [Bal81, Hab95].

[4] Diese Beschreibung ist hier sehr vereinfacht. Eine Einführung in die sowohl Standard-Hough-Transformation als auch in die generalisierte Hough-Transformation findet sich etwa in [Hab95].

Der Zugehörigkeitsgrad $\mu_{\mathcal{A}}$ eines Bildpunktes wird ausgehend von einem festen Schwellenwert T als absolute Differenz zwischen dem Grauwert $I(p)$ des Bildpunktes p und dem durchschnittlichen Grauwert der Region, zu der er gehört, berechnet. Die Region, zu der der Punkt gehört, ist abhängig von T. Das heißt, es wird der durchschnittliche Grauwert aller Bildpunkte bestimmt, deren Grauwert kleiner oder gleich T ist, falls der Grauwert des Punktes kleiner oder gleich T ist. Ansonsten wird der durchschnittliche Grauwert der Bildpunkte, deren Grauwert größer T ist, verwendet:

$$\mu_{\mathcal{A}}(I(p)) = \begin{cases} \dfrac{1}{1 + \dfrac{\mid I(p) - m_1(T) \mid}{C}} & \text{falls } I(p) \leq T \\ \dfrac{1}{1 + \dfrac{\mid I(p) - m_2(T) \mid}{C}} & \text{sonst} \end{cases} \tag{4.4}$$

Hierbei ist C die maximale Anzahl möglicher Graustufen, das heißt $C = G_{\max} - G_{\min} + 1$. Die $m_i(T)$ beschreiben den mittleren Grauwert der durch T definierten Region und berechnen sich mit $h(g)$ als Histogrammfunktion über die Grauwerte wie folgt:

$$m_1(T) = \frac{\sum\limits_{g=G_{\min}}^{T} g \cdot h(g)}{\sum\limits_{g=G_{\min}}^{T} h(g)} \quad \text{und} \quad m_2(T) = \frac{\sum\limits_{g=T+1}^{G_{\max}-1} g \cdot h(g)}{\sum\limits_{g=T+1}^{G_{\max}-1} h(g)} \tag{4.5}$$

Zur Minimierung des Maßes der Unschärfe wird für jeden möglichen Schwellenwert $T \in [G_{\min}, G_{\max}]$ die in Gleichung (4.2) definierte Fuzzy-Entropie berechnet und der Schwellenwert gewählt, bei dem diese minimal ist. Abbildung 4.2 zeigt einen Vergleich der Berechnung des Schwellenwertes mit Hilfe eines Wertes, der im Minimum des Histogramms zwischen den beiden Maxima liegt und der hier vorgestellten Methode der Fuzzy-Entropie-Minimierung, die zwar einen höheren Rauschanteil enthält, dafür aber das zu segmentierende Objekt (hier den Kopf) weitaus vollständiger enthält. [HW95] betrachtet noch ein weiteres Maß zur Minimierung, das hier aber aus Platzgründen nicht vorgestellt wird. Die Idee ist die gleiche, jedoch wird als Maß das Yager-Maß verwendet, welches den Abstand (bezüglich einer zu definierenden Metrik) zwischen dem Bild und seinem Komplement berechnet[5].

4.2.3 Beschreibung

Die unscharfe Beschreibung von Bildeigenschaften oder Objekteigenschaften ist von herausragender Bedeutung und bietet die Möglichkeiten, Variabilitäten zu

[5] Dieser muß nicht wie in der zweiwertigen Logik maximal sein, da der Satz des ausgeschlossenen Dritten unter Verwendung von unscharfen Methoden nicht mehr gilt.

(a) Original-Bild (b) Histogramm-Minimum (c) Minimales Fuzzy-Maß

Abbildung 4.2: Berechnung des Schwellenwertes für die Segmentierung durch Minimierung des Maßes der Unschärfe.

modellieren. Diverse Interpretationsmöglichkeiten unscharfer Begriffe werden ausführlich in Abschnitt 5.2.3 vorgestellt.

Generell kann hierzu gesagt werden, daß versucht wird, ein Objekt mit Hilfe der Beschreibungen seiner Eigenschaften zu erkennen. Das heißt, es müssen Eigenschaften verwendet werden, anhand derer eine Klasse von Objekten von anderen Klassen unterschieden werden kann. Da aber gerade bei medizinischen Bildern die Klassen eine größere Variabilität in ihren Eigenschaften besitzen als dies etwa bei technischen Fertigungsprozessen vorkommt, ist die Verwendung unscharfer Beschreibungen ein wichtiges Hilfsmittel.

Als unscharfe Beschreibungen können sämtliche Eigenschaften verwendet werden, die für Bilder oder Segmente in den Bildern berechnet werden können. Für eine unscharfe Beschreibung wird diese Berechnung in einen unscharfen Term übersetzt, was auch als Fuzzifizierung oder Verunschärfung bezeichnet wird. Neben den in Abschnitt 5.2.3 genannten Beschreibungen werden hier deshalb keine weiteren Arbeiten vorgestellt.

4.2.4 Klassifikation

In der Klassifikation werden anhand der Eigenschaften, welche für die Segmente berechnet wurden, die Segmente den gesuchten Objekten zugeordnet bzw. den Segmenten Objektnamen zugewiesen.

Welche Eigenschaften für ein Segment berechnet werden, wird in einer Wissensbasis definiert. Diese beschreibt die gesuchten Objekte. Vereinfacht gesagt wird ein Segment als das Objekt identifiziert, bei dem die Übereinstimmung zwischen den Beschreibungen in der Wissensbasis auf der einen Seite und den Eigenschaften des Segmentes auf der anderen Seite maximal ist. Allerdings sind hier noch gewisse Randbedingungen zu beachten, etwa daß bestimmte Schwellenwerte für die Übereinstimmung nicht unterschritten werden dürfen. Eine detaillierte Vorstellung für die Durchführung einer Klassifikation ist in Abschnitt 5.2.4 zu finden.

4.3 Anwendung neuronaler Netze in der medizinischen Bildverarbeitung

4.3.1 Bildvorverarbeitung

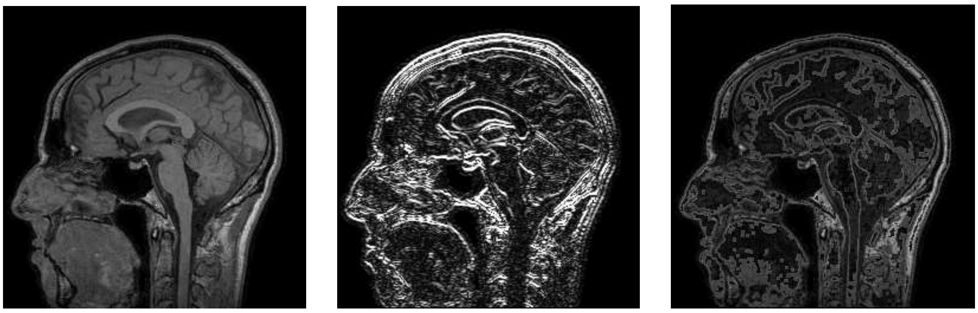
In [AAMM98, Aiz99, AAH+01] beschreiben Aizenberg et al. die Anwendung von CNN, einer speziellen Form neuronaler Netze (vgl. Abschnitt 3.1.3.2), in der Bildverarbeitung. Vereinfacht kann hier gesagt werden, daß die CNN mit Booleschen Funktionen trainiert werden, die auf den Bilddaten definiert sind. Zur Verstärkung der Kanten wird ein Graustufenbild in seine einzelnen Bitebenen zerlegt, so daß bei einem 256 (bzw. n) Graustufenbild insgesamt 8 (bzw. $\lceil log_2\, n \rceil$) Bitebenen entstehen, die dann Binärbildern entsprechen. Auf diesen Binärbildern wird für jeden Punkt x_{22} und dessen N_8 Nachbarschaft folgende Boolesche Funktion Y_{edge} angewandt:

$$Y_{edge}\left(\begin{bmatrix} x_{11} & x_{12} & x_{13} \\ x_{21} & x_{22} & x_{23} \\ x_{31} & x_{32} & x_{33} \end{bmatrix}\right) = x_{22} \wedge (\overline{x}_{11} \vee \overline{x}_{12} \vee \overline{x}_{13} \vee \overline{x}_{21} \vee \overline{x}_{23} \vee \overline{x}_{31} \vee \overline{x}_{32} \vee \overline{x}_{33}) \tag{4.6}$$

Diese Operation wird für jede Bitebene getrennt berechnet und die Bitebenen anschließend wieder zu einem Gesamtbild überlagert. Die Funktion Y_{edge} erhält auf den einzelnen Bitebenen also nur die Bildpunkte, die mindestens einen Nachbarpunkt aus N_8 (bzw. N_1^{CNN}) haben, der nicht gesetzt ist (und somit zwischen diesen Punkten eine „Kante“ definiert wird). Abbildung 4.3 zeigt die Anwendung von Gleichung (4.6) im Vergleich zu einem Sobel-Filter. Deutlich zu erkennen ist, daß die Kanten in Abbildung 4.3(c) geschlossener sind, das gesamte Bild ist allerdings insgesamt dunkler.

4.3.2 Segmentierung

Valli et al. beschreiben in [VPCC98] die Anwendung neuronaler Netze zur Segmentierung von Bilddaten. In einem ersten Schritt wird dabei die Auflösung des Bildmaterials verringert, um so eine ROI mit Hilfe eines vorwärtsgerichteten Netzes zu berechnen (ein sogenannter *Attention Focuser*). Die Eingabeneuronen beinhalten das gesamte (in der Auflösung reduzierte) Bild, als Ausgabe wird der Mittelpunkt der ROI berechnet. In einem zweiten Schritt wird mit einem weiteren neuronalen Netz (bestehend aus 2 Teilnetzen) auf dem hochaufgelösten Bereich



(a) Original-Bild (b) Sobel-Filterung (c) CNN-Filterung

Abbildung 4.3: Anwendung von CNN für die Hervorhebung von Kanten im Vergleich zum Sobel-Filter.

diese ROI genauer segmentiert, wobei als Eingabe der durchschnittliche Grauwert und die Position einer $N \times N$-Maske für jeden Bildpunkt berechnet wird. Als Trainingsmuster dienten hierbei zuvor manuell segmentierte Regionen.

In [SHP01] beschreiben Sieg et al. die Anwendung eines neuronalen Netzes zur Segmentierung von Hirntumoren. Das Netz verwendet vier verschiedene (bereits registrierte, d.h. abgeglichene) Bilddatensätze des gleichen Patienten als Eingabe. Dabei werden dem Netz die jeweils positionsgleichen Bildpunkte jeden Bildes als Eingabe präsentiert, als Ausgabe erfolgt die binäre Entscheidung, ob der Bildpunkt zu einer Tumorregion gehört oder nicht. Segmentiert werden können mit dieser Methode kontrastmittelaufnehmende Tumoren, die sich im Bildmaterial deutlich von der umliegenden Hirnmasse unterscheiden. Als Tumor wird das Segment bezeichnet, welches die größte 3-dimensionale Zusammenhangskomponente bildet. Es wird bei einer Trainingsdatenmenge von 22 Datensätzen eine 86-prozentige korrekte Segmentierungsleistung genannt, wobei das Netz 3 verdeckte Schichten besitzt. In einer anschließenden Nachverarbeitung erfolgt eine Auffüllung evtl. vorhandener Segmentierungslöcher.

4.3.3 Beschreibung

Da KNN auch als Funktionennetze aufgefaßt werden können [Roj94], ist die Anwendung als Funktion zur Berechnung einer Eigenschaft naheliegend. Die Berechnung von Eigenschaften gegebener Strukturen kann mit Hilfe neuronaler Netze insofern erfolgen, als daß dem Netz ein Segment, für welches die Eigenschaft(en) berechnet werden, entweder direkt oder indirekt (vorverarbeitet) präsentiert wird, um daraus weitere Eigenschaften (bzw. einen Zugehörigkeitswert zu diesen Eigenschaften) zu berechnen. Sinnvoll ist dies bei leicht berechenbaren Eigenschaften als Eingabe und aufwendig (oder schwer formulierbar) zu berechnende Eigenschaften als Ausgabe. In [HRK+99] wird etwa, nachdem die Eigenschaftsmenge mit Hilfe genetischer Algorithmen optimiert wurde (vgl. Abschnitt 4.4.3), die Berechnung der Eigenschaften mit Hilfe eine FF-Netzes berechnet.

4.3.4 Klassifikation

Falls aus den Eigenschaften unmittelbar die Klassenzugehörigkeit berechnet werden kann, kann der Schritt einer Beschreibung übersprungen werden. Dem Netz wird als Eingabe ein Eigenschaftsvektor präsentiert, anhand dessen die Klassenzugehörigkeit wie etwa in Abschnitt 5.3.2 beschrieben, berechnet wird.

4.4 Anwendung evolutionärer Algorithmen in der medizinischen Bildverarbeitung

4.4.1 Bildvorverarbeitung

Filter werden häufig zur Bildvorverarbeitung verwendet, wobei die Festlegung der Filterelemente entscheidend für die Qualität der Ergebnisse ist. Die Optimierung eines Filters zur Hervorhebung spezieller Kantenformen wird z. B. in [HF00, FH01] vorgestellt. Dazu wird ein 3×3-Filter mit Werten aus $\mathbb{R}$ als Individuum kodiert. In der Iterationsphase wird der Filter anhand von gesuchten Kantenformen adaptiert, im o. g. Fall wurde der Filter auf die Betonung von Kreisstrukturen optimiert. Als Fitneßfunktion diente hierbei die Differenz zwischen der gewünschten Kante und der durch Faltung des Bildes mit der Maske berechneten Kante. Als Ergebnis wurde ein Filter zur Hervorhebung von Kanten für einen speziellen Bildtyp mit Inhalten gleichen Typs gewonnen. Auch [BA97] beschreibt die Optimierung von Filtern zur Rauschreduktion mit Hilfe genetischer Algorithmen.

4.4.2 Segmentierung

Eine Segmentierung medizinischer Bilddaten wird etwa in [BKKS96, BGKM98] vorgestellt. In dieser Arbeit werden Angiographie[6]-Bilder mit Hilfe eines evolutionären Algorithmus segmentiert. Ziel ist die automatische Segmentierung des zweidimensionalen Gefäßskelettes. Dazu wird das Ausgangsbild in gleichgroße Teilbilder (32×32 Bildpunkte) zerlegt, die unabhängig voneinander segmentiert werden. Die Idee liegt in der evolutionären Modifikation geometrischer Primitive, wie zum Beispiel von Linien oder Rechtecken. Der Ausgangspunkt sind bereits segmentierte Bilder, welche eine sogenannte „Grundpopulation“ bilden und die aus den Primitiven bestehen. In der Segmentierungsphase eines unbekannten

[6] Angiographie beschreibt die röntgenologische Darstellung der Blutgefäße.

Bildes, welches Zielbild genannt wird, wird dieses evolutionär aus der Grundpopulation rekonstruiert. In der Rekombinations- und Mutationsphase werden neben den Grauwertbildern auch deren Segmentierung mit berücksichtigt, so daß ein Individuum aus einem Grauwertbild und dem dazugehörigen segmentierten Bild, der sogenannten Labelmatrix, besteht. Die Fitneß eines Individuums berechnet sich als Differenzbild zwischen dem zu segmentierenden Zielbild und dem Grauwertbild des Individuums. Das Ergebnis des Algorithmus ist schließlich ein Individuum, dessen Binärteil dem segmentierten Zielbild entsprechen soll bzw. der besten Segmentierung des Zielbildes entspricht. Für Details der Mutationskomponente sei hier auf [MVKG97] verwiesen.

Die Berechnung der Schwellenwerte mit Hilfe genetischer Algorithmen in einem Multischwellenwertverfahren wird in [Yin99] beschrieben. Dabei werden die c Schwellen[7] binär kodiert und als Chromosom (Individuum) betrachtet. Als Fitneßfunktion wird die Maximierung der Entropie der zugehörigen Histogrammfunktion des Bildes verwendet. Zur Optimierung wird ein Standard-GA genutzt. Es wird lediglich eine Gültigkeitsprüfung der neu erzeugten Chromosomen durchgeführt, wobei dieselbe Schwelle nicht mehrfach vorkommen darf.

4.4.3 Beschreibung

Die Verwendung von evolutionären Algorithmen zur Bildauswertung kann zur Auswahl der geeigneten Eigenschaften, die betrachtet werden sollen, erfolgen. Dabei wird die Menge aller möglichen Eigenschaften in einem Individuum binär kodiert, wobei jedes Bit eine Eigenschaft repräsentiert. Anschließend kann eine Bildauswertung mit den im Individuum definierten Eigenschaften erfolgen, wobei das Ergebnis der Auswertung als Fitneß des Individuums gewertet werden kann [BA97]. Auch in [HRK+99] werden die zu berechnenden Eigenschaften mit Hilfe genetischer Algorithmen bestimmt.

4.5 Resümee

Wie gezeigt wurde, kann die Verwendung von Fuzzy-Methoden in der Bildverarbeitung an vielen Stellen stattfinden. Über den Sinn einer regelbasierten Kontrasterhöhung eines Bildes, wie sie in Abschnitt 4.2.1 vorgestellt wird, läßt sich streiten. Was aber gezeigt wurde, ist die Machbarkeit. Die Lesbarkeit der Regeln ist sicherlich leichter als eine Sigmoidfunktion. Auch ist eine Erweiterung der Regeln um weitere Anforderungen leichter lesbar. Der Ansatz von Arakawa [Ara96] zeigt, daß bestehende Ansätze auch mit Fuzzy-Ansätzen ergänzt werden können. Insgesamt muß abgewägt werden, ob eine Erweiterung um Fuzzy-Methoden im Bereich der Bildvorverarbeitung den Aufwand lohnt. Wie in Abbildung 4.2 gezeigt ist, kann eine Binarisierung mittels einer unscharfen Schwellenwertberechnung ein

[7] Die Anzahl c muß bekannt sein, die Länge der Kodierung wird dann $\lceil log_2 c \rceil$ gewählt.

besseres[8] Ergebnis im Vergleich zu einem Standardverfahren liefern, welches das Histogramm-Minimum als Schwellenwert wählt. Auf die Beschreibung und Klassifikation ist hier aus schon genannten Gründen nicht näher eingegangen worden, da dies sehr ausführlich in Kapitel 5 geschieht. Hier soll nur betont werden, daß in dem Bereich der unscharfen Wissensbeschreibung eine besondere Bedeutung zu sehen ist. Eine natürlichsprachliche Beschreibung mit unscharfen Interpretationen der Begriffe erlaubt dem Experten die Modellierung vorhandener Variabilitäten.

Der Einsatz neuronaler Methoden wurde ebenfalls in einigen Bereichen vorgestellt. Zellulare neuronale Netze haben aufgrund ihrer Struktur den Vorteil, direkt auf Bildpunktebene Berechnungen durchzuführen. Zunächst läßt sich am Beispiel von Abschnitt 4.3.1 nicht ersehen, warum die Funktion aus Gleichung (4.6) mittels eines Netzes berechnet wird. Der Vorteil dieser Art liegt aber in der Möglichkeit zur Adaption des Netzes durch Parameter-Modifikation (also im besonderen den Gewichten). Dadurch sind die CNN *programmierbar* und somit für verschiedene Aufgaben anwendbar. Für die Segmentierung können neuronale Netze besonders gut für den Bereich der Vorsegmentierung (ROI) eingesetzt werden, eine vernünftige Anwendung im Bereich der exakten Segmentierung läßt sich aber im medizinischen Bereich aufgrund mangelnder Verfügbarkeit ausreichender Trainingssätze fast ausschließen. Beschreibung und Klassifikation hingegen sind gut geeignet, mit neuronalen Netzen realisiert zu werden, da durch die höhere Abstraktionsebene die Trainingsmenge deutlich reduziert werden kann.

Die direkte Nutzung evolutionärer Algorithmen auf den Bilddaten ist nur ganz speziellen Anwendungen vorbehalten. Sie sollten orthogonal in der Anwendung der Bildverarbeitung gesehen werden, wie dies bspw. zur Optimierung verschiedener Parameter oder Faltungsmasken geschehen ist. Der Grund, warum eine Anwendung auf die Bilddaten nur wenig zweckvoll erscheint, liegt in der Größe des Suchraums einer möglichen Lösung, bei der die Segmentierung einer Region für ein Bild der Größe $M \times N$ bei $2^{M \cdot N}$ liegt.

[8] Was im Hinblick der Aufgabenstellung zu verstehen ist.

5 Ein CI-Modell für die medizinische Bildanalyse

Wie im vorherigen Kapitel beschrieben, finden sich in der Literatur sehr viele Ansätze, CI-Methoden in der (medizinischen) Bildverarbeitung anzuwenden. Hierbei handelt es sich jedoch meist um die Verwendung einer *einzelnen* der drei CI-Methoden für eine sehr spezielle Aufgabe. In diesem Kapitel wird in Abgrenzung zu diesen Arbeiten ein Rahmenmodell entwickelt, welches die Verwendung aller CI-Methoden in der medizinischen Bildverarbeitung beinhaltet. Dieses Modell wird **CIMMBA** (**CI**-**M**odell für die **M**edizinische **B**ild**A**nalyse, kurz CIM^{2}BA) genannt. Als konkreten Vorteil bietet das Rahmenmodell eine – wie sich in der Praxis gezeigt hat – sinnvolle Kombination der Methoden für die einzelnen Teile der Prozeßkette (vgl. Abbildung 2.3). Generell ist die Kombination in Verbindung mit der Modellierung des zuvor eingeführten unscharfen Wissens verbunden, wodurch ein recht vollständiges Instrumentarium zur Modellierung der genannten Problembereiche der Bildverarbeitung bereit steht. Natürlich ist auch dieses Modell darüber hinaus nicht ad hoc für alle Bildanalyseprobleme geeignet. Jedoch treten bei der Analyse medizinischer Bilddaten häufig gleiche oder ähnliche Probleme in Bezug auf die Variabilität in den Vordergrund, die sich mit Hilfe einer unscharfen Wissensbeschreibung gut erfassen lassen. So ist eine Anpassung an die konkret gegebene Aufgabenstellung in diesem Kontext normalerweise einfach durchführbar. Dies erfolgt spezifisch durch Trainieren der eingesetzten neuronalen Netze oder Optimieren der unscharfen Mengen mit Hilfe von Evolutionsstrategien. An dieser Stelle ist dann hautpsächlich die gewünschte Sollausgabe für das Netz bzw. die Fitneßfunktion für die Evolutionsstrategie festzulegen. Zudem ermöglicht der vorgeschlagene modulare Aufbau eine Anpassung an veränderte Anforderungen auf sehr einfache Weise.

Als Voraussetzung ist dennoch folgendes zu bedenken. Als Rahmenmodell hat CIM^{2}BA den Anspruch, flexibel und einfach um weitere, spezifische CI-Methoden erweiterbar zu sein. Dies wird durch die zugrundeliegende Architektur von CIM^{2}BA auch erreicht, wie sich im folgenden herausstellen wird. Allerdings besitzt das Rahmenwerk weder den Anspruch, für jeden Operator eine konkrete Ausprägung zu besitzen, noch erfordert es die spezifische Implementierung für einen Operator. Vielmehr können Operatoren für ein Problem durchaus unterschiedlich ausgeprägt werden. In der Arbeit wird oftmals eine konkrete Umsetzung vorgestellt und zudem darauf hingewiesen, welche anderen Wege denkbar sind.

5.1 Rahmenmodell

Der hier vorgeschlagene Rahmen baut auf dem Konzept modularer Komponenten auf, die jeweils elementare Grundprobleme der Bildverarbeitung lösen und somit oben genannte spezielle Lösungen integrieren können. Die Komponenten besitzen definierte Schnittstellen, über die sie Daten mit weiteren Komponenten austauschen können.

In der klassischen Bildverarbeitung existiert eine Vielzahl von einzelnen Operatoren, die auf Bilddaten arbeiten. Bilddaten werden entweder wieder in Bilddaten transformiert oder aber Informationen aus den Bilddaten extrahiert. Viele dieser Operatoren haben eine spezielle Aufgabe, für die sie immer wieder eingesetzt werden können (vgl. Kapitel 2). Die Einteilung von Bildverarbeitungsproblemen in kleine Teilprobleme hat sich wie bei vielen anderen Problemstellungen als besonders geeignet erwiesen. Auf monolithische Systeme wird zugunsten komponentenbasierter Entwicklungen verzichtet [Neu95, BRJ99, GT00].

Diese Idee wird im entwickelten Rahmenmodell aufgegriffen und um Methoden erweitert, die *zusätzlich* mit unscharfen Daten und unscharfem Wissen arbeiten und anhand von Beispielen generalisieren können. Dabei integrieren sich die neuen Methoden nahtlos in das gesamte Modell, welches klassische Verfahren weiter verwendet, wo diese einsetzbar sind. Dadurch erhält ein Entwickler als Anwender des Bildverarbeitungssystems die Möglichkeit, abhängig von der aktuellen Problemstellung seine Lösung auch mit Hilfe unscharfen Expertenwissens zu formulieren. Weiterhin sind Optimierungen dieser Wissensdarstellung automatisch durchführbar. Um dieses zu gewährleisten, wird ein Baukastensystem (Abbildung 5.1) zur Verfügung gestellt, aus dem einzelne Komponenten ausgewählt und um problemspezifische Teile ergänzt werden können, etwa um neues Expertenwissen oder neue Operatoren. Aber auch das während der Analyse gewonnene (temporäre) Wissen wird genutzt, um Parametereinstellungen vorzunehmen oder zu ändern. Dadurch wird ein streng linearer Ablauf (im Sinne eines einmaligen Durchlaufens der Prozeßkette) der Analyse zugunsten eines iterativen Prozesses aufgegeben. Diese Möglichkeit der regel- oder ergebnisabhängigen Steuerung des Ablaufes eines Analyseprozesses wird in Abbildung 5.2 dargestellt und verdeutlicht, daß die Prozeßkette wiederholt durchlaufen werden kann.

Die Nutzung von Wissen wird auch schon in früheren Arbeiten gefordert (etwa in [NB87, GW93]). In vielen Grundlagenbüchern über den Bereich Bildverarbeitung wird das Thema aber gänzlich gemieden (vgl. [KZ95, Jäh97]) oder nur am Rande erwähnt (z. B. in [Abm94]). Zudem gibt es auch Bücher zu speziell diesem Thema der wissensbasierten Bildverarbeitung [LE89, GB97].

Eine konkrete Beschreibung, wie genau dieses Wissen aber handzuhaben bzw.

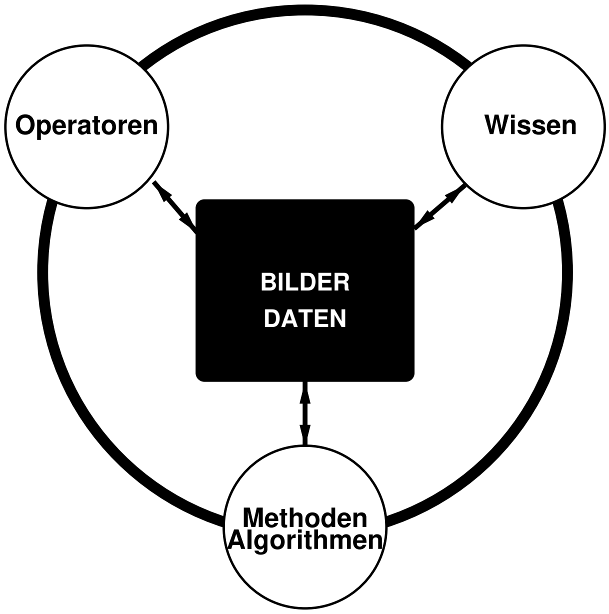


Abbildung 5.1: Baukasten von Methoden, Operatoren und Wissen zur Anwendung in der Bildverarbeitung.

wie mit den im Wissen inhärenten Unsicherheiten und Unschärfen umzugehen ist, wird dort nicht ausgeführt. Selbst ein spezielles Buch wie das von Graham und Barrett [GB97] beinhaltet lediglich zwei umfangreiche Kapitel über Wissensrepräsentationsmöglichkeiten und Grundlagen der Bildverarbeitung, während die eigentliche Kombination dieser Themengebiete dann nur sehr kurz angerissen wird. Die Anwendung von Fuzzy-Methoden wird nur als eine in der Zukunft sinnvolle Ergänzung erwähnt.

Das Ziel der vorliegenden Arbeit ist, sich diesem Problem anzunehmen und die bestehende Lücke zu schließen, um neben den Werken zur klassischen Bildverarbeitung auch einen Einblick in die Verwendung unscharfen Expertenwissens für diverse Problemstellungen zu bekommen. In Abbildung 5.2 ist das vorgeschlagene Rahmenmodell zu sehen. Hier wird gezeigt, an welchen Stellen sich die einzelnen CI-Methoden sinnvoll und effektiv in ein Gesamtsystem integrieren lassen.

Ziel dieses Rahmenmodells ist die umfangreiche Nutzung von Expertenwissen aus verschiedenen Bereichen. Zum einen ist das Wissen des Bildverarbeitungsexperten angesprochen, der auf das Rahmenmodell aufsetzen und es mit den anwendungsspezifischen Methoden und Operatoren befüllen kann, zum anderen das Wissen des Domänenexperten[1]. Da dieses Wissen nicht immer exakt formulierbar ist (vgl. Abschnitt 3.2.2), sind unscharfe Methoden, wie sie in Abschnitt 4.2 beschrieben werden, ein einfaches und adäquates Hilfsmittel, existierende Modelle

[1] In den hier beschriebenen Anwendungen also ein Mediziner.

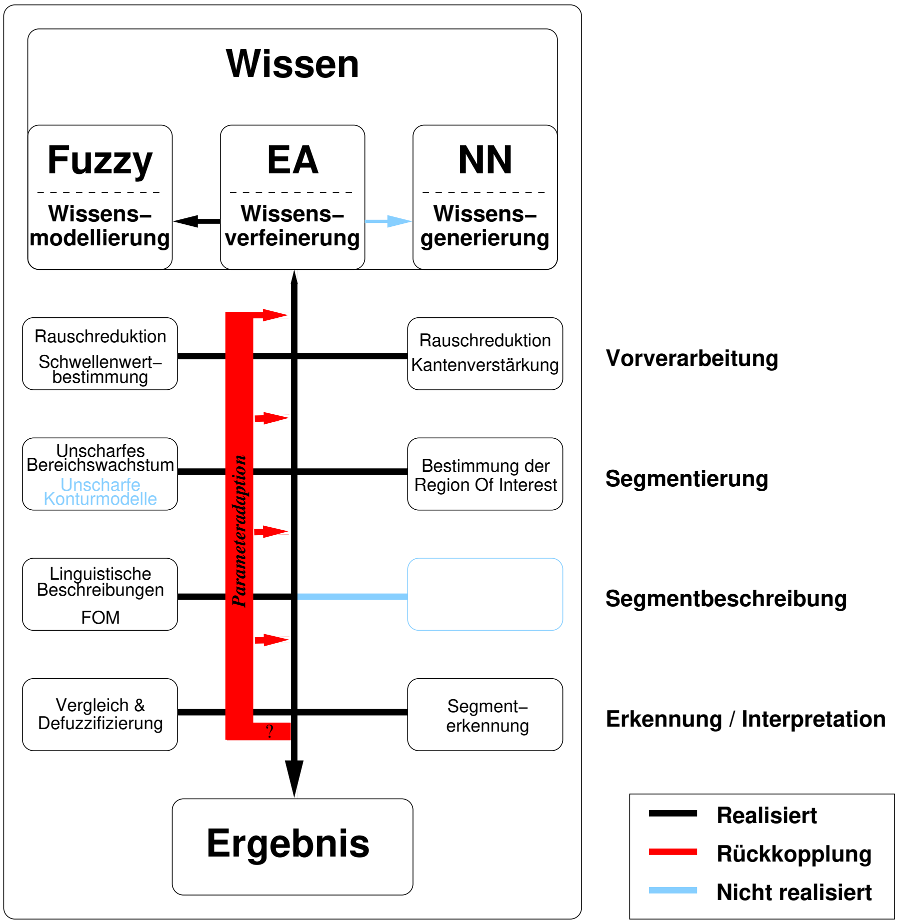


Abbildung 5.2: Prozeßkette im CI-basierten Rahmenmodell CIM²BA.

zu ergänzen. Falls das Wissen nicht verbalisierbar ist, kann durch Einsatz lernender Systeme anhand von Beispielen eine funktionale Beziehung zwischen Eingabewerten und gewünschten Ausgaben genutzt werden. Schließlich werden langwierige Optimierungsaufgaben weitestgehend automatisiert. Im folgenden wird das Standardmodell der Bildverarbeitung aus [GW93] um CI-Methoden erweitert. Zentraler Bestandteil von CIM²BA ist der modulare Aufbau, der eine einfache Erweiterung und Anpassung an neue Gegebenheiten erlaubt. Einzelne Komponenten, wie sie im weiteren vorgestellt werden, sind dann bei Bedarf beliebig austauschbar.

Die Idee des Baukastens ist schließlich die Realisierung verschiedener Anwendungen, in denen (unscharfes) Expertenwissen genutzt wird. Dazu wird die eigentliche Umsetzung der Bildanalyse im System CIM²BA/P (vgl. Kapitel 6) realisiert. Das Ergebnis ist ein Programm, für das noch eine geeignete grafische Oberfläche (engl. *graphical user interface*, GUI) entwickelt werden muß, welche die Interaktion mit dem Benutzer erlaubt und den eigentlichen Programm-Code vor diesem verbirgt (vgl. Abbildung 5.3). Die exemplarisch gezeigten Anwendungen sind Systeme von Projektgruppen [FT94, RFH98, HFR99] und einer Diplomarbeit [JM96] der Universität Dortmund, Fachbereich Informatik. Umgesetzt wurden Anomalia [RFH98] und DAWN [HFR99] mit einem Baukastenprototypen namens *Bambus* [FHJT97, RFH98, HFR99], während die anderen Systeme frühere Vorläufer des Baukastens waren, die zur Analyse von MRT-Bilddaten mit Hilfe unscharfen Wissens erstellt wurden. Diese Projekte wurden unter Mitwirkung und Leitung des Autoren dieser Arbeit realisiert und haben die Eignung der vorgestellten Methoden zur Segmentierung und Erkennung von Pathologien mit Hilfe von Fuzzy-Methoden bewiesen. Das Konzept der grafischen Programmierung, wie Bambus sie zur Verfügung stellt, hat sich aber als sehr zeitaufwendig herausgestellt. Das vorliegende Rahmenmodell beinhaltet somit ein weitaus unfassenderes CI-Modell und stellt mit der Realisierung von CIM²BA/P zudem eine vollständig eigene Implementierung des Rahmenmodells dar.

Im folgenden Teil werden vorwiegend theoretische und von der Implementierung unabhängige Methoden beschrieben, während das anschließende Kapitel 6 dann die Realisierung der Ideen beinhaltet und die Implementierung des Systems CIM²BA/P vorstellt.

5.2 Verwendung unscharfer Methoden in CIM²BA

Unscharfe Methoden lassen sich besonders gut auf der Modellierungsebene des Expertenwissens nutzen. Meistens läßt sich eine scharfe Abgrenzung der Eigenschaften eines beschriebenen Objektes nicht treffen, was speziell in der Medizin noch deutlicher zu beobachten ist. Wird bspw. das Kleinhirn eines Menschen in einem sagittalen MRT-Bild beschrieben, so besitzt es eine *mittlere Helligkeit*, einen *hohen Rundheitsgrad*, liegt *rechts vom* Hirnstamm und besitzt eine *homo-*

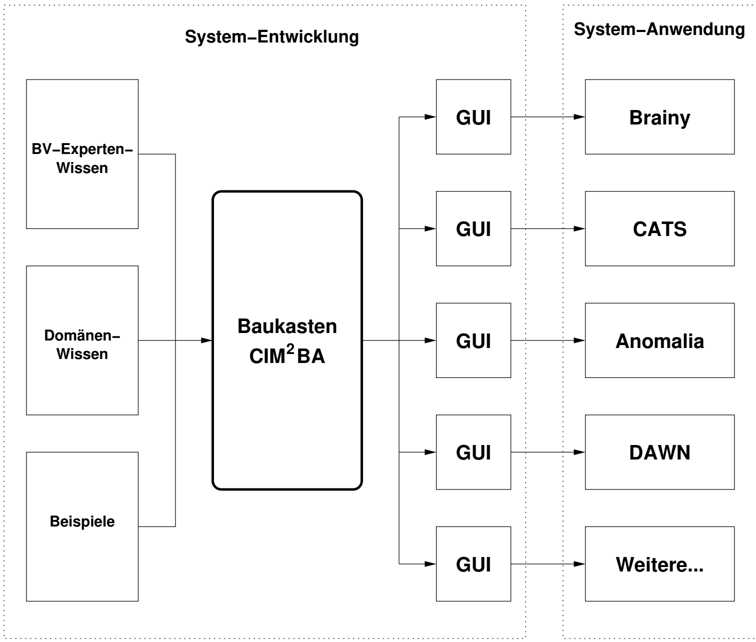


Abbildung 5.3: Grundidee der Verwendung des Baukastenprinzips für die Erstellung von Bildverarbeitungsanwendungen mit CIM^{2}BA.

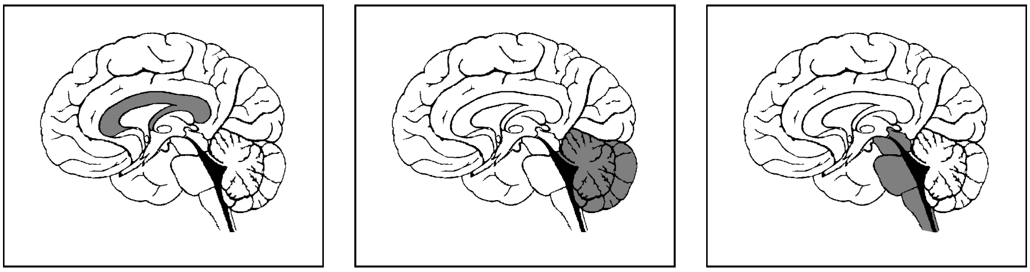
gene Struktur. Die Interpretation derartiger Beschreibungen sind mit klassischen (scharfen) Abgrenzungen nur schwer nachzubilden. Warum etwa der Grauwertbereich von 100 bis 130 eine mittlere Helligkeit besitzen soll, aber nicht die Grauwerte 99 und 131, ist kaum nachvollziehbar.

Im folgenden werden häufiger Strukturen des Gehirns als Beispiel genannt, weshalb diese in Abbildung 5.4 schematisch gezeigt werden. Hierbei handelt es sich um das Corpus Callosum, das Kleinhirn und den Hirnstamm sowie das Gehirn als Ganzes.

5.2.1 Wissensrepräsentation in CIM^{2}BA

Die umfangreiche Nutzung von Expertenwissen ist schon mehrfach in dieser Arbeit gefordert worden. Deshalb wird hier beschrieben, welche Art der Wissensrepräsentation an welchen Stellen im Rahmenmodell eingesetzt wird.

Für alle Wissensbestandteile besteht die Möglichkeit, diese in einem *Kontext* zu definieren, der einen *Gültigkeitsbereich* für die Interpretation festlegt. Während der Entstehung dieser Arbeit hat sich bei der Wissensakquisition herausgestellt, daß es nicht nur synonyme Begriffe für dieselbe Interpretation gibt, sondern auch, daß dieselben Begriffe (Namen, Bezeichner) mehrdeutig interpretiert werden können [FHTT96]. Die Einführung eines Kontextes erlaubt dem Wissensingenieur die intuitiv verwendeten Begriffe (linguistische Variablen) ge-



(a) Balken (Corpus Callosum)

(b) Kleinhirn (Cerebellum)

(c) Hirnstamm (Truncus Cerebri)

Abbildung 5.4: Darstellung der als Beispiele verwendeten Strukturen des Gehirns in Sagittalansicht [May99]. Die jeweilige Struktur wird grau dargestellt.

eignet zu interpretieren. Ist kein Kontext definiert, wird die Variable global interpretiert, wobei dann die Eindeutigkeit der Interpretation sichergestellt sein muß. Somit wird eine Namenseindeutigkeit vorausgesetzt. Die Definition 3.10 wird entsprechend erweitert:

Definition 5.1 (CIM^{2}BA-Linguistische Variable) Eine linguistische Variable $\mathcal{F}_\diamond$ in CIM^{2}BA wird durch ein 4-Tupel $(C, X, T, \mathbb{U})$ charakterisiert, wobei C der Kontext ist, in dem die Variable interpretiert wird. X ist der Bezeichner (Name) der Variablen und T die Menge der linguistischen Terme, deren Fuzzy-Mengen über dem Universum $\mathbb{U}$ definiert sind. □

Auch bei Konzepten wird die Verwendung eines Kontextes erlaubt, um eine Umgebung zu definieren, in der genutzte Begriffe (Variablen, Terme, Relationen) geeignet interpretiert werden. Als Beispiel sei hier ein Konzept *Gehirn* genannt. Falls in der FM-Basis Variablen, die im Konzept *Gehirn* verwendet werden, für verschiedene Kontexte definiert sind (etwa Erwachsener, Kind), so sind bei derselben Sprache andere Interpretationen der Begriffe notwendig. Durch den Kontext „*Erwachsener*.Gehirn“ kann eine Variable anders interpretiert werden als dieselbe Variable (genauer derselbe Variablenname) in einem Kontext „*Kind*“. Um eine geeignete Interpretation der Konzepte einer Wissensbasis zu ermöglichen, wird auch die Definition 3.19 geeignet erweitert:

Definition 5.2 (CIM^{2}BA-Konzept) Ein Konzept wird durch ein 4-Tupel *(C, N, E, I)* definiert, wobei C der Kontext ist, in dem das Konzept interpretiert wird und N den Namen des Konzeptes beschreibt. E ist die Menge aller Objekte, die zum Konzept gehören. I gibt die Eigenschaften an, die ein Objekt haben muß, um zu diesem Konzept zu gehören. □

5.2.1.1 Repräsentation von Faktenwissen

CIM²BA ermöglicht die Verwendung einer Wissensbasis, welche die Verwendung unscharfer Beschreibungen der Merkmale von und Relationen zwischen Segmenten zuläßt. Von den in Abschnitt 3.2 vorgestellten Methoden wurden im Rahmen der Forschungsarbeiten zur Segmentierung des Gehirns aus MRT-Daten und darauf aufbauenden Forschungsaktivitäten sowohl Frames [FT94, HFR99] als auch semantische Netze [HJMT99] getestet.

Aufgrund beliebiger Relationen zwischen den einzelnen Objekten haben sich die semantischen Netze hier als besonders sinnvoll erwiesen, um das Faktenwissen zu modellieren. Das hier entwickelte Modell CIM²BA verwendet jedoch eine hybride Wissensrepräsentation, die die Vorteile von frame-basierter und netzbasierter Wissensrepräsentation vereint (vgl. Abschnitt 3.2.4.4). Basis ist die Struktur eines semantischen Netzes, welches allerdings die direkte Angabe der Objekteigenschaften im Konzept erlaubt (und diese nicht in eigenen Eigenschaftskonzepten hinterlegt (vgl. Abschnitt 3.2.4.2)). Weiterhin besteht die Möglichkeit, Prozeduren zu integrieren (die somit mit den Dämonen in den Slots vergleichbar sind (vgl. Abschnitt 3.2.4.3)), um Vergleiche durchführen zu können oder zusätzliche Berechnungen zu starten.

Dabei werden alle Eigenschaften, die ein gesuchtes Objekt besitzt, als linguistische Variablen mit linguistischen Termen belegt. Da nicht alle Objekte alle Eigenschaften bzw. keine zur Differenzierung von anderen Objekten sinnvoll beschreibbare Eigenschaft haben[2], existieren auch sogenannte *Don't-Care*-Belegungen für die Variablen, alternativ kann die Angabe der Variablen entfallen. Auch das Prinzip der Vererbung ist realisiert, so daß bei einer *is-a*-Relation alle nicht überschriebenen Eigenschaften bzw. deren Werte aus dem Eltern-Knoten übernommen werden. Falls für ein Kind eine ererbte Eigenschaft nicht berücksichtigt werden soll, *muß* eine *Don't-Care*-Belegung erfolgen.

Warum die Beschreibung der Eigenschaften in unscharfer Form erlaubt sein soll, läßt sich an einem sehr einfachen Beispiel *Länglichkeit* erläutern. Falls in einem Bild *längliche* Objekte gesucht werden, so ist die Grenze im klassischen Fall, bis wann ein Objekt noch als länglich zu bezeichnen ist, *scharf* festzulegen. Dies wird in der Abbildung 5.5 durch die gestrichelte Linie angedeutet. Intuitiv ist es aber nicht einsichtig, daß das zweite Objekt noch länglich ist, das dritte Objekt aber nicht mehr. Die Verwendung unscharfer Methoden erlaubt in dieser Situation einer graduelle Abstufung, in wieweit dieses Objekt noch länglich ist. Dadurch wird ein Grad angegeben, mit dem das Objekt zur Menge der länglichen Objekte gehört. Auch das zu Beginn genannte Beispiel des Grauwertbereiches mittlerer Helligkeit fällt in diese Kategorie von Problemen. In Abschnitt 5.2.3 werden Berechnungsmöglichkeiten für verschiedene Eigenschaften vorgestellt, die zur unscharfen Beschreibung von gesuchten Objekten herangezogen werden können.

[2]Es macht z. B. wenig Sinn, für runde Strukturen eine Orientierung anzugeben.

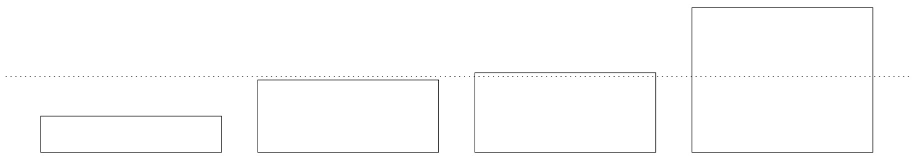


Abbildung 5.5: Problem der genauen Abgrenzung einer Eigenschaft hier am Beispiel, bis wann ein Objekt noch als „länglich“ gilt.

5.2.1.2 Repräsentation von Handlungswissen

Zur Parametereinstellung für die verschiedenen Operatoren haben sich sehr gut unscharfe Regelbasen erwiesen [HJMF97, HKH98]. Bspw. kann regelgesteuert die Auswahl bestimmter Operatoren für die Bildauswertung erfolgen. Auch lassen sich viele Operatoren über Eingabeparameter steuern. Wie die Parameter jedoch eingestellt werden müssen, ist nur schwer *exakt* formulierbar, so daß auch hier ein idealer Ansatzpunkt für die Ursprünge der Fuzzy-Logik, dem Fuzzy-Control, liegt.

CIM^{2}BA gestattet die Verwendung von Regeln zur Formulierung von Handlungswissen in einer Form, wie sie in Abschnitt 3.2.4.1 beschrieben und in Abbildung 6.12 gezeigt wird. Die Prämisse kann dabei aus mehreren Teilbedingungen, die per fuzzy-logischem *und* verknüpft werden, bestehen, während die Konklusion einstellig ist. Genutzt werden dürfen sowohl scharfe als auch unscharfe Variablen und Terme.

5.2.2 Unscharfes Bereichswachstumsverfahren

Für die Segmentierung, etwa durch ein Bereichswachstumsverfahren, wird unscharfes Wissen genutzt, um zu entscheiden, ob ein Punkt zum Segment hinzugefügt wird oder nicht. Ein weiterer Schritt ist der Verzicht auf Forderung nach Disjunktheit, d.h. der Forderung nach der *binären* Entscheidung, ob ein Punkt zu einer Region zugehörig gezählt wird oder nicht. Vielmehr werden die Bildpunkte mit einem Vertrauensgrad zu einem Segment hinzugefügt, so daß hier auch eine sich überlappende Segmentierung ermöglicht wird. Dies ist sinnvoll, da die Ortsauflösung bei der Bildgewinnung gering sein kann und durch den in Abschnitt 2.6 erwähnten Partialvolumeneffekt eine Verwischung bzw. Entfernung real existierender Kanten möglich ist. Somit müssen Bildpunkte in den Randgebieten der gefundenen Objekte nicht eindeutig zugeordnet werden. Die Idee der Repräsentation von Unsicherheit auf der Bildpunktebene wurde von Prewitt [Pre70] vorgeschlagen:

> *[...], a pictorial object is a fuzzy set which is specified by some membership function defined over all picture points. From this point of*

> *view, each image point participates in many memberships. Some of this uncertainty is due to degradation, but some of it is inherent. [...]*

und von Rosenfeld [Ros79] wieder aufgegriffen:

> *However, segmentation of picture into subsets represents a strong commitment; in many cases, it would be preferable to weaken this commitment by „extracting“ fuzzy subsets, rather than ordinary subsets, from the picture.*

Der Prozeß der Segmentierung wird in zwei Stufen durchgeführt, der Filterung und dem eigentlichen Bereichswachstum. Dabei kommt der Verwendung unscharfen Wissens eine herausragende Bedeutung zu [HJMT99].

Die Filterung dient zur Auswahl geeigneter Startpunkte des Bereichswachstums. Dazu beschreibt das Wissen die ungefähre Lage und den Grauwertbereich des gesuchten Objektes. Da sowohl Lage als auch Grauwert des gesuchten Objektes nur vage bekannt sind, kommen hier schon die unscharfen Beschreibungen direkter Bildeigenschaften[3] zum Tragen.

Aus jeder einzelnen unscharfen Beschreibung einer Eigenschaft wird aus dem Bild eine ikonische Fuzzy-Menge (vgl. Definition 3.17) [Men90] generiert. Diese beschreibt die Zugehörigkeit jedes Bildpunktes zu der Menge, die diese Eigenschaft besitzt, d.h. der Beschreibung entspricht. Abbildung 5.6 zeigt die Beschreibung des gesuchten Objektes mittels der Eigenschaften *Lage* und *Helligkeit* und die daraus resultierenden ikonischen Fuzzy-Mengen.

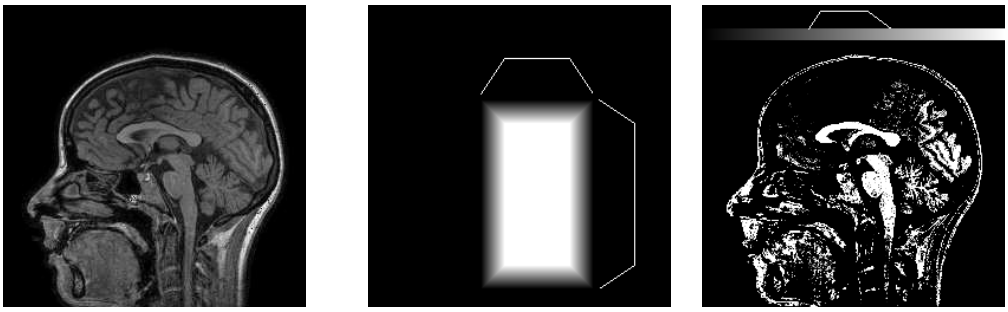
Der Filter $\mathcal{F}_{position}$, wie er exemplarisch in Abbildung 5.6(b) zu sehen ist, wird wie folgt berechnet, wenn die zylindrische Erweiterung der beiden eindimensionalen unscharfen Mengen miteinander verknüpft wird. Dazu ist die Definition der Lage über die horizontale Achse mit $\mu_{\mathcal{H}}$ und über die vertikale Achse mit $\mu_{\mathcal{V}}$ bezeichnet, wobei diese entsprechend Definition 6.1[4] (Seite 124) mittels vier Parametern als Trapez (Typ $\theta = 1$) definiert sind und τ die verwendete T-Norm ist.

$$\mathcal{F}_{position} \stackrel{\text{def}}{=} \tau(\mu_{\mathcal{V}2D_x}, \mu_{\mathcal{H}2D_y}) \tag{5.1}$$

Der Filter $\mathcal{F}_{brightness}$ kann mit der Bildfunktion I des Originalbildes sehr einfach ermittelt werden, wenn die Fuzzy-Menge $\mu_{\mathcal{B}}$ den gesuchten Grauwertbereich definiert:

[3] Unter direkten Bildeigenschaften sind Informationen aus dem Bild zu verstehen, die direkt ablesbar sind, also z. B. der Grauwert oder die Position.

[4] Diese Definition wird im folgenden Teil häufiger verwendet, besonders, wenn es um konkrete Ausprägungen einer Fuzzy-Menge geht. Da sie aber implentierungsspezifisch ist, wird sie erst in Kapitel 6 konkret vorgestellt. Neben der Definition 6.1 muß im folgenden auch die Festlegung 6.1 Berücksichtigung finden.



(a) Eingabebild

(b) Filter $\mathcal{F}_{position}$ der Eigenschaft *Lage*

(c) Filter $\mathcal{F}_{brightness}$ der Eigenschaft *Helligkeit*

Abbildung 5.6: Bestimmung ikonischer Fuzzy-Mengen aus einem Bild zur Beschreibung unscharfer Eigenschaften, die das gesuchte Objekt (hier der Hirnstamm) nach Definitionen in der Wissensbasis besitzen soll. In die Bilder werden die unscharfen Mengen zur Verdeutlichung mit eingeblendet.

$$\mathcal{F}_{brightness} \stackrel{\text{def}}{=} \mu_{\mathcal{B}}(I) \tag{5.2}$$

Neben dieser einfachen Lagedefinition sind auch andere, komplexere geometrische Formen denkbar, wie sie bspw. in Abschnitt 5.2.3.4 und Abschnitt 5.3.1 vorgestellt werden. Dies ist abhängig vom jeweiligen Wissen über die gesuchten Strukturen und der Möglichkeit zur Eingrenzung. Bei der Verwendung komplexerer Filter, die schon lokale Eigenschaften mit berücksichtigen, muß ein Kompromiß zwischen Aufwand und Nutzen getroffen werden, sofern dadurch die Anzahl möglicher Startpunkte vorzeitig begrenzt werden kann. Ist der Aufwand für diese Einschränkung so hoch, daß effektiv kein Gewinn dadurch erzielt wird, sollte auf einfachere Filter zurückgegriffen werden. Die Entscheidung ist vom Anwender des Systems zu treffen und aufgabenspezifisch, weshalb hier keine konkrete Aussage getroffen werden kann.

Jede dieser ikonischen Fuzzy-Mengen beschreibt eine Eigenschaft und stellt einen Filter $\mathcal{F}_i$ für die Eigenschaft i dar. Um einen „Gesamtfilter" $\mathcal{F}_{\&}$ zu generieren, werden die einzelnen Filter miteinander kombiniert. Da das gesuchte Objekt alle Eigenschaften gleichzeitig besitzen muß, wird eine verallgemeinerte Form der T-Norm[5], ein *T-Quantor* TQ_τ, verwendet (vgl. Definition 3.14).

$$\mathcal{F}_{\&} : M \times N \to \langle 0, 1 \rangle \text{ mit } \mathcal{F}_{\&}(x, y) = TQ_\tau\left(\mathcal{F}_1(x, y), \ldots, \mathcal{F}_n(x, y)\right) \tag{5.3}$$

[5]Eine T-Norm erlaubt nach Definition 3.11 nur zwei Funktionswerte.

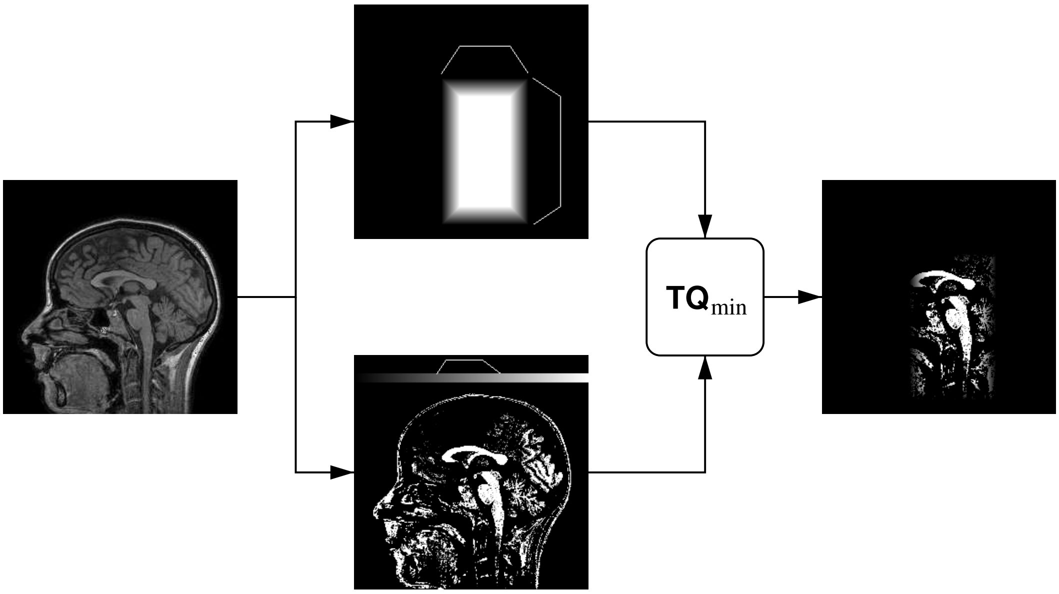


Abbildung 5.7: Anwendung des T-Quantors $TQ_{\min}$ auf die Filter $\mathcal{F}_{position}$ und $\mathcal{F}_{brightness}$ zur Berechnung eines Gesamt-Filters $\mathcal{F}_{\&}$, der zur Bestimmung der Startpunkte verwendet werden kann. Das Originalbild dient als Eingabe für den Lagefilter, um Größennormierungen durchführen zu können.

In Abbildung 5.7 wird ein Filter für die Segmentierung des Hirnstammes dargestellt. Die Eingabe des Originalbildes in den Lagefilter dient zur Normierung. Lage und Größe des Suchbereiches sind abhängig von Größennormierungen, die (in diesem Beispiel) anhand der Kopfgröße[6] durchgeführt werden. Um die möglichen Größenunterschiede zu verdeutlichen, sei an dieser Stelle nochmals auf Abbildung 2.13 verwiesen. Im genannten Beispiel werden die zwei Filter $\mathcal{F}_{position}$, der die ungefähre Lage des Objektes beschreibt, und $\mathcal{F}_{brightness}$, der die ungefähre Helligkeit angibt, verwendet. Die beiden Filter werden über einen T-Quantor $TQ_{\min}$ verknüpft.

In einem anschließenden Schritt beginnt die Auswahl der Startpunkte für das Bereichswachstumsverfahren (vgl. Abschnitt 2.3.2.3). Dazu werden nur Punkte betrachtet, deren Zugehörigkeit maximal ist. Dadurch wird, wie in Abbildung 5.7 zu sehen ist, eine deutliche Reduktion möglicher Punkte erreicht. Weiterhin darf es sich nicht um einen verrauschten Punkt handeln (im Vergleich zur Nachbarschaft). Als unscharfes Maß für die Verrauschtheit wird der gleiche Ansatz wie bei Arakawa [Ara96] verwendet. Zudem werden zusammenhängende Bereiche gleicher (maximaler) Zugehörigkeit als ein Startpunkt betrachtet.

Wenn die Startpunkte eingeschränkt worden sind, beginnt die Betrachtung der benachbarten Punkte, um zu überprüfen, ob diese zu der Region hinzugefügt wer-

[6] Umgekehrt kann auch statt des Filters das Bild normiert werden.

den dürfen. Dabei müssen die Punkte wiederum gewissen Bedingungen genügen. Ein Punkt p darf nur zu einer Region mit dem Startpunkt p_0 mit dem Grad der Verbundenheit $C_{p_0}(p)$ (vgl. Definition 5.5) hinzugefügt werden. Dazu muß zunächst der Begriff des Pfades eingeführt werden.

Definition 5.3 (Pfad) Ein Pfad $P(p_A, p_B)$ von p_A nach p_B ist eine Sequenz $(p_0 = p_A, p_1, \ldots, p_{n_p} = p_B)$ der Länge n_p von Bildpunkten, wobei zwei Punkte p_{i-1} und p_i für $\forall i \in \{1, \ldots, n_p\}$ auch im Bild benachbart sind[7]. Es gilt zudem $p_i \neq p_j$ für $i \neq j$.

$$p_i \in N_4(p_{i-1}) \text{ bzw. } p_i \in N_8(p_{i-1}) \quad \forall i \in \{1, \ldots, n_p\} \tag{5.4}$$

$\Pi_{p_1 \to p_2}$ bezeichnet die Menge aller Pfade $P_i(p_A, p_B)$ von p_A nach p_B. □

Definition 5.4 (Pfadgüte) Die Pfadgüte π_Q ist eine Funktion, die die Güte eines Pfades P_i bezüglich vorgegebener Eigenschaften f_j wie folgt bestimmt:

$$\pi_Q(P_i) \stackrel{\text{def}}{=} TQ_\tau\{f_1(P_i), ..., f_m(P_i)\} \tag{5.5}$$

$f_j(P_i)$ berechnet die Güte des Pfades bezüglich genau einer Eigenschaft j. □

Da die Güte eines Pfades durch alle betrachteten Eigenschaften bestimmt ist, wird ein T-Quantor verwendet. Dieses bedeutet, daß ein Pfad, der für eine betrachtete Eigenschaft j eine niedrige Güte aufweist, eine insgesamt niedrige Güte besitzt.

Definition 5.5 (Verbundenheit) Zwei Punkte p_A und p_B sind mit dem Grad $C_{p_A}(p_B)$ verbunden, wobei gilt:

$$C_{p_A}(p_B) \stackrel{\text{def}}{=} \max\{\pi_Q(P_i(p_A, p_B)) | \, P_i(p_A, p_B) \in \Pi_{p_A \to p_B}\} \tag{5.6}$$

□

Die Definition 5.5 besagt, daß zwei Punkte p_A und p_B zu der maximalen aller Pfadgüten π_Q verbunden sind. Es sei nochmals daran erinnert, daß die Berechnung der Verbundenheit zweier Punkte durchgeführt wird, um den Zugehörigkeitsgrad eines Punktes, der zu einer Region während des Bereichswachstums hinzugefügt wird, zu bestimmen. Im folgenden werden einige mögliche Eigenschaften j und deren Güteberechnung f_j, wie sie in Definition 5.4 eingeführt werden, angegeben.

Zunächst ist die Zugehörigkeit zum Filter $\mathcal{F}_\&$ (vgl. Gleichung (5.3)) ein entscheidendes Kriterium, da hier schon mehrere geforderte Eigenschaften an das gesuchte Objekt betrachtet werden (Lage und Grauwert).

[7] Hierbei handelt es sich eigentlich um zwei Definitionen, da der Pfad von der gewählten Nachbarschaft N_4 bzw. N_8 abhängig ist und somit ein N_4-Pfad bzw. ein N_8-Pfad existiert [Ros79].

Definition 5.6 (Filterzugehörigkeit) Die Funktion f_{member} berechnet die Güte eines Pfades P in Bezug auf die Zugehörigkeit der einzelnen Bildpunkte p_i des Pfades zum Filter $\mathcal{F}_\&$ wie folgt:

$$f_{member}(P) \stackrel{\text{def}}{=} TQ_\tau\{\mu_{\mathcal{F}_\&}(p_i) \mid p_i \in P\} \tag{5.7}$$

□

Weiterhin findet die Homogenität der Bildpunkte auf dem Pfad Einzug in die Bewertung der Güte. Dabei wird zwischen *lokaler* und *globaler* Homogenität unterschieden. Während die lokale Homogenität nur benachbarte Bildpunkte betrachtet, verwendet die globale Homogenität den Startpunkt als Referenz:

Definition 5.7 (Lokale Homogenität) Die Funktion f_{lh} berechnet die Güte eines Pfades P in Bezug auf die Grauwertdifferenz benachbarter Bildpunkte auf dem Pfad. $\mu_{\mathcal{LH}}$ definiert dazu die Zugehörigkeit zur lokalen Homogenität über die Menge der Grauwerte.

$$f_{lh}(P) \stackrel{\text{def}}{=} TQ_\tau \left\{\mu_{\mathcal{LH}}(\mid I(p_{i-1}) - I(p_i) \mid) \mid \forall i \in \{1, \ldots, n_p\}\right\} \tag{5.8}$$

□

Definition 5.8 (Globale Homogenität) Die Funktion f_{gh} berechnet die Güte eines Pfades P in Bezug auf die Grauwertdifferenz eines Bildpunktes p_i zum Startpunkt des Pfades p_A. $\mu_{\mathcal{GH}}$ definiert dazu die Zugehörigkeit zur globalen Homogenität über die Menge der Grauwerte.

$$f_{gl}(P) \stackrel{\text{def}}{=} TQ_\tau \left\{\mu_{\mathcal{GH}}(\mid I(p_i) - I(p_A) \mid) \mid \forall i \in \{1, \ldots, n_p\}\right\} \tag{5.9}$$

□

Generell gilt, daß die Güte eines Pfades (bezüglich eines Kriteriums) immer nur so hoch ist, wie die Güte seines „schwächsten“ Elements. Die Berechnung aller möglichen Pfade von p_A zu p_B ist in der Praxis sehr aufwendig. Allerdings kann hier die Eigenschaft genutzt werden, daß es sich bei der Bestimmung des Zugehörigkeitsgrades zum gesuchten Objekt um eine konvexe Funktion[8] handelt und ein Bildpunkt höchstens die Zugehörigkeit seines Nachbarn, über den der Punkt erreicht wird, besitzen kann.

Satz 5.1 *Das unscharfe Bereichswachstumsverfahren erzeugt ikonische Fuzzy-Mengen, deren α-Schnitte $\forall\alpha \in \langle 0, 1\rangle$ zusammenhängende Regionen sind.* □

[8] Konvex in dem Sinne, daß die α-Schnitte $\forall\alpha \in \langle 0, 1\rangle$ zusammenhängende Regionen sind.

Algorithmus: Unscharfes Bereichswachstumsverfahren

```
Initialisiere Segment S mit Startpunkt
Solange nicht-zugewiesene Punkte existieren, wiederhole
  Bestimme alle Nachbarn der Randpunkte von S
  Bestimme Grad Verbundenheit von allen Nachbarn
  Wenn es einen Nachbarn mit Zugehörigkeitsgrad > 0 gibt
    Füge den Nachbarn mit höchstem Zugehörigkeitsgrad zu S hinzu
  Sonst
    Terminiere
```

Abbildung 5.8: Kurzform des Algorithmus für das unscharfe Bereichswachstum.

Beweis zu Satz 5.1 Sei $\mathcal{S}$ eine ikonische Fuzzy-Menge, die aus dem Startpunkt p_s generiert wurde und $S^{\geq\alpha}$ ein α-Schnitt zu $\mathcal{S}$. Außerdem seien $p_1, p_2 \in S^{\geq\alpha}$ gegeben.

Angenommen, $S^{\geq\alpha}$ sei nicht zusammenhängend, dann gibt es keinen Pfad $P(p_1, p_2)$ mit $\pi_Q(P) \geq \alpha$. Es gibt jedoch einen Pfad $P_1(p_s, p_1)$ mit $\pi_Q(P_1) \geq \alpha$ und einen Pfad $P_2(p_s, p_2)$ mit $\pi_Q(P_2) \geq \alpha$. Sei zudem der zu $P_1(p_s, p_1)$ inverse Pfad ${P'}_1(p_1, p_s)$ gegeben. Da aufgrund der Kommutativität der T-Norm $\pi_Q(P_1) = \pi_Q({P'}_1)$ gilt, wähle den Pfad[9] $P_c = {P'}_1 p_s P_2$. Für $\pi_Q(P_c)$ gilt aber $\pi_Q(P_c) = \tau(\pi_Q(P_1), \pi_Q(P_2)) \geq \alpha$ im Widerspruch zur Annahme. ■

Aus diesem Grund ist es ausreichend, jeweils die Randpunkte der Region zu betrachten. Entsprechend wird bei dem Bereichswachstumsverfahren immer der nächste Randpunkt der aktuellen Region mit dem höchsten Zugehörigkeitsgrad gemäß Definition 5.5 hinzugefügt. Abbildung 5.8 zeigt den Algorithmus in Kurzform auf, in Abbildung 5.9 ist ein Beispiel für die Verwendung des unscharfen Bereichswachstumsverfahrens mit einigen Ergebnissen für die Suche des Hirnstammes dargestellt.

Im Gegensatz zu dem in Abschnitt 2.3.2.3 vorgestellten Bereichswachstumsverfahren ist das hier gezeigte unscharfe Verfahren nicht konkurrierend. Dies bedeutet, daß jeder Startpunkt eine Region erzeugen kann, die das gesamte Bild umfaßt. Lediglich der Grad der Zugehörigkeit entscheidet darüber, *wie sehr* ein Bildpunkt zu dieser Region gehört. Die Betrachtung von α-Schnitten (vgl. Definition 3.8) erlaubt es, aus den unscharfen Segmenten des hier vorgestellten unscharfen Bereichswachtumsverfahrens wieder scharfe Segmente zu generieren, deren Eigenschaften weiter betrachtet werden können. Ein Punkt gehört genau dann

[9] $P_c = P_i p_s P_j$ soll als Kurzschreibweise für eine Pfad von p_i nach p_j gelten, wenn P_i ein Pfad von p_i nach p_s und P_j ein Pfad von p_s nach p_j ist, wobei p_s nur einmal im Pfad vorkommt.

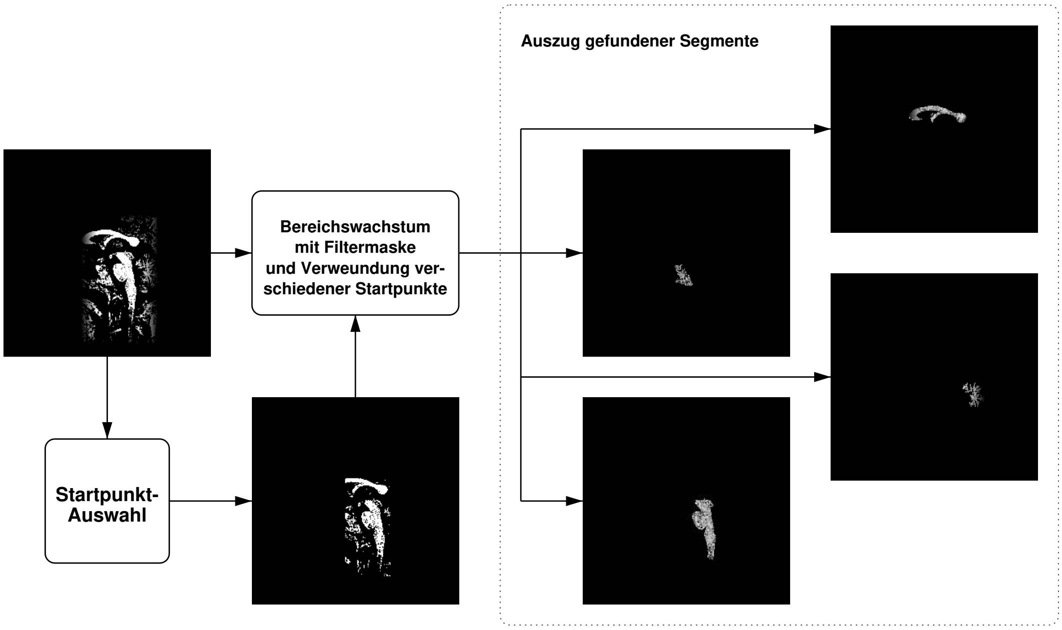


Abbildung 5.9: Beispiel für die Verwendung des unscharfen Bereichswachstums-verfahrens für die Segmentierung des Hirnstammes nach der Betrachtung unterschiedlicher Startpunkte. Gezeigt werden einige gefundene Segmente.

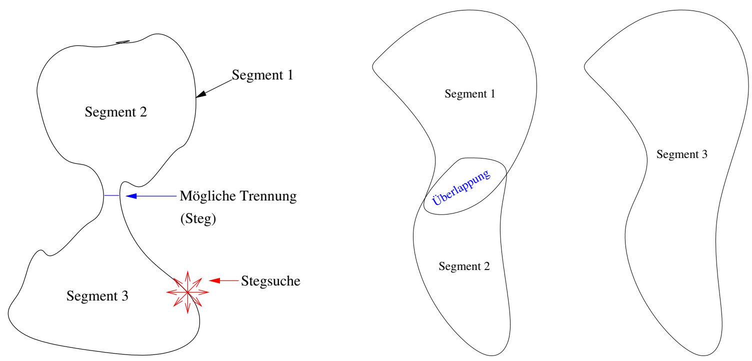
zu einem Segment, wenn der Zugehörigkeitsgrad aus dem unscharfen Bereichswachstumsverfahren größer Null ist.

5.2.2.1 Unscharfe Segmentzerlegung und Verschmelzung

Das Problem einer ggf. notwendigen Zerlegung oder Verschmelzung (Abbildung 5.10) von gefunden Segmenten (vgl. Abschnitt 2.3.2.3) besteht auch bei dem unscharfen Bereichswachstumsverfahren. Deshalb bietet CIM^{2}BA die Möglichkeit, im Anschluß an das Bereichswachstumsverfahren in CIM^{2}BA eine parametrisierte Zerlegung oder Verschmelzung[10] von Segmenten durchzuführen. Für die Verschmelzung zweier oder mehrerer Segmente wird eine S-Norm basierte Vereinigung der ikonischen Fuzzy-Mengen verwendet. Hierbei kann der Fall auftreten, daß die resultierenden unscharfen Segmente S nicht mehr für alle $S^{\geq\alpha}$ zusammenhängend sind. Um dies zu verhindern, wird die Verschmelzung $S_1 \uplus S_2$ zweier unscharfer Segmente S_1 und S_2 auf die Höhe des Durchschnitts begrenzt. Das Ergebnis berechnet sich aus:

$$\mu_{\mathcal{S}_1 \uplus \mathcal{S}_2} = \min(\sigma(\mu_{\mathcal{S}_1}, \mu_{\mathcal{S}_2}), height(\tau(\mu_{\mathcal{S}_1}, \mu_{\mathcal{S}_2}))) \tag{5.10}$$

[10] Eine Verschmelzung ist aufgrund der Forderung, daß ein Segment aus einer zusammenhängenden Region besteht, nur möglich, wenn der Durchschnitt der Punktmengen des Segmentes nicht leer ist.



(a) Zerlegung (Split) (b) Verschmelzen (Merge)

Abbildung 5.10: Zerlegung und Verschmelzung von Segmenten.

Ob und wann die Anwendung des Verschmelzens oder Zerlegens erfolgt, muß im Analyseablauf selbst entschieden werden (etwa abhängig von Parametern oder aber direkt bei der Auswahl der Bausteine für die Segmentierung). Bspw. können unmittelbar nach der Segmentierung mittels des Bereichswachstumsverfahrens alle gefunden Segmente auf eine mögliche Trennung untersucht werden, oder es kann in einem weiteren Iterationsschritt erfolgen, wenn die ermittelten Ergebnisse im ersten Durchlauf keine zufriedenstellenden Resultate liefern. Mit Hilfe unscharfer Regeln kann abhängig von den Segmenteigenschaften oder anderen Parametern eine Trennung erfolgen. Diese Regeln berechnen eine Stegbreite, bis zu der eine Trennung von Segmenten erfolgt. Betrachtet werden dabei z. B. die Größe des Segmentes selbst, die Größe des gesuchten Objektes und die Anzahl der Durchläufe, die bisher ohne Ergebnisse erfolgt sind. Wenn ein gefundenes Segment groß ist und das gesuchte Objekt klein, so ist bspw. eine Trennung des Segmentes sinnvoll und die Stegbreite, bis zu der getrennt werden darf, kann vergrößert werden. Ebenso kann sie bei wiederholten Durchläufen vergrößert werden.

Die Überprüfung, ob ein schmaler Steg existiert, erfolgt durch „Ablaufen“ der Kontur des zu trennenden Segmentes, wie dies in Abbildung 5.10(a) gezeigt wird. Für jeden Konturpunkt wird überprüft, ob im Umkreis der Stegbreite ebenfalls ein Konturpunkt des Segmentes liegt. Ist dies der Fall, wird das Segment an dieser Stelle aufgetrennt.

5.2.3 Unscharfe Beschreibungen

Medizinische Bilddaten beinhalten Strukturen, deren Eigenschaften nicht scharf abgrenzbar sind. Um derartige Strukturen finden bzw. klassifizieren zu können, müssen Variabilitäten und Toleranzen in den Beschreibungen erlaubt sein.

Es gibt eine Vielzahl von Eigenschaften, die für die Klassifikation einer Struktur herangezogen werden können. Diesen Eigenschaften können Eigenschafts*werte* zugeordnet werden, anhand derer die Zugehörigkeit zu unscharfen Termen bestimmt wird. Die Terme werden in einer Wissensbasis als eine geforderte Eigenschaft an das gesuchte Objekt hinterlegt.

Die Interpretation der Eigenschaften muß jedoch festgelegt werden und ist der Wissensbasis entsprechend zu wählen. Bspw. kann für die Eigenschaft *Rundheit* das Verhältnis von Fläche zum Umfang gewählt [LE89] werden oder aber z. B. die Ähnlichkeit zu einem Kreis[11] [HJMT99]. An dieser Stelle sei ein Zitat von Anaïs Nin genannt: *We don't see things as they are, we see things as we are*, womit treffend beschrieben wird, daß das Verständnis und somit auch die Beschreibung von Dingen inter-individuell unterschiedlich sein kann. Daraus resultiert die Problematik, daß natürlichsprachlich verwendete Begriffe korrekt interpretiert werden müssen, um den semantischen Sachverhalt, der in der Wissensbasis beschrieben ist, korrekt zu erfassen. Im folgenden Abschnitt wird deshalb eine umfangreiche Liste an Interpretation der verwendbaren Begriffe gegeben.

Die Erweiterung um neue Beschreibungen und dazugehörigen Interpretationen ist aufgrund des modularen Aufbaus möglich, die Auswahl der hier vorgestellten Interpretationen ist aus der Anwendung der Segmentierung vom MRT-Bilddatensätzen entstanden.

5.2.3.1 Unscharfe Eigenschaften

Es folgt eine Menge von Eigenschaftsdefinitionen, die sich für die Klassifikation medizinischer Strukturen als besonders geeignet erwiesen haben, wie dies auch in Kapitel 7 deutlich wird. Deshalb werden sie hier mit der dazugehörigen Berechnung, also deren Interpretation[12], angegeben. Der besondere Vorteil bei der hier beschriebenen Vorgehensweise der Verwendung von unscharfen Methoden ist die Nutzung bekannter und etablierter Eigenschaftsberechnungen, wodurch die Entwicklung neuer Methoden oder Beschreibungen nicht notwendig ist. Viele der Objekteigenschaften sind in dieser oder ähnlicher Form in [DH73, Nie83, Hab95] zu finden, während einige neue Lagerelationen in [HJMT99] beschrieben sind.

Für die Ermittlung einiger Eigenschaften wird die Kontur der zu bewerteten Struktur benötigt. Sie wird für ein Segment S entsprechend mit K_S bezeichnet und wie in Abschnitt 2.3.3.1 beschrieben als Richtungsketten-Code dargestellt. Die Transformation von einer Flächendarstellung in eine Konturdarstellung (bzw. vice versa) ist mittels eines Operators *contour* möglich (vgl. [KZ95]) und wird deshalb hier nicht dargestellt. Es ist allerdings zu berücksichtigen, daß bei der Konturdarstellung die Grauwertinformationen verloren gehen.

Da hier als Segmente auch ikonische Fuzzy-Mengen zugelassen sind, ist eine

[11] Unter einem Kreis wird in dieser Arbeit eine ausgefüllte Kreisscheibe verstanden, ansonsten wird von einem Ring gesprochen (wenn etwa Löcher erlaubt sind).

[12] Die Berechnung der Eigenschaftswerte muß zur Interpretation der Terme passen, weshalb hier ebenfalls der Begriff *Interpretation* gewählt wird.

Erweiterung sowohl des Segment- als auch des Konturbegriffs notwendig, die sich auf den Zugehörigkeitsgrad eines Bildpunktes zu diesem Segment bezieht. Für Eigenschaften, die die Größe oder den Umfang des Segmentes verwenden, können ansonsten keine Eigenschaftswerte berechnet werden.

Definition 5.9 (α-Segment) Das α-Segment $S^{\geq\alpha}$ eines Segmentes S, welches durch eine ikonische Fuzzy-Menge $\mu_{\mathcal{S}}$ über $M \times N$ dargestellt ist, wird definiert durch:

$$S^{\geq\alpha} \stackrel{\text{def}}{=} \{(x,y) \mid \mu_{\mathcal{S}}(x,y) \geq \alpha \quad \forall (x,y) \in M \times N\} \tag{5.11}$$

Ein α-Segment muß aus einer Menge zusammenhängender Punkte bezüglich einer gewählten Nachbarschaft (N_4 bzw. N_8) bestehen. □

Definition 5.10 (α-Kontur) Die α-Kontur $K_S{}^{\geq\alpha}$ eines Segmentes S, welches durch eine ikonische Fuzzy-Menge $\mu_{\mathcal{S}}$ dargestellt ist, wird definiert durch:

$$K_S{}^{\geq\alpha} \stackrel{\text{def}}{=} contour(S^{\geq\alpha}) \tag{5.12}$$

□

In den folgenden Definitionen wird auf die Differenzierung zwischen der Betrachtung von Konturen unterschiedlicher Segmente (Bildsegmente S und α-Segmente $S^{\geq\alpha}$) verzichtet, da für die Berechnung selbst scharfe Segmente betrachtet werden. Jedoch ist in der Bewertung der Güte eines Segmentes zu beachten, welchen Ursprung sie haben. Für die Berechnung einzelner Eigenschaften anhand der Kontur K_S wird davon ausgegangen, daß eventuell vorhandene Steuer-Codes (vgl. Abschnitt 2.3.3.1) bei der Bestimmung der Länge der *Zeichenkette* nicht berücksichtigt werden. Dies leistet die Funktion *strlen*. Hierbei handelt es sich bei K_S um eine Zeichenkette von Richtungssymbolen aus $\{0, \ldots, 7\}$. Ebenso wird auf die Angabe von Rundungen verzichtet und stattdessen die Funktion $round(x) = \lfloor x + \frac{1}{2} \rfloor$ implizit angenommen, um eine Abbildung in den korrekten Wertebereich $\mathbb{W}$ zu gewährleisten, falls $\mathbb{W} \subseteq \mathbb{Z}$ gilt.

Festlegung 5.1 (Bildpunktmaße) Im folgenden werden die Variablen d_x und d_y verwendet. Diese entsprechen den metrischen Ausmaßen einer Bildpunkteinheit entsprechend des in Abbildung 2.1 festgelegten Koordinatensystems (bei Voxeln existiert zusätzlich noch d_z als Tiefemaß). □

Definition 5.11 (Umfang) Ausgehend von der Kontur des Segmentes K_S mit den Elementen k_{i_S} berechnet sich der Umfang von S entsprechend der geforderten Exaktheit wie folgt:

1. Einfache Berechnung (nicht metrisch)

$$perimeter_{fast}(S) = strlen(K_S) \tag{5.13}$$

2. Genaue Berechnung (nicht metrisch)

$$perimeter_{exact}(S) = strlen\{even(K_S)\} + strlen\{odd(K_S)\} \cdot \sqrt{2} \quad (5.14)$$

3. Genaue Berechnung (metrisch)

$$\begin{aligned} perimeter_{metric}(S) \quad = \quad & d_y \cdot strlen\left(\{i \mid k_{i_S} \in \{0,4\} \wedge k_{i_S} \in K_S\}\right) + \\ & d_x \cdot strlen\left(\{i \mid k_{i_S} \in \{2,6\} \wedge k_{i_S} \in K_S\}\right) + \\ & \sqrt{d_x^2 + d_y^2} \cdot strlen\left(\{i \mid k_{i_S} \in \{1,3,5,7\} \wedge k_{i_S} \in K_S\}\right) \end{aligned} \quad (5.15)$$

□

Auch die Betrachtung des Umfangs als Summe aller Außenflächen (das heißt, ein einzelner Bildpunkt hat den Umfang 4) ist denkbar, hat allerdings in der diskreten Auflösung den Nachteil, daß ein Kreis mit Durchmesser d den gleichen Umfang ($4 \cdot d$) hat wie ein Quadrat mit der Kantenlänge d. Dadurch ergeben sich weitere Konsequenzen, da ein Quadrat dann als kompakter gilt als ein Kreis (gleichen Umfangs bzw. gleicher Fläche) (vgl. Definition 5.17), was wenig intuitiv ist und somit einer natürlichsprachlichen Beschreibung widerspricht.

Die Größe (Fläche) eines Segmentes läßt sich ebenfalls unterschiedlich genau berechnen:

Definition 5.12 (Größe) Sei S das Segment mit $card(S)$ Bildpunkten. Dann gilt:

1. Einfache Berechnung (nicht metrisch)

$$size_{fast}(S) = card(S) \quad (5.16)$$

2. Genaue Berechnung (metrisch)

$$size_{metric}(S) = card(S) \cdot d_x \cdot d_y \quad (5.17)$$

□

Definition 5.13 (Helligkeit) Die Helligkeit eines Segmentes S mit den Bildpunkten $p_i \in S$ entspricht dem durchschnittlichen Grauwert. Besteht das Segment S aus n Bildpunkten und beschreibt I die Bildfunktion, dann gilt:

$$brightness(S) = \frac{1}{n} \sum_{p_i \in S} I(p_i) \quad (5.18)$$

Auch die Verwendung einer *LUT* (vgl. Abschnitt 2.3.1) ist möglich, um z. B. bekannte Grauwertfehler in der Helligkeitsberechnung zu vermeiden:

$$brightness_{LUT}(S) = \frac{1}{n} \sum_{p_i \in S} LUT\left(I(p_i)\right) \quad (5.19)$$

□

Definition 5.14 (Flächenschwerpunkt) Der Flächenschwerpunkt (geometrischer Schwerpunkt) COG_{area} eines Segmentes S mit den Bildpunkten $p_i \in S$ berechnet sich wie folgt:

$$COG_{area}(S) = \left(d_x \cdot \frac{\sum\limits_{(x_i,y_i)\in S} x_i}{size_{fast}(S)}, d_y \cdot \frac{\sum\limits_{(x_i,y_i)\in S} y_i}{size_{fast}(S)} \right) \tag{5.20}$$

□

Definition 5.15 (Massenschwerpunkt) Der Massenschwerpunkt COG_{mass} eines Segmentes S mit den Bildpunkten $p_i \in S$ berechnet sich wie folgt:

$$COG_{mass}(S) = \left(d_x \cdot \frac{\sum\limits_{(x_i,y_i)\in S} x_i \cdot I(x_i, y_i)}{\sum\limits_{(x_i,y_i)\in S} I(x_i, y_i)}, d_y \cdot \frac{\sum\limits_{(x_i,y_i)\in S} y_i \cdot I(x_i, y_i)}{\sum\limits_{(x_i,y_i)\in S} I(x_i, y_i)} \right) \tag{5.21}$$

□

Auch die Eigenschaft *Rundheit* läßt sich verschiedenartig definieren, wobei allerdings hier die Mehrdeutigkeit weniger in der Genauigkeit als vielmehr in der Interpretationsmöglichkeit des Begriffs selbst zu finden ist. Welche der möglichen Interpretationen verwendet wird, ist abhängig von der Aufgabenstellung und dem in der Wissensbasis hinterlegten Wissen.

Bemerkung 5.1 Betrachtungen der metrischen Ausmaße eines Bildpunktes sind für die folgenden Merkmale von hoher Bedeutung, da bei $d_x \neq d_y$ für die korrekte Interpretation eine einsprechende Anpassung notwendig ist. Sofern diese nicht schon in der Wissensbasis berücksichtigt wird. Eine dortige Berücksichtigung erfordert dann aber eine rotationsabhängige Beschreibung, was in Abhängigkeit von der Aufgabenstellung zu beachten ist. □

Definition 5.16 (Rundheit) Das Maß $roundness_C$ beschreibt die Rundheit eines Segmentes S durch den Vergleich zu einem Kreis C, dessen Fläche $size_{metric}(C)$ der Fläche $size_{metric}(S)$ des Segments entspricht. Der Vergleich wird durch die Differenzpunkte der Überlagerung von Segment S und C im Schwerpunkt des Segmentes und Mittelpunkt des Kreises C bestimmt. Um ein größeninvariantes Rundheitsmaß zu erhalten, wird das Maß normiert.

$$radius_C(S) = \sqrt{\frac{size_{metric}(S)}{\pi}} \tag{5.22}$$

$$roundness_C(S) = 1 - \frac{size_{metric}(S \cap \overline{C}) + size_{metric}(\overline{S} \cap C)}{2 \cdot size_{metric}(S)} \tag{5.23}$$

Das Maß $roundness_R$ berechnet die Rundheit eines Segmentes S anhand von Längenverhältnissen von Strahlen R_i vom Schwerpunkt des Segmentes zur äußersten Kontur in definierten Winkelabständen φ ausgehend von der Orientierung des Segments. Das heißt, es werden $n = \frac{360}{\varphi}$, $\varphi > 0$ „Abtastungen" durchgeführt. Das Verhältnis von der durchschnittlichen Länge der Strahlen zum längsten Strahl definiert den Grad der Rundheit. Die Qualität des Ergebnisses ist abhängig vom gewählten φ. Alternativ kann die Kontur K_S in n äquidistanten[13] Bildpunkten abgetastet werden.

$$roundness_R(S) = \frac{\frac{1}{n}\sum\limits_{i=0}^{n-1} length(R_i)}{\max(length(R_i))} \tag{5.24}$$

□

Das Rundheitsmaß $roundness_R$ liefert bei Segmenten, deren Schwerpunkt außerhalb der äußeren Hülle[14] des Segmentes liegt, keine sinnvollen Ergebnisse. Dies ist bei der Anwendung zu berücksichtigen. Allerdings können mit diesem Maß wiederum auch Ringe als runde Objekte klassifiziert werden.

Definition 5.17 (Kompaktheit) Das Maß für die Kompaktheit *compactness* eines Segmentes S läßt sich aus dem Verhältnis der Fläche $size_{metric}(S)$ zum quadrierten Umfang $perimeter_{metric}(S)$, welches normiert wird[15], berechnen.

$$compactness(S) = \min\left(1.0, \frac{4 \cdot \pi \cdot size_{metric}(S)}{perimeter_{metric}(S)^2}\right) \tag{5.25}$$

□

Die Minimum-Funktion in Gleichung (5.25) dient zum Ausgleich von Fehlern bei sehr kleinen Objekten, damit das Ergebnis in $\langle 0, 1 \rangle$ liegt.

Die Kompaktheit, also das Verhältnis von Fläche zum quadrierten Umfang eines Segmentes S, kann auch zur Berechnung der Länglichkeit herangezogen werden: Die Kompaktheit nimmt bei sinkender Fläche und gleichem Umfang ab, so daß ein nicht kompaktes Segment als länglich interpretiert werden kann. Allerdings ist dieses Maß nur für konvexe Objekte sinnvoll einsetzbar, da schon ein Quadrat ein anderes Länglichkeitsmaß hat als ein Kreis, was wenig intuitiv ist.

[13] Äquidistant in Bezug auf die Anzahl der Bildpunkte, nicht auf deren euklidischen Abstand.

[14] Die äußere Hülle wird durch den *äußeren* Rand eines Segmentes gebildet, welches ggf. auch Löcher haben kann.

[15] In der Literatur, etwa [GW93], wird häufig das umgekehrte Verhältnis (also Umfang^2 zu Fläche) betrachtet. Dann besitzt ein Kreis den minimal möglichen Kompaktheitswert. Da dieses aber in Bezug auf die Zugehörigkeitswerte zur Menge kompakter Objekte nicht intuitiv ist, wird für die Interpretation des Begriffs das hier beschriebene Verhältnis betrachtet. Ein Kreis besitzt dann eine maximale Kompaktheit mit dem Wert 1.

Eine andere Berechnungsmöglichkeit ist über das Verhältnis der beiden Hauptachsen einer umschreibenden Ellipse möglich. Die längere der beiden Achsen kann auch zur Bestimmung der Orientierung verwendet werden. Sind beide Achsen gleich lang, besitzt das Segment keine ausgeprägte Orientierung. Die Notationen folgender zwei Beschreibungen sind an [Hab95] angeleht.

Es gelten dazu folgende Hilfsvariablen, wobei (x_{cp}, y_{cp}) den Flächenschwerpunkt des Segmentes S beschreiben (vgl. Definition 5.14). Die Korrekturberechnung gegebenenfalls unterschiedlicher Längenverhältnisse in horizontaler und vertikaler Richtung mittels d_x und d_y wird hier vernachlässigt, um eine einfachere Darstellung zu ermöglichen.

$$v_{xx} = \frac{\sum_{(x_i,y_i)\in S} (x_i - x_{cp})^2}{size_{fast}(S)} \tag{5.26}$$

$$v_{yy} = \frac{\sum_{(x_i,y_i)\in S} (y_i - y_{cp})^2}{size_{fast}(S)} \tag{5.27}$$

$$v_{xy} = v_{yx} = \frac{\sum_{(x_i,y_i)\in S} (x_i - x_{cp}) \cdot (y_i - y_{cp})}{size_{fast}(S)} \tag{5.28}$$

$$\lambda_1 = \frac{(v_{xx} + v_{yy}) + \sqrt{(v_{xx} - v_{yy})^2 + 4 \cdot v_{xy} v_{yx}}}{2} \tag{5.29}$$

$$\lambda_2 = \frac{(v_{xx} + v_{yy}) - \sqrt{(v_{xx} - v_{yy})^2 + 4 \cdot v_{xy} v_{yx}}}{2} \tag{5.30}$$

Definition 5.18 (Orientierung) Die Orientierung *orientation* eines Segmentes S berechnet sich wie folgt:

$$orientation(S) = \arctan\left(\frac{\lambda_1 - v_{xx}}{v_{xy}}\right) \tag{5.31}$$

□

Definition 5.19 (Länglichkeit) Die Länglichkeit *elongateness* eines Segmentes S berechnet sich aus dem Verhältnis der beiden Hauptachsen λ_1 und λ_2 einer umschreibenden Ellipse wie folgt:

$$elongateness(S) = 1 - \frac{\min(\lambda_1, \lambda_2)}{\max(\lambda_1, \lambda_2)} \tag{5.32}$$

□

Definition 5.20 (Konvexität) Das Maß für die Konvexität *convexity* eines Segmentes S läßt sich aus dem Verhältnis der Fläche des Segmentes $size_{metric}(S)$ zur Fläche der konvexen Hülle $size_{metric}(convexhull(S))$ berechnen. Der Wert wird maximal ($= 1$), wenn das Segment konvex ist.

$$convexity(S) = \frac{size_{metric}(S)}{size_{metric}\left(convexhull(S)\right)} \tag{5.33}$$

□

Definition 5.21 (Schmalheit) Ein Segment S wird als schmal bezeichnet, wenn das Maximum aller Radien r_{inside} eines Kreises C_I, der vollständig im Segment liegt, klein ist. Zur Normierung des Maßes wird der kleinste Radius $r_{outside}$ eines Kreises C_O gewählt, der das Segment vollständig enthält.

$$narrowness(S) = \frac{\max(\{r_{inside} | r_{inside} = radius(C_I) \wedge C_I \subseteq S\}}{\min(\{r_{outside} | r_{outside} = radius(C_O) \wedge S \subseteq C_O\}} \tag{5.34}$$

□

Die Homogenität eines Segmentes beschreibt die Gleichmäßigkeit der Grauwertverteilung. Diese kann unter Berücksichtigung des Bezugswertes wieder in eine lokale und globale Homogenität differenziert werden.

Definition 5.22 (Homogenität) Die globale Homogenität $homogeneity_{global}$ eines Segmentes S berechnet sich aus der Abweichung der Grauwerte von S in Relation zum Grauwertbereich mit Definition 5.13 wie folgt:

$$homogeneity_{global}(S) = 1 - 2 \cdot \frac{\sqrt{\sum\limits_{p_i \in S} (I(p_i) - brightness(S))^2}}{n \cdot (G_{\max} - G_{\min})} \tag{5.35}$$

Die Funktion $homogeneity_{local}$ berechnet die Homogenität in einem lokalen 3×3-Fenster, wobei über alle Werte gemittelt wird. Dabei muß beachtet werden, daß bei Randpositionen des Segmentes die Fensterwerte nicht existieren müssen. Hier wird das Fenster entsprechend der Kontur angepaßt. Für ein W_{p_i} mit $W_{p_i} = (N_8(p_i) \cup p_i) \cap S$, welches ein dem Segment angepaßtes 3×3-Fenster entspricht, gilt:

$$homogeneity_{local}(S) = \frac{\sum\limits_{p_i \in S} \left(1 - 2 \cdot \frac{\sqrt{\sum\limits_{p_j \in W_{p_i}} (I(p_j) - brightness(W_{p_i}))^2}}{card(W_{p_i}) \cdot (G_{\max} - G_{\min})}\right)}{size_{fast}(S)} \tag{5.36}$$

□

5.2.3.2 Unscharfe Relationen

Neben den bisherigen Eigenschaften wird auch die Lage (Position) eines Segmentes als Merkmal betrachtet. Hierzu kann sowohl die absolute Position im Bild wie auch die relative Lage zu anderen Segmenten dienen.

Als absolute Lagebeschreibung wird mit Hilfe unscharfer Beschreibungen und dazugehörigen unscharfen Mengen ein Filter wie $\mathcal{F}_{position}$ (Gleichung (5.1)) herangezogen. Dadurch sind Beschreibungen der Art

- liegt links im Bild
- liegt rechts im Bild
- liegt oben im Bild
- liegt unten im Bild

oder Kombinationen daraus möglich, ebenso direkte Beschränkungen über einen unscharfen Raum (vgl. Abbildung 5.6(b)). Für die Klassifikation sind aber besonders auch Relationen zwischen den Segmenten notwendig, da einige Strukturen einfacher segmentierbar und klassifizierbar sind als andere. Diese grenzen den Bereich für die Suche nach anderen Strukturen deutlich ein. Hierbei ist aber nicht nur die relative Lage wichtig, auch weitere Eigenschaften, wie etwa die Helligkeit, werden betrachtet.

Die folgenden Relationen zwischen zwei Objekten O_A und O_B haben sich für die bisherige Praxis in der MRT-Analyse als besonders sinnvoll erwiesen und können sowohl scharf als auch unscharf formuliert werden. Ein Teil davon wurde etwa in [JM96] implementiert und angewandt. Begriffe in eckigen Klammern dienen als eine Alternative, allen Relationen kann ein (ggf. unscharfer, maximaler bzw. minimaler) *Abstand* als Parameter übergeben werden.

- O_A liegt [$links|rechts|oben|unten$] von O_B.
- O_A liegt total [$links|rechts|oben|unten$] von O_B.
- O_A ragt [$links|rechts|oben|unten$] über O_B.
- O_A ragt total [$links|rechts|oben|unten$] über O_B.
- O_A liegt [$mindestens|höchstens$] [m, n] [Punkte|mm] entfernt von O_B.
- O_A ist [$heller|dunkler$] als O_B.
- O_A ist Teil von O_B.
- O_A liegt innerhalb von O_B.

Der Unterschied zwischen der „normalen“ und der „totalen“ Lagerelation liegt darin, daß eine Total-Relation sich auf den entsprechend *äußersten* Objektpunkt des Referenzobjektes bezieht, während sich die anderen (nicht totalen) Relationen auf die gesamte Kontur des Referenzobjektes beziehen.

Ob eine Relation ihr „umgangssprachlich semantisches Gegenstück“ durch eine Negation erreichen kann, ist abhängig von den gewählten Interpretationen von Relation und Negation der Relation. Bspw. muß die Relation „O_A ist heller als O_B“ nicht automatisch gleichbedeutend mit der Relation „O_A ist nicht dunkler als O_B“ sein, da $\mu(dunkler) \neq \nu(\mu(heller))$ sein kann. Aus gleichem Grund muß diese Relation nicht gleichbedeutend mit „O_B ist dunkler als O_A“ sein. Um das intuitive Verständnis zu erfüllen, müssen die Fuzzy-Mengen entsprechend definiert werden.

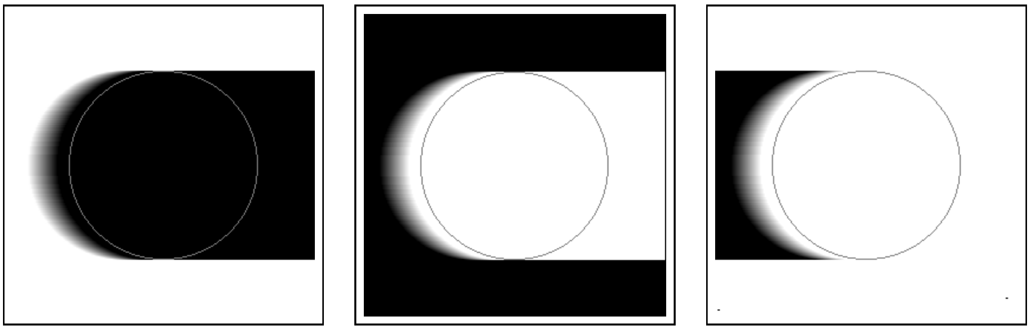
Eine grafische Darstellung der Interpretationen dieser Relationen ist in Abbildung B.7 zu finden, in der die Bildpunkte, welche die Relation zu einem Grad größer Null erfüllen, dargestellt werden. Zudem ist dort gezeigt, daß die Negation einer Relation „Objekt A liegt mindestens 10-35 Bildpunkte links neben Objekt B“ nicht dem bezüglich der hier gewählten Interpretation intuitiven Verständnis entspricht, da eine Negation auch Bildpunkte über und unter dem Objekt A ausschließt. Aus diesem Grund wird die Relation „ragt über“ eingeführt (vgl. Abbildung 5.11). Der Grund für die derartige Interpretation der Lagerelationen liegt darin, daß Objekte nicht ausgeschlossen werden sollen, wenn sie nicht in der Projektion der Objekte liegen. Falls das verhindert werden soll, kann dies durch die Verknüpfung mehrerer Relationen erreicht werden. Soll etwa angegeben werden, daß ein Objekt A *total links oben* vom Referenzobjekt B liegt, so ist eine Verknüpfung von „liegt total links“ und „liegt total oben“ mittels eines T-Quantors TQ_τ über den Bildbereich durchzuführen (vgl. Abbildung 5.12).

Der Unterschied bspw. zwischen „liegt links von“ und „ragt links über“ liegt im Bezugspunkt des Objektes, dessen Lage beschrieben werden soll. Das wird an Abbildung 5.13(a) verdeutlicht. Dort liegt das Objekt A links vom Referenzobjekt R, da $b_1 < c$ gilt. Objekt B liegt nicht links von R, da $b_2 \geq c$, außerdem ragt es nicht über R, weil $a_2 \geq c$. Das Objekt C liegt nicht links von R, da $b_3 \geq c$, aber es ragt links über R, da $a_3 < c$ gilt.

Definition 5.23 (Liegt mindestens m bis n Bildpunkte links von)
Ein Bildpunkt p mit den Koordinaten (x, y) erfüllt die Relation $leftof_B$ zum Referenzobjekt B wie folgt:

$$leftof_B(p) = \begin{cases} 1 & \text{falls } x < \min(p_B.x),\ p_B \in B \\ 1 & \text{falls } x > \max(p_B.x),\ p_B \in B \\ \mu_{atleast,leftof_B}\left(dist_{leftof_B}(p, B)\right) & \text{sonst} \end{cases} \tag{5.37}$$

Hierbei ergibt $dist_{leftof_B}$ den horizontalen Abstand vom Punkt p zu dem am weitesten links liegenden Punkt $p_{B,mostleft(x)}$ mit $p_{B,mostleft(x)}.x = x$ und

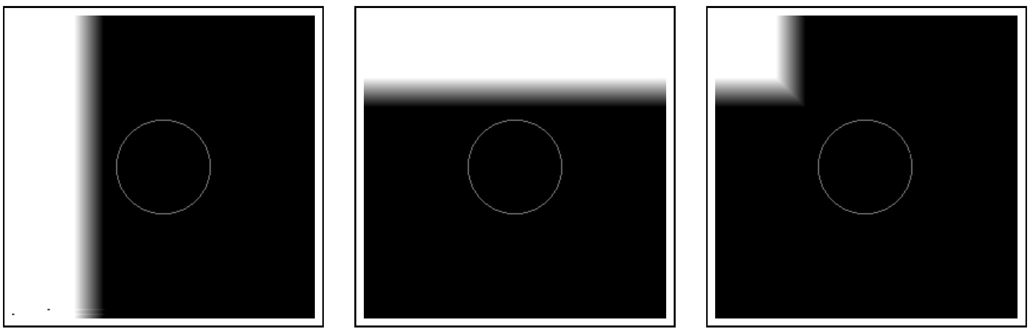


(a) Liegt mindestens 10-35 Bildpunkte links von

(b) Liegt nicht mindestens 10-35 Bildpunkte links von

(c) Ragt höchstens 10-35 Bildpunkte links über

Abbildung 5.11: Vergleich von Lagerelationen und deren Negation bezogen auf die Richtung links*, bei der die über und unter dem Referenzobjekt liegenden Bildpunkte auch ausgeschlossen werden. Zur besseren Anschauung wird das Referenzobjekt (Kreis) mit eingeblendet.*

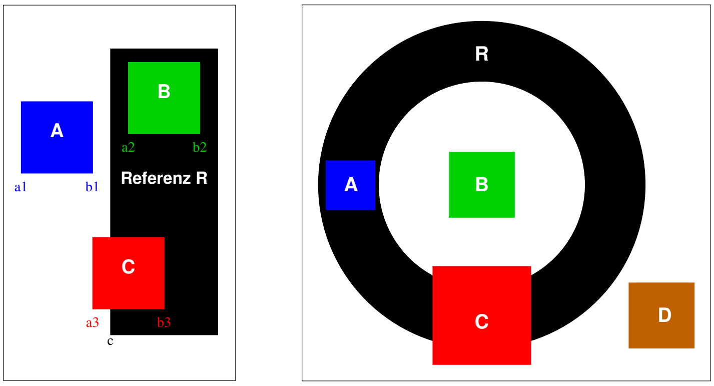


(a) Liegt total links von

(b) Liegt total oben von

(c) Liegt total links oben von

Abbildung 5.12: Darstellung der Relation „liegt total links von" und „liegt total oben von" und deren Verknüpfung zu „liegt total links oben von", jeweils mit einem unscharfen Abstand von 10-35 Bildpunkten. Für die Verknüpfung wird die T-Norm min *verwendet.*



(a) *liegt* vs. *ragt über*

(b) *Teil von* vs. *innerhalb*

Abbildung 5.13: Vergleich von Lagerelationen bezogen auf die Richtung links. *Die a_i, b_i geben die linke und rechte Koordinate des Objektes an (Abbildung 5.13(a)). Abbildung 5.13(b) zeigt das Verhältnis zwischen den Relationen „ist Teil von" und „liegt innerhalb".*

$p_{B,mostleft(x)} \in B$.

$$dist_{leftof_B}(p, B) = min(p_B.y) - y \text{ für } p_B.x = x,\ p_B \in B \tag{5.38}$$

$\mu_{atleast,leftof_B}$ beschreibt die durch das Intervall $[m, n]$ festgelegte unscharfe Menge eines zu definierenden Typs θ mit dem 5-Tupel[16] $(\theta, m, n, dist_{\max}, dist_{\max})$.

□

Alternativ können hier statt der Bildpunkte auch metrische Abstände verwendet werden, wobei dann die Größe eine Bildpunktes bekannt sein muß. In der Regel ist das bei medizinischen Bilddaten der Fall. Zu berücksichtigen ist allerdings, daß entsprechende Rundungen aufgrund der diskreten Bilddarstellung notwendig sind.

Für die Richtung *rechts* ist lediglich die Abstandsfunktion zu ändern, da hier der am weitesten rechts liegende Punkt des Referenzobjektes bezüglich jeder Zeile betrachtet werden muß:

$$dist_{rightof_B}(p, B) = max(p_B.y) - y \text{ für } p_B.x = x,\ p_B \in B \tag{5.39}$$

Wenn der Abstand nicht mit *mindestens*, sondern mit *höchstens* angegeben werden soll, so ist eine unscharfe Menge $\mu_{atmost,leftof_B}$ entsprechend mit dem 5-Tupel $(\theta, 0, 0, m, n)$ zu definieren. Die Richtungen *oben* und *unten* erfordern ein

[16] Vergleiche hierzu Festlegung 6.1.

entsprechendes Vertauschen der Koordinaten, ansonsten gelten die gleiche Definitionen.

Die Total-Relationen beziehen sich auf den in der Richtung jeweils äußeren Bildpunkt des Referenzobjektes und muß deshalb anders definiert werden:

Definition 5.24 (Liegt mindestens m bis n Bildpunkte total links von) Ein Bildpunkt p mit den Koordinaten (x, y) erfüllt die Relation $totalleftof_B$ zum Referenzobjekt B wie folgt:

$$totalleftof_B(p) = \mu_{atleast,totalleftof_B}\left(dist_{totalleftof_B}(p, B)\right) \tag{5.40}$$

Hierbei ergibt $dist_{totalleftof_B}$ den horizontalen Abstand vom Punkt p zu dem am weitesten links liegenden Punkt $p_{B,mostleft} \in B$.

$$dist_{totalleftof_B}(p, B) = min(p_B.y) - y,\ p_B \in B \tag{5.41}$$

$\mu_{atleast,totalleftof_B}$ beschreibt die durch das Intervall $[m, n]$ unscharfe Menge eines zu definierenden Typs θ mit dem 5-Tupel $(\theta, m, n, dist_{\max}, dist_{\max})$. □

Die bisherigen Lagerelationen zeigen jeweils die Zugehörigkeit eines einzelnen Bildpunktes zum Referenzobjekt. Bei der Betrachtung eines einzelnen Punktes unterscheiden sich aber die Relationen „liegt *Richtung* von" und „ragt *Richtung* über" nicht. Für die Betrachtung eines gesamten Objektes ist dessen Kontur zu berücksichtigen, da die Lagerelation für das gesamte Objekt gelten soll. Dementsprechend wird ein T-Quantor TQ_τ für die Aggregation über die Konturpunkte des Objektes verwendet, dessen Lagerelation zu dem Referenzobjekt bestimmt werden soll, die in der jeweiligen gewählten Richtung dem Referenzobjekt am nächsten („liegt neben") bzw. am weitesten („ragt über") gelegen sind.

Die Relation der Helligkeiten kann wie folgt über die Helligkeitsdifferenz berechnet werden. Dazu wird die Definition 5.13 verwendet.

Definition 5.25 (Heller als) Die Helligkeitsrelation $brighterthan_R$ eines Segmentes S zu einem Referenzobjekt R berechnet sich wie folgt:

$$brighterthan_R(S) = \max\left(0, brightness(S) - brightness(R)\right) \tag{5.42}$$

□

Definition 5.26 (Dunkler als) Die Helligkeitsrelation $darkerthan_R$ eines Segmentes S zu einem Referenzobjekt R berechnet sich wie folgt:

$$darkerthan_R(S) = \max\left(0, brightness(R) - brightness(S)\right) \tag{5.43}$$

□

Die Relation „ist Teil von“ kann ebenfalls unscharf formuliert werden, wobei die Schnittmenge im Verhältnis zur Fläche des betrachteten Objektes als Maß dient. Dem gegenüber bezieht sich die Relation „innerhalb“ auf die konvexe Hülle des Referenzobjektes R, so daß auch Objekte im Inneren einer Ringstruktur oder in Löchern betrachtet werden können. Abbildung 5.13(b) zeigt bspw. eine Ringstruktur als Referenz R. Dort liegt Objekt A innerhalb von R und ist Teil von R. Objekt B liegt zwar innerhalb von R, ist aber kein Teil von R. Das Objekt C ist partiell Teil von R und liegt partiell innerhalb von R, während D weder innerhalb von R liegt noch Teil von R ist. Diese Differenzierung ist notwendig, da bspw. der Corpus Callosum etwa innerhalb des Gehirns liegt, jedoch nicht Teil davon ist[17]. Eine ähnliche Unterscheidung macht Menhardt [Men90], der mit der Relation *liegt in* alle Bildpunkte definiert, die zur konvexen Hülle des Referenzobjektes gehören, während er mit der Relation *innerhalb* alle Bildpunkte bezeichnet, die in der Differenzmenge der konvexen Hülle des Referenzobjektes und dem Referenzobjekt selbst liegen. Entsprechend der obigen Beschreibungen der Relationen können diese jetzt definiert werden. Negationen erfolgen gemäß Definition 3.13 mittels einer Funktion ν.

Definition 5.27 (Ist Teil von) Der Grad der Teilmengenbeziehung $partof_R$ eines Segmentes S zu einem Referenzobjekt R berechnet sich wie folgt:

$$partof_R(S) = \frac{size_{fast}(S \cap R)}{size_{fast}(S)} \tag{5.44}$$

□

Definition 5.28 (Liegt innerhalb) Der Grad der Teilmengenbeziehung $inside_R$ eines Segmentes S zu einem Referenzobjekt R berechnet sich wie folgt:

$$inside_R(S) = \frac{size_{fast}(S \cap convexhull(R))}{size_{fast}(S)} \tag{5.45}$$

□

Wichtig für alle Eigenschaftsberechnungen ist, daß die Ähnlichkeit zweier Objekte sich auch in ähnlichen Funktionswerten ausdrückt, das heißt, die Funktion darf kein chaotisches Verhalten aufweisen und sollte (stückweise) stetig sein.

Einige der Eigenschaften sind invariant bezüglich ihrer Größe, Lage (Translationsinvarianz) oder Drehung (Rotationsinvarianz). Bei der Verwendung der Wissensinhalte muß dieser Umstand auf jeden Fall beachtet werden. Die Beschreibung gesuchter Objekte muß ausreichend groß sein, um sie von anderen Objekten, mit denen sie einige invariante Eigenschaften gemein haben, trotzdem noch unterscheiden zu können. Die Untersuchung ausschließlich dieser invarianten Merkmale erlaubt keine Differenzierung. Bspw. läßt sich eine Unterscheidung

[17] Dies ist nicht im medizinischen Sinne gemeint, vielmehr geht es hier um eine Segmentzugehörigkeit.

vom linken und rechten Auge in einem MRT-Bilddatensatz nur durch die Lage durchführen, während die übrigen Eigenschaften annähernd gleich sind.

In Abschnitt B.1.1 werden einige Beispiele für die Berechnung von oben genannten Segmenteigenschaften bzw. Relationen zwischen Segmenten vorgestellt.

Probleme und Besonderheiten

Sicherlich läßt sich mit den oben genannten Beschreibungsmöglichkeiten nicht jede Gegebenheit im Bild darstellen. Auch sind die Interpretationen der Begriffe von herausragender Bedeutung und sollten für die Modellierung der Wissensbasis bekannt sein, was in CIM^{2}BA/P durch die zweistufige Erzeugung der Wissensbasis unterstützt wird (vgl. Kapitel 6). Daneben gibt es Probleme, die gegebenenfalls gesondert behandelt bzw. entsprechende Beschreibungsmöglichkeiten dafür realisiert werden müssen.

Bspw. ist die Verwendung der konvexen Hülle für nicht geschlossene Figuren, etwa wie das Objekt aus Abbildung B.1(e), nicht intuitiv für die „innerhalb"-Relation, da ein Objekt auch dann als „innerhalb" des Referenzobjektes gilt, wenn es in der Öffnung liegt. Eine Alternative ist die Berechnung, ob es einen Pfad von dem Objekt zum Rand gibt, der nicht Teil des Referenzobjektes ist, also mindestens einen gemeinsamen Punkt mit dem Referenzobjekt hat. Existiert dieser nicht, liegt das Objekt innerhalb des Referenzobjektes. Allerdings ist die Berechnung eines derartigen Pfades aufwendig, da der Pfad nicht gradlinig sein muß.

Ebensowenig muß bei einem nicht geschlossenen Segment für die Berechnung $roundness_R$ (vgl. Definition 5.16) jeder Strahl einen Konturpunkt treffen. Das Ergebnis ist stark abhängig von der Wahl des Parameters für die Anzahl der Strahlen und des Startwinkels. Da derartige Strukturen jedoch in der Minderzahl und die genannten Berechnungsmöglichkeiten vergleichsweise einfach sind, ist diese Interpretation meistens anwendbar.

Für die Berechnung der Orientierung wird die Hauptachse verwendet. Das angegebene Verfahren berechnet immer eine Hauptachse, auch wenn diese nicht deutlich ausgeprägt ist (runde und quadratische Strukturen besitzen keine ausgeprägte Hauptachse). In diesem Fall wird die Orientierung immer mit 90° Grad bezüglich des gewählten Koordinatensystems und dessen Orientierung berechnet.

Eigenschaften der Relationen

Da die Transitivität für die Verwendung von Relationen wichtig ist, sind die angegebenen Relationen darauf hin zu untersuchen:

Satz 5.2 *Die Relation „liegt* [*Richtung*] *von" ist nicht transitiv.* □

Beweis zu Satz 5.2 Der Beweis ergibt sich unmittelbar aus der Definition 5.23. Ein Gegenbeispiel wird in Abbildung 5.14 gezeigt. Bei der Interpretation der Relation „liegt [*Richtung*] von" werden die außerhalb der Projektion des Referenzobjektes liegenden Bildpunkte nicht berücksichtigt, wodurch die Transitivität

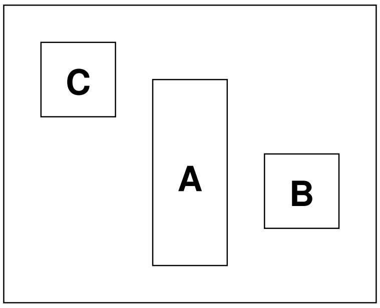


Abbildung 5.14: Beispiel für eine Konstellation, die zeigt, das die einfache Lagerelation nicht transitiv ist, da nur die Projektion des jeweiligen Referenzobjektes berücksichtigt wird. Aufgrund der Definition 5.23 gilt: „A liegt links von B“, „B liegt links von C“, aber es gilt nicht „A liegt links von C“.

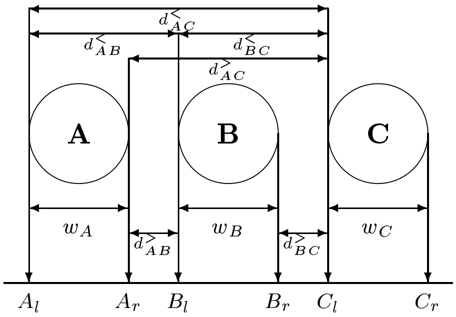


Abbildung 5.15: Projektion der Extremkoordinaten dreier Objekte, anhand derer die Transitivität der „liegt total links von“-Relation gezeigt wird.

verloren geht Abbildung 5.14 zeigt: A liegt links von B, Objekt B liegt links von C, aber es gilt nicht A liegt links von C. ■

Für die einfache (nicht totale) Lagebeziehung wurde schon gezeigt, daß sie nicht transitiv ist. Diese Einschränkung gilt nicht für die *totale* Lagerelation. Um dies zu zeigen, ist es ausreichend, die zur Betrachtungsrichtung orthogonale Projektion zu betrachten, wie in Abbildung 5.15 dargestellt ist. Zu zeigen ist, daß wenn A total links neben B liegt und B total links neben C, dann liegt auch A total links neben C. Soll statt der „liegt total von“-Relation ($d^<$) die „ragt total über“-Relation verwendet werden, wird $d^>$ als Abstand benutzt.

Satz 5.3 *Die Relation „Objekt A liegt mindestens m_{AB} bis n_{AB} Bildpunkte total links von Objekt B“ ist transitiv und die untere Schranke des Mindestabstandes ist additiv.* □

Beweis zu Satz 5.3 Hierfür muß gezeigt werden, daß falls $totalleftof_B(A) > 0 \wedge totalleftof_C(B) > 0$, dann gilt $totalleftof_C(A) > 0$. Mit den in Abbildung

5.15 eingeführten Variablen ist es ausreichend zu zeigen, daß $d^{>}_{AC} \geq d^{>}_{AB} + d^{>}_{BC}$. Dieses folgt aber unmittelbar aus $d^{>}_{AC} = d^{>}_{AB} + w_B + d^{>}_{BC}$ und $w_B \geq 0$. Der neue Mindestabstand beträgt $m_{AB} + m_{BC} + w_B$. ■

Leider kann hierbei keine unmittelbare Aussage über die obere Grenze des Mindestabstandes des Intervalls $[m_{AC}, n_{AC}]$ getroffen werden. Ein weiteres Problem ist, daß die Breite w_B nicht bekannt ist, wenn Aussagen über die Bildpunkte getroffen werden, welche die Relation erfüllen, um einen Suchbereich zu beschränken. Denn wenn B bekannt ist, dann wird die Transitivität nicht mehr benötigt und die Lagebeziehung zu B kann direkt verwendet werden. Die Transitivität wird dann benötigt, wenn die Aussagen „A liegt total links von B" und „B liegt total links von C" gelten und B *nicht* bekannt ist. Dann kann aufgrund der Transitivität zumindest davon ausgegangen werden, daß auch „A liegt total links von C" gilt. Nur bei Kenntnis von w_B kann eine genauere Eingrenzungen getroffen werden (sonst $w_B = 0$). Der gesamte unscharfe Übergang wird aus den beiden einzelnen unscharfen Übergängen gebildet, das heißt, die entsprechenden Relationen lauten:

1. Objekt A liegt mindestens m_{AB} bis n_{AB} Bildpunkte total links von Objekt B

2. Objekt B liegt mindestens m_{BC} bis n_{BC} Bildpunkte total links von Objekt C

3. Objekt A liegt mindestens m_{AC} bis n_{AC} Bildpunkte links total von Objekt C mit $m_{AC} = m_{AB} + m_{BC}$ und $n_{AC} = n_{AB} + n_{BC}$

Sollen die Abstandsmaße nicht *mindestens m bis n* Bildpunkte, sondern *höchstens m bis n* Bildpunkte betragen, können keine Aussagen über die Abstände getroffen werden, da wiederum hierfür w_B bekannt sein muß. Lediglich bei Kenntnis einer Maximalbreite $w_{B_{\max}}$ für Objekt B kann die obere Grenze festgelegt werden. Das bedeutet, der Höchstabstand beträgt $n_{AB}+n_{BC}+w_{B_{\max}}$. Die Relationen lauten:

1. Objekt A liegt höchstens m_{AB} bis n_{AB} Bildpunkte total links von Objekt B

2. Objekt B liegt höchstens m_{BC} bis n_{BC} Bildpunkte total links von Objekt C

3. Objekt A liegt höchstens m_{AC} bis n_{AC} Bildpunkte total links von Objekt C mit $m_{AC} = m_{AB} + m_{BC} + w_{B_{\max}}$ und $n_{AC} = n_{AB} + n_{BC} + w_{B_{\max}}$

Durch die Kombination eines Mindest- und eines Höchstabstandes wird ein Bereich eingeschränkt, der zwischen den beiden äußeren Objekten liegt, allerdings ist dieses auch hier nur möglich, wenn die Maximalbreite $w_{B_{\max}}$ von B

bekannt ist, ansonsten kann nur die Mindestentfernung zu C genutzt werden. Da nur bei Unkenntnis von B wiederum die Transitivität genutzt werden muß, wird erneut der Mindestabstand und der maximale Höchstabstand verwendet, so daß die Relationen wie folgt aussehen:

1. Objekt A liegt höchstens m_{AB} bis n_{AB} Bildpunkte total links von Objekt B
2. Objekt B liegt mindestens m_{BC} bis n_{BC} Bildpunkte total links von Objekt C
3. Objekt A liegt mindestens $m_{1,AC}$ bis $n_{1,AC}$ und höchstens $m_{2,AC}$ bis $n_{2,AC}$ Bildpunkte total links von Objekt C mit $m_{1,AC} = m_{BC}$ und $n_{1,AC} = n_{BC}$ und $m_{2,AC} = m_{AB} + m_{BC} + w_{B_{\max}}$ und $n_{2,AC} = n_{AB} + n_{BC} + w_{B_{\max}}$

Analoge Bemerkungen gelten für die anderen Richtungen. Auch die „heller“- und „dunkler“-Relationen sind transitiv. Die Bemerkungen für die Unkenntnis des „mittleren“ Bezugsobjektes gelten hier ebenfalls, da bei dessen Bekanntsein die Transitivität nicht mehr benötigt wird.

5.2.3.3 Unscharfe Subsegment-Beschreibung

Teilweise sind Objekte schwer beschreibbar, gerade wenn es sich um größere oder komplexere Objekte handelt. Aus diesem Grund wird hier das Konzept der *Subsegmente* eingeführt, für das geeignete Beschreibungen gewählt werden können.

Definition 5.29 (Subsegment) Ein Subsegment $\mathcal{S}$ eines Segments S ist eine ikonische Fuzzy-Menge, deren Zugehörigkeit eine Lagerelation zu S erfüllt. □

Durch Verwendung von Subsegment-Beschreibungen kann der Hirnstamm etwa als längliches Objekt beschrieben werden, welches *oben eher rund* ist[18]. Speziell für längliche oder große Objekte bietet dies einen Vorteil, da eine Beschreibung präziser gefaßt werden kann, falls eine entsprechende weitere Zerlegung der gefundenen Struktur nicht durchführbar oder nicht sinnvoll ist. Die unscharfen Beschreibungen wie *oben*, *unten*, *links* oder *rechts* werden auf den Wert Eins normierte Extremkoordinaten, also der äußeren Koordinaten der konvexen Hülle des Segmentes, definiert und können allen Typen der erlaubten Fuzzy-Mengen entsprechen. Diese können über ein 5-Tupel $(\theta, a_1, a_2, a_3, a_4)$ definiert werden, etwa für die Beschreibung *oben* das Tupel $(1, 0.4, 0.6, 1.0, 1.0)$. Mit Hilfe einer zylindrischen Erweiterung über die orthogonale Achse kann dann die ikonische Fuzzy-Menge berechnet werden. Liegt eine Drehung des Segmentes vor, die störend auf die Beschreibung wirkt, so ist eine geeignete Anpassung an die Hauptachse durchzuführen. Die Verwendung von Subsegmenten ist in Abbildung B.8 dargestellt.

[18]Gemeint ist der mittlere und obere Teil des Hirnstammes, der sogenannte Pons.

5.2.3.4 Unscharfe Modelle

Häufig stellt die alleinige Verwendung verbaler Beschreibungen ein Problem für den Domänenexperten dar. In Wissensakquisitionssitzungen werden oft auch Skizzen zur Beschreibung eines Objektes verwendet. Da Skizzen aber nur vereinfachte Modelle der Realität darstellen, müssen sie ebenso unscharf betrachtet werden. Dazu können je nach Variabilität der Objekte ikonische Fuzzy-Modelle, kurz FOM genannt, genutzt werden, diese wurden erstmals in [Hil98, HFR98] vorgestellt.

Das Sprichwort „Ein Bild sagt mehr als tausend Worte" wird hier als Grundlage der Umsetzung unscharfer Modelle verwendet. Idee ist die Speicherung eines Objektmodells in der Wissensbasis, welche über unscharfe Ränder dargestellt wird. Die Art bzw. Größe der Unschärfe wird abhängig von der Variabilität der gesuchten Objekte gewählt und über einen objektabhängigen Parameter δ_O gesteuert. Definiert werden die unscharfen Objektmodelle über ikonische Fuzzy-Mengen.

Im Gegensatz zu Arbeiten von bspw. Han et al. [HKP94] oder Philip et al. [PDMP$^+$94], die unscharfe Modelle mittels einer Fuzzy-Hough-Transformation für eine Segmentierung verwenden, werden diese Modelle hier aber zur *Beschreibung* eingesetzt. Dementsprechend wird ein Vergleich zwischen einem gefundenen Segment und dem FOM durchgeführt. Da die Variabilitäten aber immer noch sehr groß sein können, sind Normierungen zwischen dem Segment und dem FOM notwendig [HFR01]:

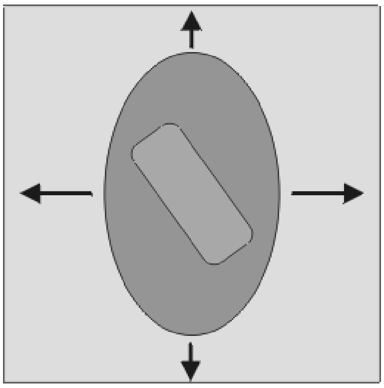
1. Translation: Durch die Betrachtung der koordinatenunabhängigen Segmente ist eine Translationsinvarianz gegeben.

2. Größe: Die Größe ist ein Hauptfaktor der varianten Strukturen in medizinischen Bilddaten. Deshalb wird eine Normierung des FOM durchgeführt. (Im Beispiel (vgl. Abbildung 5.16(b)) erfolgt die Normierung anhand der Kopfgröße.)

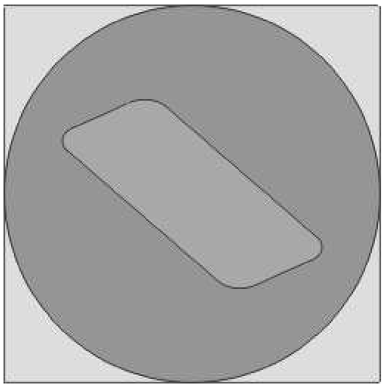
3. Rotation: Diese stellt ein weiteres Problem dar. Daher wird eine Normierung des Segmentes und des FOM an der Hauptachse der jeweiligen Struktur durchgeführt, sofern diese existiert (vgl. Abbildung 5.16(d)). Berechnet werden dabei sowohl die Zugehörigkeit des zur Hauptachse orientierten als auch des um 180 Grad gedrehten Segmentes. Als Zugehörigkeitswert wird das Maximum beider Werte gewählt.

4. Helligkeit: Da nur die Konturen der Segmente betrachtet werden, ist keine Normierung der Helligkeit notwendig.

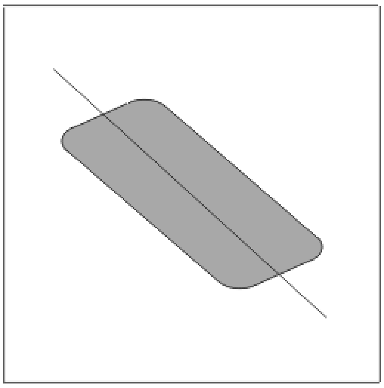
Auch Abbildung 5.17 verdeutlicht, daß erst anhand einer Größennormierung der Einsatz unscharfer Modelle möglich wird. Ohne diese Normierung sind die Variabilitäten in medizinischen Bilddaten so stark, daß ein sinnvoller Einsatz von FOM kaum möglich ist.



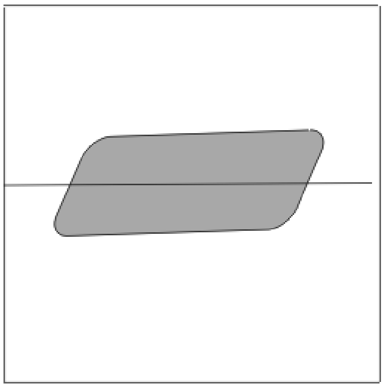
(a) Startbild



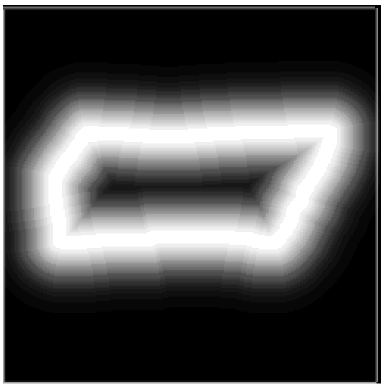
(b) Normalisiertes Bild



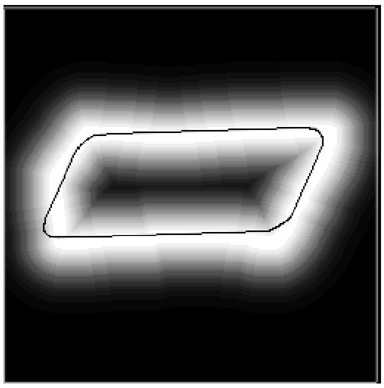
(c) Hauptachse des segmentierten Segments



(d) Ausrichtung an der Hauptachse

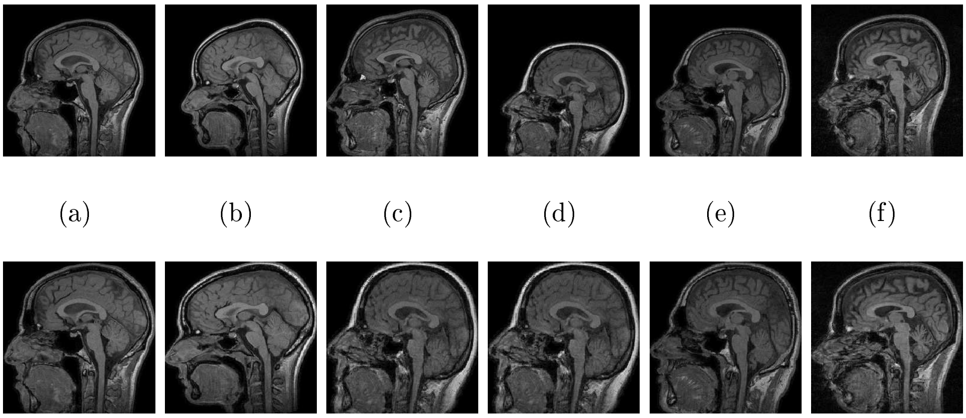


(e) Gespeichertes FOM mit $\delta_O = 16$



(f) Überlagerung von Segment (schwarze Kontur) und FOM

Abbildung 5.16: Schematische Darstellung der Normierungsschritte und der Bestimmung der Zugehörigkeit eines Segmentes zu einem unscharfen Objektmodells anhand eines Beispiels.



(a) (b) (c) (d) (e) (f)

(g) (h) (i) (j) (k) (l)

Abbildung 5.17: Darstellung der Normierung der Kopfgröße. Gezeigt werden die ursprüngliche Größe (obere Reihe, (a)-(f) und die Größe nach der Normierung (untere Reihe, (g)-(l)). Deutlich wird, daß sich Vergleiche anhand genormter Modelle deutlich einfacher durchführen lassen.

Nach diesen Normierungen werden das zu bewertende Segment und das zugehörige FOM im jeweiligen Schwerpunkt überlagert. Anschließend werden die Zugehörigkeiten an allen Konturpunkten bestimmt, wie in Abbildung 5.16(f) dargestellt ist. Hierbei muß als Kontroll- oder Korrekturfaktor die Länge der Konturlinie des Modells berücksichtigt werden. Ansonsten kann ein einzelner Punkt schon die maximale Zugehörigkeit zu dem FOM erhalten und somit als das Objekt, welches das FOM darstellt, erkannt werden. Trotzdem sind Fehlpunkte in der Kontur des Segmentes möglich, die etwa durch eine kantenorientierte Segmentierung durch nicht geschlossene Kantenzüge entstehen (vgl. Definition 5.30).

Definition 5.30 (FOM-Zugehörigkeitsgrad) Die Zugehörigkeit eines Segmentes S zu einem FOM $\mathcal{F}$ berechnet sich aus der mittleren Zugehörigkeit jedes Konturpunktes $p \in contour(S)$ zu $\mathcal{F}$, nachdem die oben genannten Normierungen durchgeführt wurden.

$$\mu_{\mathcal{F}}(S) = \min\left(1, \frac{perimeter(S)}{perimeter\left(core(\mathcal{F})\right)}\right) \cdot \frac{\sum_{p \in contour(S)} \mu_{\mathcal{F}}(p)}{perimeter(S)} \qquad (5.46)$$

□

Die FOM für eine Objektklasse O werden über ihre Konturlinie und einen unscharfen Rand der Breite δ_O beschrieben. Hierfür sind, wie in Definition 6.1 dargestellt, verschiedene Arten der Randdarstellung möglich. Diese sind jedoch zweidimensional kreisförmig um jeden Konturpunkt zu legen, so daß sie nicht

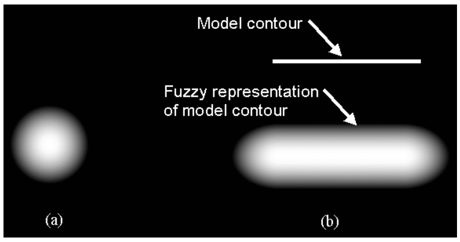
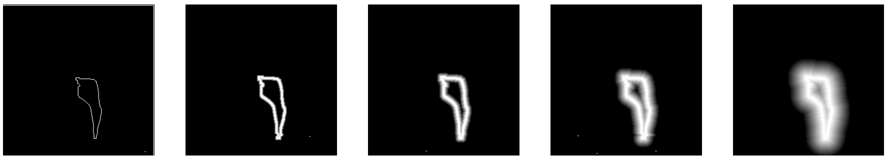


Abbildung 5.18: Exemplarische Darstellung einer Objektkontur eines FOM anhand eines einzelnen Bildpunktes und eines Linienstückes mit einer kegelförmigen Fuzzy-Menge.

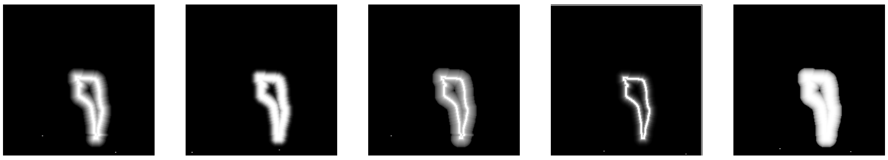


(a) Modell O (b) $\delta_O = 4$ (c) $\delta_O = 8$ (d) $\delta_O = 16$ (e) $\delta_O = 32$

Abbildung 5.19: Darstellung der Bedeutung unterschiedlich gewählter Unschärfebereiche δ_O auf die Repräsentation von unscharfen Objekt-Modellen. Als Beispiel wird das Modell eines Hirnstammes gezeigt. Die Fuzzy-Menge ist vom Typ 1 (linear).

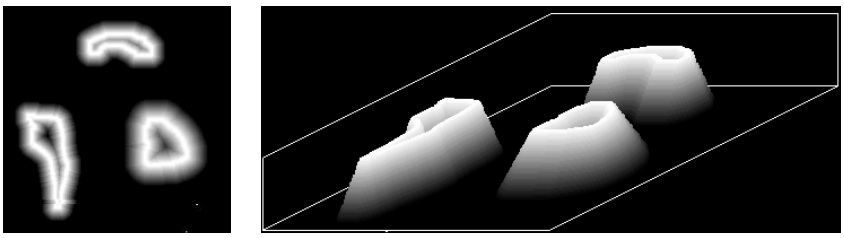
asymmetrisch ausgelegt sein können und neben dem Typ der Fuzzy-Menge die Angabe der Breite des Kerns und des Trägers ausreichend sind. Die Überlappungen benachbarter Bildpunkte werden mittels einer S-Norm berechnet. Abbildung 5.18 zeigt ein Beispiel für die Darstellung eines einzelnen Bildpunktes und eines Linienstückes.

In der Abbildung 5.19 wird dargestellt, wie sich die Breite δ_O auf das unscharfe Objektmodell auswirkt. Abbildung 5.20 zeigt die Bedeutung der verschiedenen Typen unscharfer Mengen anhand des Beispiels eines Hirnstammes mit den zugehörigen (eindimensional dargestellten) Mengen. Schließlich zeigt Abbildung 5.21 ein Beispiel für eine perspektivische Sicht auf die unscharfen Modelle des Kleinhirns, des Corpus Callosum und des Hirnstammes, jeweils mit einer kegelförmigen Fuzzy-Menge. Die Berechnung der Zugehörigkeiten mit Beispielen segmentierter Objekte wird in Abschnitt B.1.2 gezeigt.



(a) Typ 1 (b) Typ 2 (c) Typ 3 (d) Typ 4 (e) Typ 5

Abbildung 5.20: Darstellung der Bedeutung unterschiedlicher Typen von Fuzzy-Mengen auf die Repräsentation von unscharfen Objekt-Modellen. Als Beispiel wird das Modell eines Hirnstammes aus Abbildung 5.19(a) gezeigt, für die Breite gilt $\delta_O = 16$.



(a) 2D-Ansicht (b) 3D-Ansicht

Abbildung 5.21: Exemplarische dreidimensionale Darstellung der Objektkonturen unscharfer Modelle des Kleinhirns, Corpus Callosum und des Hirnstammes mit jeweils einer kegelförmigen unscharfen Menge.

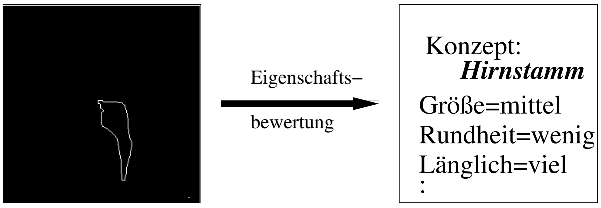


Abbildung 5.22: Generierung eines Konzeptes aus dem Kern eines Fuzzy-Objekt-Modells.

5.2.3.5 Beschreibungsgenerierung aus Modellen

Es besteht in CIM^{2}BA auch die Möglichkeit, aus den grafischen Modellen eine textuelle Beschreibung mittels eines Konzeptes (vgl. Abschnitt 5.2.1.1) zu realisieren (Abbildung 5.22). Dazu wird von einen FOM der Kern betrachtet. Zu allen unscharfen Variablen $\mathcal{F}_\diamond$, die auch für ein Segment (vgl. Abschnitt 6.1.2.6) berechnet werden können, werden jeweils alle Terme berechnet. Der Variablen $\mathcal{F}_\diamond$ wird dann der Term mit dem höchsten Zugehörigkeitsgrad zugewiesen. Weiterhin ist die Definition eines Schwellenwertes T möglich, der als Zugehörigkeit überschritten werden muß. Ist dies nicht der Fall, wird der Variablen der Wert *Don't-Care* zugewiesen. Als Einschränkung gilt, daß nur geometrische Eigenschaften betrachtet werden können, während photometrische Eigenschaften im FOM nicht angegeben sind (diese werden im zugehörigen Konzept selbst definiert). Die so generierte Beschreibung kann nach Überprüfung durch den Wissensingenieur dann in die Wissensbasis mit aufgenommen werden. Dementsprechend handelt es sich hierbei nicht um die Generierung temporären Wissens, sondern mehr um ein Werkzeug zur Wissensakquisition (vgl. Abschnitt 3.2.3).

5.2.4 Klassifikation mit Hilfe unscharfer Methoden

Klassifikation bezeichnet die Benennung der gefundenen Segmente mit den Namen der in der Wissensbasis bekannten Objekte oder deren Ablehnung (Klassifikation als *unbekannt*) (Abbildung 5.23).

Bevor ein Objekt aber als *unbekannt* zurückgewiesen wird, ist ein – wie in Abbildung 5.2 gezeigt – mehrstufiger Analyseprozeß mit Rückkopplungen möglich. Es gibt zwei mögliche Wege zur Klassifikation, die beschritten werden können.

- Es kann entweder eine vollständige Segmentierung des Bildes erfolgen und anschließend eine Klassifikation der Segmente vorgenommen werden.
- Auch ist es möglich, gezielt ein in einem Konzept beschriebenes Objekt zu suchen.

Während im ersten Weg eine Segmentierung nach generellen, zur Segmentierung notwendigen Eigenschaften erfolgt, wie sie in Abschnitt 5.2.2 beschrieben

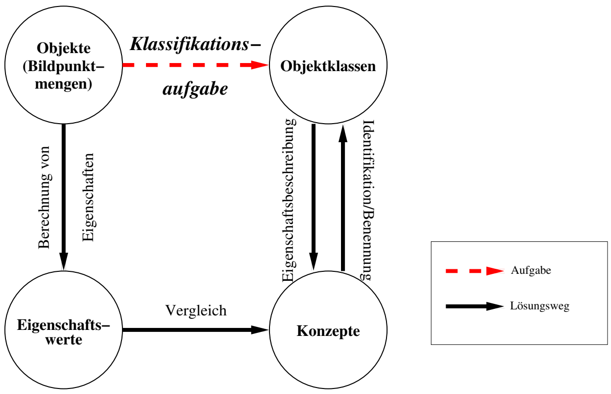


Abbildung 5.23: Prinzipielles Vorgehen bei der Klassifikation von Segmenten.

werden, sind für den zweiten Weg gezielt nur die Eigenschaften zu betrachten, die im gesuchten Objekt O mit dem Konzept C_O angegeben sind. Allerdings können hier nur direkt aus dem Bild ablesbare Eigenschaften wie Lage oder Grauwert verwendet werden, da noch keine Segmente existieren. Auf dem über die Eigenschaften eingeschränkten Suchraum wird dann eine Segmentierung durchgeführt. Im Gegensatz zum ersten Weg werden die gefundenen Segmente nur mit dem Konzept C_O verglichen, während im ersten Weg selbst für alle Segmente auch alle Konzepte C_i der Wissensbasis ω verglichen werden. Die gezielte Suche ist sinnvoll für einfach[19] zu findende Objekte. Diese können in einem weiteren Schritt dann in den Relationen zu den anderen Objekten verwendet werden. Für diese Sortierung der Suche kommt deshalb ein zum Konzept gehörenden Prioritätsparameter zum Tragen, mit dem eine Ordnung auf den Eigenschaften definiert werden kann (vgl. Abschnitt 6.1.2.4). Somit werden in der Umsetzung immer beide Wege der Klassifikation genutzt.

5.2.4.1 Behandlung der Unschärfe des Segmentes

Da das Segment als ikonische Fuzzy-Menge vorliegt, das heißt, jedem Bildpunkt des Segmentes ist ein Zugehörigkeitswert zugeordnet, erfolgt obige Bewertung mehrstufig. Dazu wird eine einfache Diskretisierungsfunktion $\zeta_n(i) = \frac{i}{n},\ i \in \{1, \ldots, n\}$ verwendet, die für einen wählbaren Parameter $n > 0,\ n \in \mathbb{N}$ den Wertebereich für α diskretisiert. Aus dem unscharfen Segment S werden mit Hilfe von ζ genau $n + 1$ α-Segmente $S_i,\ i \in \{0, \ldots, n\}$ berechnet[20].

[19] Bspw. sind das Objekte, die eindeutig abgrenzbare Merkmale besitzen und für die zur Klassifikation keine Relationen notwendig sind.
[20] Es wird davon ausgegangen, daß $S_i \neq S_j$ für $i \neq j$ gilt.

Eigenschaft	S_0	S_1	$\cdots$	S_n
$e_{1,C} = t_1$	$\mu_{(e_1=t_1)_C}(e_{1,C}(S_0))$	$\mu_{(e_1=t_1)_C}(e_{1,C}(S_1))$		$\mu_{(e_1=t_1)_C}(e_{1,C}(S_n))$
$e_{2,C} = t_2$	$\mu_{(e_2=t_2)_C}(e_{2,C}(S_0))$	$\mu_{(e_2=t_2)_C}(e_{2,C}(S_1))$		$\mu_{(e_2=t_2)_C}(e_{2,C}(S_n))$
$\vdots$				
$e_{m,C} = t_m$	$\mu_{(e_m=t_m)_C}(e_{m,C}(S_0))$	$\mu_{(e_m=t_m)_C}(e_{m,C}(S_1))$		$\mu_{(e_m=t_m)_C}(e_{m,C}(S_n))$

Tabelle 5.1: Zugehörigkeiten der Segmente S_i zu den Eigenschaften e_j.

$$S_i = \begin{cases} S^{>0} & \text{falls } i = 0 \\ S^{\geq \zeta_n(i)} & \text{falls } i \in \{1, \ldots, n\} \end{cases} \tag{5.47}$$

Die S_i repräsentieren somit scharfe Segmente, deren Zugehörigkeitsgrad einen Schwellenwert $\alpha_i = \zeta_n(i)$ überschreitet. Interpretiert werden können die Segmente in der Art, daß die α-Werte ausdrücken, wie sehr die Bildpunktmenge S_i der gesuchten Bildpunktmenge entspricht, die im Bereichswachstumsverfahren mittels definierter Kriterien berechnet wird. Für jeden der $n+1$ Segmente S_i kann die Zugehörigkeit zu einer Eigenschaft berechnet werden. Als Ergebnis wird ein Vektor mit $n+1$ Eigenschaftswerten (Singletons) berechnet. Dieser Vektor kann im Rahmen obiger Bemerkungen für alle im Konzept C definierten Eigenschaften und Relationen berechnet werden (vgl. Tabelle 5.1). Dabei berechnet $\mu_{(e_i=t_j)_C}(e_{i,C}(S_k))$ den Zugehörigkeitswert des Segmentes S_k zu der Eigenschaft $e_{i,C}$ des Konzeptes C zur Fuzzy-Menge des Terms t_j.

5.2.4.2 Defuzzifizierung

Aus der Menge der möglichen $n+1$ Segmente wird ein Ähnlichkeitswert zum Konzept C berechnet. Da jeweils alle Eigenschaften erfüllt sein sollen, werden die Einzelergebnisse $\mu_{(e_j=t_j)_C}(e_{j,C}(S_i))$ mittels eines T-Quantors TQ_τ über alle Eigenschaften $j \in \{1, \ldots, m\}$ aggregiert, so daß eine Menge von Ähnlichkeitswerten (je Segment S_i ein Wert) berechnet wird. Aus dieser Menge, welches einer $n+1$-elementigen diskreten Fuzzy-Menge $\mathcal{R} = \{(S_i, \mu(S_i)) | S_i \subseteq I,\ i \in \{0, \ldots, n\}\}$ über die α-Segmente entspricht, muß ein Lösungskandidat ausgewählt werden. Ein Beispiel ist in Abbildung 5.24 dargestellt.

Möglichkeiten zur Defuzzifizierung werden bspw. in [BG93, Bie97] vorgestellt. Unterschieden werden prinzipiell zwei Hauptmethoden, die *Schwerpunktmethode* und die *Maximummethode*. Die Schwerpunktmethode verwendet alle Elemente der Fuzzy-Menge, um daraus einen scharfen Wert zu berechnen. Dazu wird die mit dem Zugehörigkeitswert gewichtete Mittelung berechnet. Die Überlegung ist hierbei, daß *alle* Elemente mit einem Zugehörigkeitswert > 0 Einfluß auf das Ergebnis haben, wobei ggf. ein Schwellenwert T verwendet werden kann, den die Zugehörigkeitswerte überschreiten müssen [Ped89]. Diese Menge sei $\mathcal{R}^{\geq T} \subseteq \mathcal{R}$ mit $\mathcal{R}^{\geq T} = \{(S_i, \mu(S_i)) | S_i \subseteq I,\ i \in \{0, \ldots, n\},\ \mu(S_i) \geq T\}$ der Kardinalität m.

Dann berechnet die Schwerpunktmethode den Index l des scharfen Segmentes wie folgt (mittels einer Funktion ϱ ganzzahlig gerundet). $\gamma(i)$ gibt dabei den Index des i-ten Elementes aus $\mathcal{R}$ an, für den $\mu(S_{\gamma(i)}) \geq t$ gilt. Für $T = 0$ gilt $\mathcal{R} = \mathcal{R}^{\geq T}$, ϱ rundet den Wert zum nächsten (ganzzahligen) Index, so daß S_l das gesuchte Element ist.

$$l = \varrho\left(\frac{1}{m}\sum_{i=1}^{m} i \cdot \mu\left(S_{\gamma(i)}\right)\right) \tag{5.48}$$

Die Maximummethode hingegen wählt im einfachsten Fall das Element aus der Fuzzy-Menge, welches den höchsten Zugehörigkeitswert hat. In der Regel ist dieses Element nicht eindeutig, so daß hier eine Menge von Maxima existiert, aus denen ein Element ausgewählt werden muß. Dabei gibt es die Möglichkeit, dieses zufällig zu wählen. Auch besteht die Möglichkeit das kleinste (LOM, engl. *left of maxima*) oder größte (ROM, engl. *right of maxima*) zu wählen. Weiterhin gibt es die Möglichkeit über alle Elemente zu mitteln oder aber den Mittelpunkt der beiden äußeren (linken und rechten Elemente) zu wählen (MOM, engl. *mean of maxima*). Dann kann es aber sein, daß das gemittelte Ergebnis[21] nicht zur Ergebnismenge gehört, so daß der nächste Vertreter gewählt werden muß. $\mathcal{R}_{\max}$ sei hier die Menge der Maxima $\mathcal{R}_{\max} = \{(S_i, \mu(S_i)) | S_i \subseteq I,\ i \in \{0, \ldots, n\},\ \mu(S_i) = \max\}$ der Kardinalität n. Der entsprechende Index berechnet sich in $\mathcal{R}_{\max}$ wie folgt: Eine zu γ ähnliche Abbildung β bildet den Index auf das richtige Element in $\mathcal{R}$ ab, um das korrekte Element S_i (mit $i = \beta(l)$) zu bestimmen.

$$\begin{aligned}
l_{LOM} &= 1 && (5.49)\\
l_{MOM} &= \varrho\left(\frac{n+1}{2}\right) && (5.50)\\
l_{ROM} &= n && (5.51)\\
l_{MEAN} &= \varrho\left(\frac{1}{n} \cdot \sum_{1}^{n} i\right) = \varrho\left(\frac{1}{n} \cdot \frac{n \cdot (n+1)}{2}\right) = L_{MOM} && (5.52)
\end{aligned}$$

Abhängig von der Anwendung muß die richtige Defuzzifizierungsmethode gewählt werden, wobei eine semantisch korrekte Auswahl zu beachten ist. An dieser Stelle kann davon ausgegangen werden, daß die am besten zur Beschreibung passende Punktmenge S_i mit einem Ähnlichkeitswert `Si.compare(C)` (vgl. Abschnitt 6.1.2.7) vorliegt. Soll eine Identifizierung eines Segmentes S durchgeführt werden, muß die Berechnung des Ähnlichkeitswertes für alle in der Wissensbasis ω hinterlegten Konzepte C_j durchgeführt werden. Entsprechend wird dann für das Segment S für jedes Konzept C_j ein Wert berechnet. Es muß eine Auswahl

[21] Bei diskreten Mengen wird statt des Mittelwertes der Median gewählt.

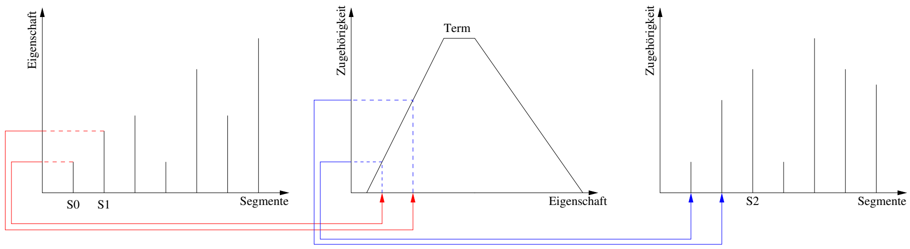


Abbildung 5.24: Bestimmung der Zugehörigkeitswerte der α-Segmente zu einer Eigenschaft. Zunächst wird die Eigenschaft für jedes S_i berechnet (links) und die Zugehörigkeit zur unscharfen Menge der Solleigenschaft bestimmt (Mitte). Die Zugehörigkeit wird für jedes S_i entsprechend aufgetragen (rechts).

(Zuordnung) des Konzeptes zum Segment erfolgen, wobei hier die gleichen Bemerkungen bzw. Methoden Anwendung finden können wie bei der Auswahl des Segmentes S_i. Unterstützt werden von CIM^{2}BA beide Methoden. Entsprechende Auswahlmöglichkeiten der einzelnen Komponenten und der Iterationsschritte werden in Abschnitt 5.5 erläutert, nachdem alle Bauteile von CIM^{2}BA vorgestellt worden sind.

5.3 Verwendung neuronaler Netze in CIM^{2}BA

Neuronale Netze sind sehr spezifisch für die jeweilige Aufgabe zu entwerfen. Ihre Topologie und Lernverfahren stellen praktisch Parameter für einen Operator dar, die geeignet eingestellt werden müssen. Deshalb sind die in diesen Beispielen erwähnten Parametergrößen entsprechend zu verstehen. CIM^{2}BA stellt einen derartigen Operator zur Verfügung, der über eine externe Beschreibung die Topologie des Netzes erfährt, wie sie später im Detail in Abschnitt 6.1.3.1 beschrieben wird.

5.3.1 Neuronale Netze zur Bestimmung der ROI

Neuronale Netze haben den besonderen Vorteil, dort eingesetzt werden zu können, wo eine funktionale Beziehung zwischen Ein- und Ausgabedaten existiert, diese jedoch nicht bekannt ist. Wie in [HAMM98, Rit98, RHM99] erstmalig gezeigt wurde, kann genau diese Fähigkeit genutzt werden, um in medizinischen Bilddaten eine Bestimmung der *Region of interest* (ROI) durchzuführen. Die Grundidee liegt darin, anhand von einfach zu bestimmenden Konturen auf die Kontur der ROI zu schließen. Dieses wird am Beispiel für die Segmentierung des Gehirns aus MRT-Datensätzen gezeigt. Dazu wird die Kontur des Kopfes genutzt, welche durch einen einfachen Kantenoperator ermittelt und dann mittels eines Richtungsketten-Codes (vgl. Abschnitt 2.3.3.1) dargestellt werden kann.

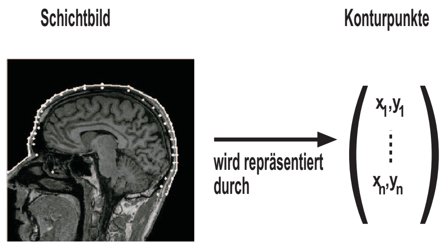


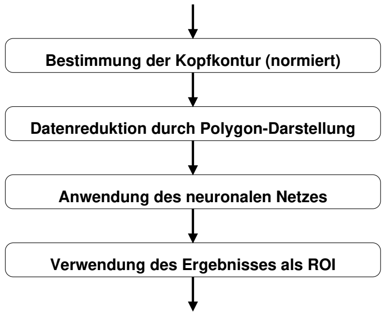
Abbildung 5.25: Die den Kopf umschließende Kontur wird durch eine Menge von Punkten im Netz repräsentiert. Das Netz selbst berechnet wiederum eine Menge von Punkten, die der ROI des Gehirns entsprechen.

Diese Konturlinie enthält aber immer noch zu viele Punkte, so daß hier eine Reduktion auf eine Teilmenge markanter Punkte erfolgt. Im genannten Beispiel hat sich die Menge von 30 Bildpunkten als gut geeignet erwiesen. Werden deutlich weniger Neuronen verwendet, kann es bei manchen Datensätzen vorkommen, daß Teilbereiche abgeschnitten werden (das heißt, die gesuchte ROI ist nicht vollständig im Polygon enthalten). Für die Verwendung von deutlich mehr Eingabeneuronen war die Anzahl vorhandener Trainingsdatensätze zu gering, so daß keine ausreichende Generalisierung erzielt werden konnte. Zudem ist zu vermuten, daß eine präzise ROI-Berechnung aufgrund der großen Variabilität der Anatomie nicht erreichbar ist. Markante Punkte sind etwa der Nasenscheitel oder der obere und hintere Scheitelpunkt des Kopfes. Zwischen diesen 30 Punkten stellt eine lineare Interpolation eine gute Approximation der Kopfkontur dar. Um eine größeninvariante Darstellung der Kontur zu erreichen, wird eine Normierung der Kopfgröße durchgeführt. Dies ist notwendig, da die Kopfgrößen sich deutlich voneinander unterscheiden können (vgl. Abbildung 2.13).

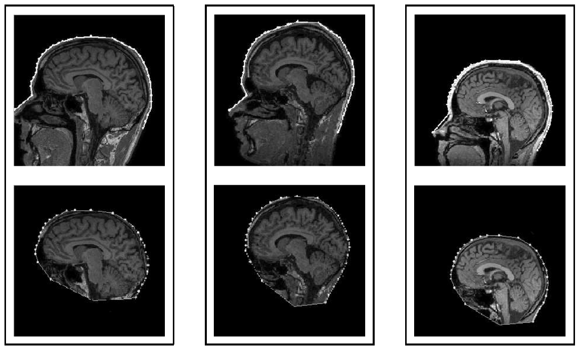
Die auf diese 30 Punkte reduzierte Kopfkontur (vgl. Abbildung 5.25) dient dann als Eingabevektor für das vorwärtsgerichtete neuronale Netz. Das Netz wird überwacht trainiert, wobei als Ausgabevektor (Sollvektor) die manuell vorgegebene Kontur der ROI dient. Diese wird ebenfalls über 30 Bildpunkte dargestellt, so daß das Netz aus 30 Eingabe- und 30 Ausgabeneuronen besteht. Getestet wurden mehrere Netze mit verschiedener Anzahl von verdeckten Schichten (1-3) mit 75 bis 1000 Knoten (jeweils mit 1000 Lernzyklen).

Für die jeweilige Anwendung ist wichtig, daß die vorgegebene Kontur ausreichend grob ist, um sicherzustellen, eine für die Weiterverarbeitung geeignete ROI zu erhalten. Für das Training eines Netzes muß ebenfalls darauf geachtet werden, eine ausreichende Spanne an Variabilitäten für die Eingabemuster abzudecken.

In der Anwendungsphase des Netzes ist wiederum die Ermittlung der Kopfkontur des Eingabedatensatzes, für den die ROI bestimmt werden soll, notwendig. Diese wird wie oben beschrieben auf einen Polygonzug mit 30 Punkten reduziert und dem Netz präsentiert, welches daraus einen Ausgabevektor mit 30 Eckpunk-



(a) Algorithmus



(b) Beispiel-Ergebnisse

Abbildung 5.26: Verwendung neuronaler Netze zur Bestimmung der Region of interest in MRT-Kopfdatensätzen. Abbildung 5.26(b) zeigt in der oberen Reihe die Eingabebilddaten mit den verwendeten Konturpunkten, die untere Reihe zeigt die Ergebnisse der Verwendung des neuronalen Netzes zu ROI-Berechnung.

ten der ROI berechnet. Abbildung 5.26 zeigt nochmals den vereinfachten Algorithmus und einige Beispielergebnisse. Dabei stellen die weißen Punkte und die weiße Kontur in Abbildung 5.26(b) den Ein- bzw. Ausgabevektor dar. Die Implementierung der verwendeten Lernverfahren und Tests über die Topologie des Netzes können in [Rit98] nachgelesen werden.

Bemerkung 5.2 Das vorgestellte neuronale Netz leistet eine Art Registrierung[22] (engl. *Template matching*), wie es ähnlich über einen expliziten Modellkopf möglich ist, wobei die Berechnung mit Hilfe des Netzes in nur einem Rechenschritt erfolgt, also einen deutliche Aufwandsreduktion bedeutet. Zudem ist die Festlegung dieses Modellkopfes implizit im Netz enthalten und kann aus realen Beispieldatensätzen trainiert werden.

Neben der Weiterverarbeitung, etwa der Segmentierung einzelner Strukturen, ist diese ROI-Bestimmung auch für die Verwendung aktiver Konturen (engl. *Snakes*) [KWT87, BI98] gut geeignet. Hier ist das Hauptproblem die Festlegung der Initialkontur, die meist manuell durchgeführt wird. Diese muß aufgrund einer starken Bewertung von Kanten über die äußere *Energiegleichung*[23] sehr dicht an der zu segmentierenden Struktur liegen. Falls genügend Trainingsbeispiele vorliegen und eine funktionale Beziehung zwischen berechenbaren Bildeigenschaften und der Initialkontur existiert, kann mit dieser Methode eine Initialkontur schnell

[22] Registrierung beschreibt die geometrische Ausrichtung zweier (oder mehrerer) Bilder, so daß identische Bildinhalte an gleichen Bildpositionen zur Deckung kommen.

[23] Die Bezeichnung *Energiegleichung* ist aus der Physik entliehen [KWT87] und beschreibt verschiedene Eigenschaften einer Kontur, die mittels einer Menge von *Konturpunkten* repräsentiert wird. Eine Erläuterung findet sich auch in [Hil00].

und einfach berechnet werden. Da aktive Konturmodelle häufig für das Verfolgen von Bildobjekten (engl. *Tracking*) eingesetzt werden, läßt sich somit ein weiterer Schritt automatisieren, bspw. für die Lippenverfolgung einer sprechenden Person, wo die Position des Mundes abhängig von der Kopfkontur ist. Fehlt aber eine derartige Beziehung, etwa bei der Verfolgung von sich bewegenden Fahrzeugen in Straßenaufnahmen, versagt diese Methode, da eine Fahrzeugposition unabhängig von anderen Konturen sein kann (mit Außnahme von der Straßenkontur selbst).

□

5.3.2 Neuronale Netze als Klassifikator

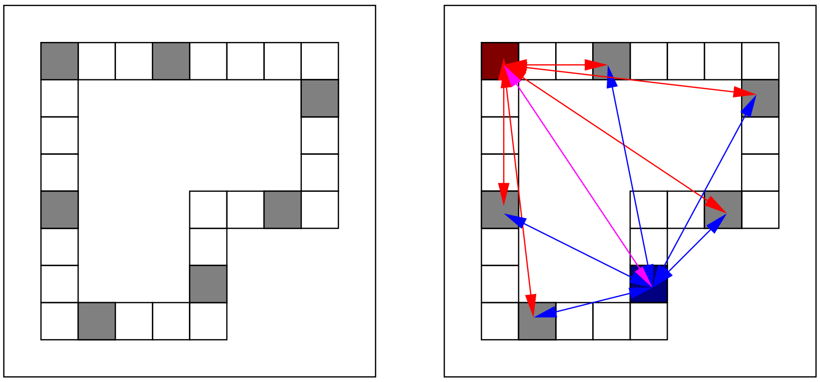
Die in Abschnitt 5.2.3 vorgestellten Beschreibungen dienen zur Interpretation natürlichsprachlicher Eigenschaften der zu klassifizierenden Objekte. Es existieren noch eine Vielzahl weiterer Beschreibungsmöglichkeiten, die aber nicht alle in intuitiver Weise verbal formulierbar sind. Eine Klassifikation aufgrund derartiger Eigenschaften mit Hilfe neuronaler Netze ist aber durchaus realisierbar, da das benötigte Wissen nicht explizit verbalisiert werden muß, sondern implizit in einem Netz enthalten ist. Beispiele solcher Beschreibungen sind (normierte, zentrale) Momente, die eine größeninvariante, lageinvariante und translationsinvariante Darstellung ermöglichen [Hab95]. Aber auch andere Eigenschaften der Kontur, bspw. die Einteilung mittels m (möglichst[24]) äquidistant auf der Kontur verteilter Punkte, kann zur Identifizierung herangezogen werden (vgl. Abbildung 5.27(a)). Anschließend werden wie in Abbildung 5.27(b) ansatzweise dargestellt, die (euklidischen) Distanzen von jedem der m Punkte zu den übrigen $n = m - 1$ Punkten als Merkmalsvektor verwendet. Je Segment existieren so m Merkmalsvektoren der Dimension n.

Bis auf die Rundungsfehler ist eine derartige Segmentrepräsentation rotationsinvariant und translationsinvariant. Eine Größeninvarianz kann dadurch erreicht werden, daß die einzelnen Abstände mit Hilfe des größten Abstands normiert werden. Allerdings werden bei dieser Kontur-Darstellung keine Löcher berücksichtigt, so daß etwa ein Ring von einem Kreis nicht unterschieden werden kann.

Werden die einzelnen Merkmalsvektoren mit $\vec{g_i},\ i \in \{1, \ldots, m\}$ bezeichnet, und d_{ij} bezeichnet den Abstand vom Punkt p_i zu p_j mit $i, j \in \{1, \ldots, m\}$, dann können folgende Merkmalsvektoren für ein Segment angegeben werden:

$$\vec{g_1} = \begin{pmatrix} d_{12} \\ d_{13} \\ d_{14} \\ \vdots \\ d_{1m} \end{pmatrix}, \quad \vec{g_2} = \begin{pmatrix} d_{23} \\ d_{24} \\ \vdots \\ d_{2m} \\ d_{21} \end{pmatrix}, \quad \cdots, \quad \vec{g_m} = \begin{pmatrix} d_{m1} \\ d_{m2} \\ d_{m3} \\ \vdots \\ d_{mn} \end{pmatrix} \tag{5.53}$$

[24]Eine äquidistante Einteilung ist aufgrund der diskreten Darstellung nicht immer möglich.



(a) Äquidistante Einteilung der Kontur

(b) Merkmalsvektoren (für 2 Punkte)

Abbildung 5.27: Einteilung eines Segmentes mit Hilfe auf der Kontur äquidistant verteilter Punkte zur Bestimmung der Merkmalsvektoren.

Zur Normierung auf das Einheitsintervall $\langle 0, 1\rangle$ wird $\overline{d}_{ij}$ mit

$$\overline{d}_{ij} = \frac{d_{ij} - d_{\min}}{d_{\max} - d_{\min}} \tag{5.54}$$

verwendet, wobei $d_{\min} = \min\{d_{ij} \mid i \neq j,\ i, j \in \{1, \ldots, m\}\}$ und $d_{\max} = \max\{d_{ij} \mid i \neq j,\ i, j \in \{1, \ldots, m\}\}$.

Die m Merkmalsvektoren $\vec{g_i}$ stellen m verschiedene *Ansichten* des Segmentes dar, die mit Hilfe eines neuronalen Netzes trainiert werden können. Dazu werden zu jedem zu trainierenden Segment die m Ansichten berechnet und in einer Trainingsdatei gespeichert. Mit Hilfe eines überwachten Trainingsalgorithmus wird das Netz trainiert. Nach dem Abschluß des Trainings kann das Netz in der Anwendungsphase eingesetzt werden, wobei es ausreichend ist, nur einen Ansichtsvektor zu berechnen, anhand dessen das Segment klassifiziert werden kann. Wie bei neuronalen Netzen üblich ist das Ergebnis der Klassifikation abhängig von der Topologie des Netzes, dem Training bzw. der Größe und Repräsentativität der Trainingsmenge und natürlich auch von der Wahl des Parameters m, also der Anzahl der verwendeten Punkte für die Einteilung des Segmentes.

Alternativ kann auch die Verwendung des Konturabstands zum Segmentschwerpunkt verwendet werden, wie in Definition 5.16 für $roundness_R$ beschrieben. Diese Darstellung ist ebenfalls rotations- und translationsinvariant und kann durch geeignete Normierung größeninvariant realisiert werden [BP92a].

In der Anwendungsphase des Netzes ist dabei zu beachten, daß das Segment in einer anderen Drehlage vorliegen kann. Dementsprechend liegen die äquidistanten Punkte nicht genau so wie in der Trainingsphase vor. Die verwendete Grundidee ist jedoch, daß die in verschiedenen Ansichten trainierte Kontur des

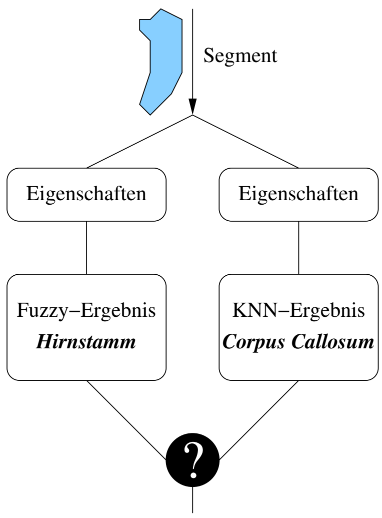


Abbildung 5.28: Problem der Verwendung unterschiedlicher Ergebnisse aus der unscharfen Klassifikation und der Klassifikation mittels neuronaler Netze.

Segmentes ähnlich zu der berechneten Ansicht des zu klassifizierenden Segmentes ist und der Generalisierungseffekt des neuronalen Netzes ausreichend ist, um die richtige Klasse zu berechnen. Um einen möglichen ersten Punkt zu bestimmen, kann der Punkt der äußeren Hülle verwendet werden, der vom Strahl des Segmentschwerpunkts in Richtung der Segmentorientierung (vgl. Definition 5.18) geschnitten wird.

Weitere Eigenschaften von Segmenten sind verwendbar, um ein neuronales Netz zu trainieren. Wichtig ist, daß ähnliche Eigenschaften über ähnliche Merkmalswerte beschrieben werden und eine ausreichende Unterscheidung gegenüber anderen Segmenten existiert.

5.3.3 Klassifikation mittels neuronaler Netze und unscharfer Beschreibungen

Auch die gemeinsame Nutzung von unscharfen Beschreibungen zur Klassifikation und der Klassifikation mittels neuronaler Netze ist möglich. Sinnvoll ist dies zur Einrichtung einer Kontrollmöglichkeit des jeweils anderen Verfahrens. Hierbei muß festgelegt werden, wie vorzugehen ist, wenn beide Verfahren zu unterschiedlichen Ergebnissen führen, wie in Abbildung 5.28 dargestellt wird. Dann stehen folgende Möglichkeiten zur Auswahl:

- Es wird einem der beiden Verfahren mehr Vertrauen geschenkt und dessen Ergebnis verwendet. Falls beide Verfahren einen Vertrauenswert zurückliefern, muß überprüft werden, ob beide Werte vergleichbar sind, bevor ggf.

das Verfahren ausgewählt wird, welches einen höheren Vertrauenswert besitzt.

- Alternativ kann auch mit geänderten Parametereinstellungen ein erneuter Segmentierungsversuch und eine erneute Klassifikation gestartet werden.

- Handelt es sich um die Berechnung eines numerischen Wertes, ist auch die Möglichkeit einer (ggf. gewichteten oder normierten) Mittelwertbildung möglich.

Wichtig ist auf jeden Fall, derartige Fälle zu protokollieren, da die Wissensbasis oder das neuronale Netz fehlerhaft bzw. unzureichend (in Bezug auf die Aufgabenstellung) sein können. Welche der oben genannten Möglichkeiten letztendlich eingesetzt wird, ist abhängig von der Anwendung und damit verbunden der Art der betrachteten Bilddaten. In Entstehung dieser Arbeit hat sich aber gezeigt, daß bei einer ausreichend umfangreichen Beschreibung der gesuchten Strukturen den Fuzzy-Methoden eine stärkere Gewichtung zuzuschreiben ist, was allerdings auch in einer relativ geringen Menge von Trainingsdaten für die verwendeten neuronalen Netze begründet ist.

5.4 Verwendung von Evolutionsstrategien in CIM^{2}BA

Evolutionäre Algorithmen können, wie in Abschnitt 3.1.4 gezeigt wurde, zur Optimierung von Daten verwendet werden, wobei die Evolutionsstrategien die direkte Verarbeitung reelwertiger Daten erlauben. Die Abbildung 5.2 stellt dar, wie die Evolutionsstrategien in CIM^{2}BA verwendet werden, nämlich zur Verfeinerung des in den Wissensbasen gespeicherten Wissens bzw. der zur Interpretation verwendeten Fuzzy-Terme.

5.4.1 Optimierung der Fuzzy-Mengen

Es hat sich in vielen Beispielanwendungen gezeigt, daß die Einstellung bzw. Optimierung der zugrundeliegenden Fuzzy-Mengen, also die Interpretation der Terme, ein langwieriger und schwieriger zyklischer Prozeß ist, der häufig zwischen Einstellung und Test wechselt. Da die Definition einer Fitneßfunktion jedoch in vielen Fällen möglich ist (wie dies für die MRT-Segmentierung schon in [HT94] gezeigt wird), ist die Optimierung der Fuzzy-Mengen mit Hilfe von Evolutionsstrategien durchführbar [Hil96]. Auch der Mensch „ist als Fitneßfunktion einsetzbar", indem er die Güte des jeweiligen Ergebnisses bewertet, wie dies z. B. im System *PhotoGenetics* [Ker00, Qbe01] durchgeführt wird.

Zunächst wird in einem ersten Schritt mit dem System ein Programm[25] P entworfen, welches das gegebene Problem lösen kann. Zumindest die Auswahl der einzelnen Abarbeitungsschritte müssen – bei geeignetem Expertenwissen – ausreichen, das Problem zu lösen. Das Programm P nutzt Fuzzy-Mengen als Interpretation der Beschreibungen des Experten. Diese gilt es zu optimieren. In der Regel wird nur eine vom Experten oder von statistischen Methoden ermittelte Beschreibung grob vorgegeben [Spi93], auch ist eine zufällige oder aber gleichverteilte Darstellung über dem gesamten Universum als Initialisierungswert erlaubt, was allerdings dem Optimierungsaufwand und somit den Zeitbedarf deutlich erhöht (eine näher Abschätzung kann hier nicht abgegeben werden, da dies stark vom Programm und den verwendeten Operatoren abhängt).

Nachdem die Initialisierung abgeschlossen ist, kann mit der Optimierung mittels evolutionärer Algorithmen begonnen werden. Die Idee ist hierbei, die Menge der Fuzzy-Mengen als ein einzelnes Individuum V_i aufzufassen. Dazu sind die Fuzzy-Mengen geeignet zu kodieren[26].

Bei der Kodierung ist zu berücksichtigen, welche Eigenschaften der Fuzzy-Mengen variabel sind. Bspw. ist festzulegen, ob Form oder Höhe der Fuzzy-Mengen modifiziert werden dürfen. Weitere Randbedingungen sind durch die relative Lage der Terme zueinander gegeben. Dies bedeutet hier, daß Terme einer Variablen nicht andere Terme derselben Variablen vollständig überdecken dürfen. Eine genaue Beschreibung der Umsetzung findet sich in Abschnitt 6.1.4.

Das Hauptproblem bei der Anwendung von evolutionären Algorithmen ist die Definition der Fitneßfunktion. Für die Segmentierung von Bilddaten kann hierzu das Differenzenbild zwischen Soll- und Istausgabe verwendet werden, ähnlich wie in [HT94, FTHB94] beschrieben (dort wurden die Grauwerte aus Definition 5.31 aufsummiert).

Eine weitere, vom Grauwert unabhängige Funktion, ist ebenfalls möglich. Diese liefert ebenfalls als Ergebnis einen Ganzzahlwert aus $\mathbb{N}_0$, so daß ein einfacher Vergleich der Güte der Individuen möglich ist. Dazu wird über alle betrachteten Bilder, die zur Optimierung der Fuzzy-Mengen herangezogen werden, die Summe derjenigen Bildpunkte berechnet, die im Ist- und im Sollbild unterschiedlich sind (vgl. Definition 5.32).

Definition 5.31 (Differenzenbild 1) Seien I_s und I_i zwei Bilder der Größe $M \times N$. Dann ist das Differenzenbild I_{d_1} wie folgt definiert:

$$I_{d_1}(x,y) = |I_s(x,y) - I_i(x,y)| \quad \forall (x,y) \in M \times N \tag{5.55}$$

□

Definition 5.32 (Differenzenbild 2) Seien I_s und I_i zwei Bilder der Größe $M \times N$. Dann ist das Differenzenbild I_{d_2} wie folgt definiert:

[25] Wie ein solches Programm aussehen kann, wird in Abschnitt 6.2 erläutert.

[26] Eine konkrete Kodierung wird in Kapitel 6 in der Definition 6.1 vorgestellt.

$$I_{d_2}(x,y) = \begin{cases} 1 & \text{falls } I_s(x,y) \neq I_i(x,y) \\ 0 & \text{sonst} \end{cases} \quad \forall (x,y) \in M \times N \tag{5.56}$$

□

Bemerkung 5.3 Für den Fall, daß I_s und I_i Binärbilder (aus dem Wertebereich $\{0,1\}$) sind, gilt: $I_{d_1} = I_{d_2}$. Beide Fitneßfunktionen sind sinnvoll und abhängig von der Aufgabenstellung anzuwenden. Entscheidend ist, ob der Grauwert eine zusätzliche Information zur Fitneß liefert oder ob der Unterschied zweier Punkte an sich die notwendige Information ist. □

Entsprechend der in Abschnitt 3.1.4 beschrieben Algorithmen kann die Menge der Fuzzy-Mengen – wie in Abbildung 5.29 dargestellt – optimiert werden. Wie bereits gesagt wurde, sollte die Initialisierung bereits eine gute Lösung darstellen, da für jede Fitneßbewertung das Programm P abgearbeitet werden muß. Das kann je nach Problemstellung sehr rechenintensiv sein, so daß ein enormer Zeitbedarf für die Optimierung benötigt werden kann. Da außerdem die Fuzzy-Mengen nicht auf einen bestimmten Bilddatensatz hin trainiert werden sollen, ist die Bewertung jedes Individuums V' auch noch für mehrere Datensätze zu testen, was den Rechenaufwand weiter erhöht. Insgesamt muß das Programm P für n zu testende Datensätze und λ Nachkommen abgearbeitet werden, also sind bei G Generationen $n \cdot \lambda \cdot G$ Durchläufe des Programms notwendig.

Für die manuelle Optimierung ist allerdings der gleiche Aufwand notwendig. Weiterhin spricht für die Optimierung mittels evolutionärer Algorithmen die Möglichkeit, daß diese unabhängig von menschlichen Ressourcen durchführbar ist, eine Parallelisierung (vgl. Abschnitt 6.1.4 und Abbildung 6.23) und ein Dauerbetrieb (24 Stunden) sind deutlich kostengünstiger realisierbar.

Eine erfolgreiche Umsetzung dieser Methode für die Segmentierung des Gehirns aus MRT-Daten wird in [HT94] beschrieben. In Abschnitt B.3.1 sind beispielhaft eine Anwendung und die Ergebnisse dokumentiert.

5.4.2 FOM-Optimierung

Die Optimierung anderer Parameter ist auf gleiche Weise möglich. In CIM^{2}BA bieten sich dafür verschiedene Parameter an. Neben den (Standard-)Fuzzy-Mengen können auch FOMs optimiert werden, wenn eine geeignete Menge von Referenzobjekten vorliegt. Besondere Bedeutung kommt dabei dem Parameter δ zu, der den Grad der Unschärfe des FOM bestimmt. Aber auch der Typ der verwendeten Fuzzy-Menge (vgl. Abbildung 5.20) ist entsprechend der Anwendung zu optimieren. Während zudem eine Variation der Form denkbar ist[27], ist eine

[27] Ähnlich wie bei aktiven Konturmodellen ist eine Veränderung der Position von Stützstellen der Kontur realisierbar.

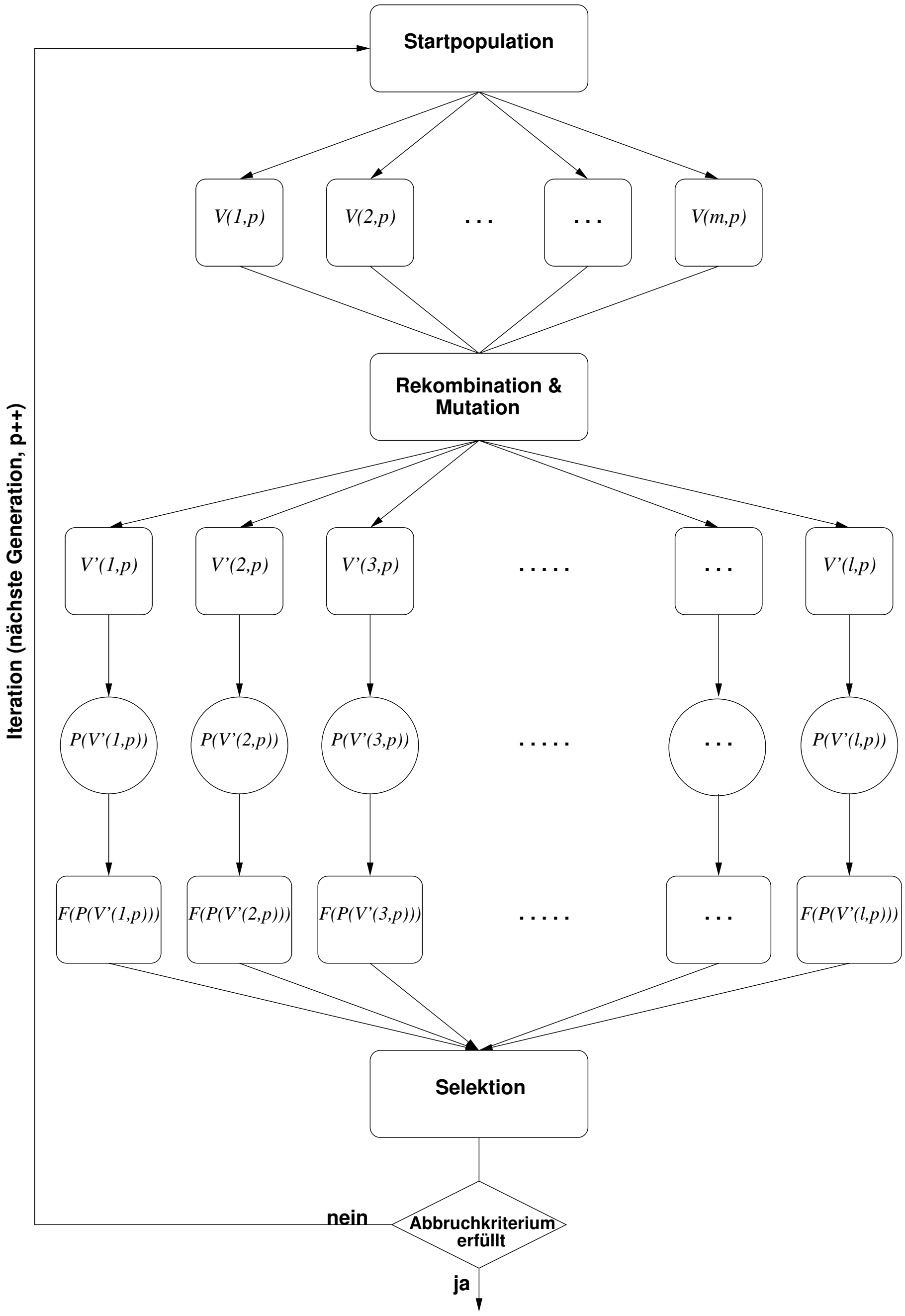


Abbildung 5.29: Anwendung evolutionärer Algorithmen (mit m Eltern und l Nachkommen) zur Optimierung einer in einer FM-Basis definierten Menge von Fuzzy-Mengen, die als Individuum aufgefaßt und im Programm P genutzt und bezüglich der Fitneßfunktion F bewertet wird.

reine Skalierung nicht sinnvoll. Diese sollte anhand des gewählten Modells durchgeführt werden, da die Größe des FOM immer in Bezug auf ein Referenzmodell zu betrachten ist. Verschiebungen können mit FOMs nicht betrachtet werden, da die Überlagerung im Flächenschwerpunkt eine Translationsinvarianz bewirkt.

Für die Optimierung von FOMs mit Hilfe von Evolutionsstrategien muß eine geeignete Kodierung der Darstellung erfolgen. Hier wird nur der einfache Fall betrachtet, daß der Parameter δ und der Typ der verwendeten Fuzzy-Menge θ (vgl. Definition 6.2) modifiziert werden. Dementsprechend besteht ein Individuum aus dem Paar (δ, θ) und einer nicht varianten Punktmenge P, welche die Kontur des FOM beschreiben. Als Fitneßfunktion kann die Zugehörigkeitsfunktion aus Definition 5.30 verwendet werden. Diese wird für alle Beispiele (korrekt segmentierte Instanzen der Objektklasse, für die das FOM ein Modell ist) berechnet und gemittelt. Ziel ist eine Maximierung des gemittelten Zugehörigkeitswertes, wobei auch Negativbeispiele (Beispiele, die nicht dem Modell entsprechen) notwendig sind, um eine maximale Ausdehnung des FOM zu vermeiden. Ohne derartige Negativbeispiele ist der Parameter δ einfach maximal zu wählen, um einen hohem Zugehörigkeitgrad zu erreichen. Bei einer Optimierung der Form oder Größe kann die gleiche Fitneßfunktion verwendet werden, hier ist jedoch eine geeignete Repräsentation des FOM als Individuum zu finden.

5.5 Verwendung der einzelnen Methoden

Die bisher vorgestellten Methoden finden sich in einzelnen Bausteinen des Systems CIM^{2}BA/P, die in der Regel sequentiell, parallel oder wie die Wissensbasisoptimierung nur eimalig oder sehr selten, abgearbeitet werden, um eine Bildanalyse durchzuführen. Eine detailliertere Beschreibung der Umsetzung der hier vorgestellten Konzepte findet sich im folgenden Kapitel 6. Gesteuert wird der Ablauf über eine Steuerkomponente, die aufgabenabhängig programmiert werden kann. Die Auswahl aus den Komponenten erfolgt mittels einer Programmierschnittstelle. Diese erlaubt eine stapelgesteuerte Anwendung der Bausteine, wobei ergebnisabhängige Steuerungen möglich sind, etwa ergebnisabhängige Terminierung oder ergebnisabhängige Neuberechnungen. Anwender dieses Baukastens ist ein Bildverarbeiter, der einen CI-unterstützen Werkzeugkasten für seine Aufgabe erhält. Neben den einzelnen Operatoren sind auch die Wissenbasis und die geeignete Interpretation der Fuzzy-Mengen zu bestimmen bzw. zu generieren. Als Ergebnis der Entwicklung steht ein Programm P (vgl. Abschnitt 5.4), mit dem ein Endanwender (ein Mediziner) in der Lage ist, Analysen[28] durchführen zu können, wobei der eigentliche Programmteil verborgen bleibt und lediglich mittels einer Oberfläche (GUI) verbunden ist (vgl. Abbildung 5.3). Die Entwicklung der GUI

[28] Die Analyse beschränkt sich auf die Bildsegmentierung und Eigenschaftenpräsentation der einzelnen Segmente, eine Diagnose ist hier nicht gemeint.

wird hier allerdings nicht weiter betrachtet (entsprechende Richtlinien finden sich etwa in [MHJ98]).

Neben der bedingten Auswahl von Einzelkomponenten (bspw. erst die Verwendung schnell berechenbarer Komponenten und erst im Bedarfsfall komplexere Berechnungen durchzuführen) kommt für einen Iterationsschritt besonders eine Parameteränderung in Frage. Zu diesen Parametern können Zugriffe auf die Wissensbasis zählen oder auf die Interpretation der Terme. Das heißt, Parameter der Fuzzy-Mengen wie Typ, Breite des Kerns oder des Trägers können angepaßt werden, ebenso wie die Breite δ eines FOM (vgl. Abbildung 5.20). Eine Änderung gesetzter Schwellenwerte erlaubt zudem eine Änderung (in beide Richtungen) der Anforderung an ein gefundenes Segment. Zum Verständnis wird ein ausführliches Beispiel des Systems in Kapitel 7 vorgestellt, während Teilkomponenten auch in einfacher Form in Anhang B präsentiert werden.

6 Beschreibung des Systems CIM^{2}BA/P

Die im vorherigen Kapitel 5 vornehmlich theoretisch beschriebenen Konzepte werden an dieser Stelle durch die Vorstellung des Prototypen CIM^{2}BA/P ergänzt, der weite Teile des entwickelten Rahmenmodells in Form von konkreten Programmkomponenten umsetzt [Hil01]. Das angefügte „P“ deutet hierbei an, daß es sich um die zu CIM^{2}BA gehörende Programmierumgebung mit den einzelnen spezifischen Werkzeugen handelt. Der Prototyp wird durch eine konkrete Beispielanwendung, der Segmentierung von MRT-Datensätzen des Kopfes, in Kapitel 7 ergänzt. Das Beispiel ist ein seit Jahren bekanntes Problem, dessen Lösung für verschiedene praktische Fragestellungen in der modernen medizinischen Forschung von großer Bedeutung ist. Somit kann die Umsetzung des CIM^{2}BA-Rahmenmodells tatsächlich sinnvoll in der Praxis Anwendung finden.

Neben der im System CIM^{2}BA/P realisierten Programmierumgebung zur Anwendung der einzelnen Operatoren, werden weiterhin die Werkzeuge zur Verwaltung und Bearbeitung der Wissensbasis, Regelbasis, der Eingabe der Fuzzy-Mengen, sowie der künstlichen neuronalen Netze vorgestellt und deren Repräsentationsformat aufgezeigt. Auch die Beschreibung der Verwendung von den in CIM^{2}BA/P geschriebenen Programmen im EA-Werkzeug erfolgt an dieser Stelle. Abbildung 6.1 zeigt eine Übersicht über die einzelnen Teile von CIM^{2}BA/P.

Bemerkung 6.1 Die im folgenden Text dargestellten Größenverhältnisse der Bildschirmabdrucke entsprechen nicht den realen Größenverhältnissen, sondern wurden dem Papierformat angepaßt. Es werden auch nicht alle Funktionen im Detail erklärt. Es soll lediglich ein Einblick gegeben werden, der die Grundzüge der Implementierung verdeutlicht. □

6.1 Programmkomponenten

Die Programmkomponenten beinhalten einzelne Teile des Systems zur Bearbeitung der verschiedenen Inhalte, die für eine Bildanalyse benötigt werden. Hierbei handelt es sich um Komponenten, die als Objekte in eine Oberfläche eingebunden und auch von der Programmierumgebung (vgl. Abschnitt 6.2) aufgerufen werden

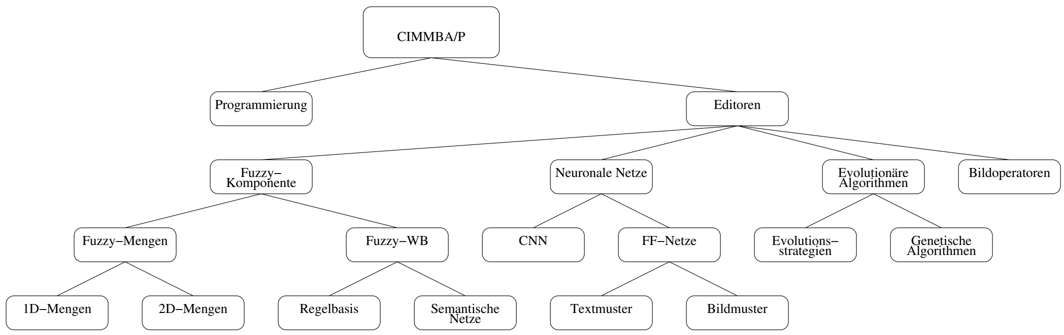


Abbildung 6.1: Struktur des Systems CIM²BA/P.

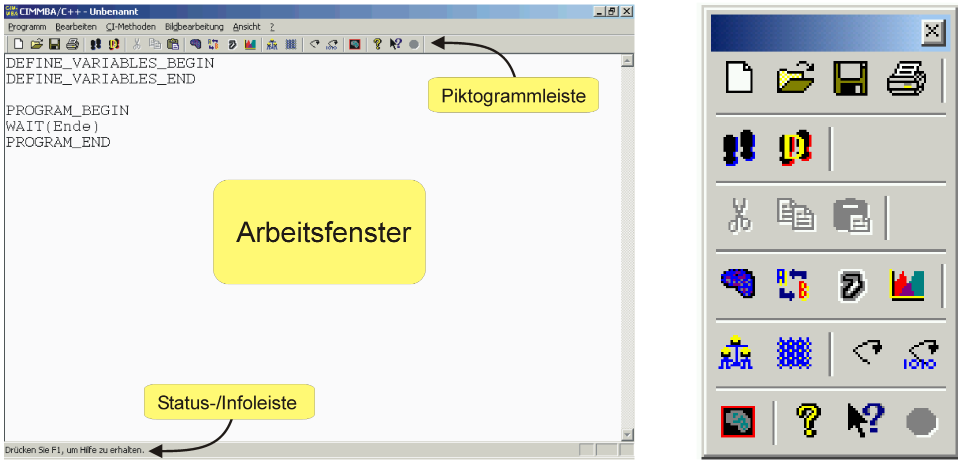
können. Abbildung 6.2(a) zeigt die Oberfläche des Systems, von der aus alle einzelnen Komponenten, wie sie im folgenden beschrieben sind, gestartet werden. In Abbildung 6.2(b) sind nochmals vergrößert die *Piktogramme* zu sehen. Weiterhin dient die Oberfläche als Eingabefenster für die Programmierumgebung, so daß nach dem Start unmittelbar mit der Programmierung begonnen werden kann.

6.1.1 Bildoperatoren

Im Bildbetrachter kann der Anwender die einzelnen Operatoren nutzen, die auf Bilddaten angewendet werden können, um deren Wirkung für den Einsatz in einem Programm auszutesten. Zunächst wählt der Anwender einen Bilddatensatz aus, wobei Formate wie RAW (reine Bildinformationen), VFF (Volume File Format) mit einem Beschreibungsteil (engl. *Header*) davor oder DICOM als mittlerweile Standard gewordenes Format für radiologische Bilddaten unterstützt werden. Zusätzlich wird das BMP-Format unterstützt, um Ergebnisse der Anwendung bildlich dokumentieren zu können. Eine übersichtliche Beschreibung über die im medizinischen Umfeld bedeutsamen Formate und dazugehörige weitere Literatur findet sich in [LHH02].

Das eingeladene Bild ist in der Quell- und Ergebnisdarstellung zu sehen, um einen direkten Vergleich bei der Anwendung einer Operation zu ermöglichen. Operationen, die Parameter erfordern, fragen diese über ein Parameterabfragefenster mit entsprechender Beschreibung der benötigten Parameter ab. Neben der Auswahl der angezeigten Schicht bei 3D-Datensätzen sind verschiedene Sichten (sagittal, coronar oder transversal) auf den Datensatz möglich. Weiterhin lassen sich Informationen über den Datensatz und darin enhaltene Teilbereiche abfragen, die zuvor mit einem Eingabegerät markiert wurden. In Abbildung 6.3 ist das Werkzeugfenster dargestellt.

Ziel dieser Komponente ist das Austesten der Wirkung der einzelnen Operatoren auf die Bilddaten, die bearbeitet werden sollen. Dadurch können ohne Programmierschritte schon erste Erfahrungen gewonnen werden. Zudem können



(a) Gesamtoberfläche (b) Piktogramme

Abbildung 6.2: Oberfläche und Piktogrammleiste des Systems CIM²BA/P. Von hier aus können alle Werkzeuge über die Menüleiste oder die Piktogrammleiste gestartet werden.

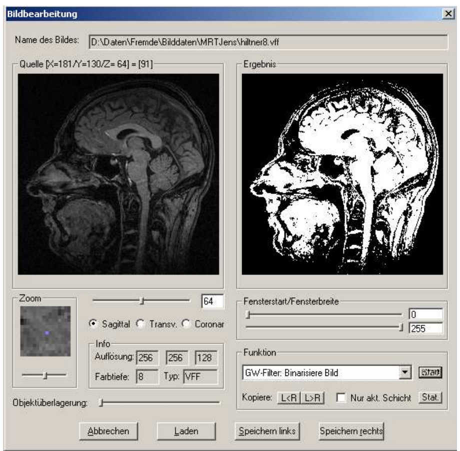


Abbildung 6.3: Werkzeug zur Bearbeitung und Betrachtung der Bilddaten.

Teilbereiche markiert und Eigenschaften wie *Grauwertverteilung* oder *Größe* abgefragt werden, um damit die Interpretationen der Eigenschaften befüllen zu können.

6.1.2 Wissensrepräsentation

Die Verwendung von Wissen besitzt in der gesamten Arbeit eine herausragende Bedeutung. Daher erfolgt an dieser Stelle eine konkrete Darstellung, wie dieses Wissen repräsentiert und implementiert werden kann.

6.1.2.1 Fuzzy-Mengen

Da der Begriff der Fuzzy-Menge eine zentrale Rolle spielt, werden an dieser Stelle die in CIM^{2}BA/P definierten Typen und Interpretationsmöglichkeiten von Fuzzy-Mengen vorgestellt (Definition 6.1). Die Liste ist dabei erweiterbar, jedoch wurde im Hinblick auf die leichte Berechenbarkeit der Zugehörigkeit und den Wunsch nach einer Optimierung (vgl. Abschnitt 5.4) auf eine einfache Repräsentationsform geachtet. Beispiele für die verschiedenen Typen der Fuzzy-Mengen sind in Abbildung 6.4 dargestellt. Die Abbildung 6.5 zeigt mögliche Formen der Mengen (am Beispiel vom Typ $\theta = 1$ (*linear*)).

Definition 6.1 (CIM^{2}BA/P-Fuzzy-Menge) Eine CIM^{2}BA/P-Fuzzy-Menge wird definiert durch ein 9-Tupel $(\theta, a_1, a_2, a_3, a_4, h_1, h_2, h_3, \triangle)$, wobei θ den Typ der Variablen (vgl. Definition 6.2), $[a_1, a_4]$ den Träger und $[a_2, a_3]$ den Bereich der Höhe h_2 definiert (für $h_2 = 1$ also den Kern (vgl. Definition 3.6)). $\triangle \in \{yes, no\}$ gibt an, ob die Menge triangulär ($a_3 \stackrel{\text{def}}{=} a_2$) sein soll. h_1 gibt die Höhe an der Stelle a_1 an, h_3 die Höhe an der Stelle a_4.
Dabei gilt $a_1 \leq a_2 \leq a_3 \leq a_4$, wobei $a_i \in \mathbb{U}$ für $i \in \{1, \ldots, 4\}$. $\mathbb{U}$ ist dabei kontextabhängig von der betrachteten linguistischen Variablen $\mathcal{F}_\diamond$ (vgl. Definition 5.1). Weiterhin gilt $h_1 \leq h_2$, $h_3 \leq h_2$, $h_1, h_2, h_3 \in \langle 0, 1 \rangle$. Die Fuzzy-Menge ist normal, falls $h_2 = 1$ gilt. □

Definition 6.2 (CIM^{2}BA/P-Fuzzy-Mengen-Typen) Eine CIM^{2}BA/P-Fuzzy-Menge $\mu_\theta(x)$ für $x \in [a_1, a_4]$ ist wie folgt für den Typ θ, $\theta \in \{1, \ldots, 5\}$ definiert:

$$\mu_1(x) \stackrel{\text{def}}{=} \begin{cases} h_1 + \frac{x-a_1}{a_2-a_1} \cdot (h_2 - h1) & \text{falls } x \in [a_1, a_2]\ ,\ a_1 \neq a_2 \\ h_2 & \text{falls } x \in [a_2, a_3] \\ h_2 - \frac{x-a_3}{a_4-a_3} \cdot (h_2 - h_3) & \text{falls } x \in [a_3, a_4]\ ,\ a_3 \neq a_4 \end{cases} \tag{6.1}$$

$$\mu_2(x) \stackrel{\text{def}}{=} \begin{cases} h_1 + 2 \cdot \left(\frac{x-a_1}{a_2-a_1}\right)^2 \cdot (h_2 - h_1) & \text{falls } x \in [a_1, \frac{a_1+a_2}{2}]\ ,\ a_1 \neq a_2 \\ h_2 - 2 \cdot \left(\frac{a_2-x}{a_2-a_1}\right)^2 \cdot (h_2 - h_1) & \text{falls } x \in [\frac{a_1+a_2}{2}, a_2]\ ,\ a_1 \neq a_2 \\ h_2 & \text{falls } x \in [a_2, a_3] \\ h_2 - 2 \cdot \left(\frac{x-a_3}{a_4-a_3}\right)^2 \cdot (h_2 - h_3) & \text{falls } x \in [a_3, \frac{a_3+a_4}{2}]\ ,\ a_3 \neq a_4 \\ h_3 + 2 \cdot \left(\frac{a_4-x}{a_4-a_3}\right)^2 \cdot (h_2 - h_3) & \text{falls } x \in [\frac{a_3+a_4}{2}, a_4]\ ,\ a_3 \neq a_4 \end{cases} \tag{6.2}$$

$$\mu_3(x) \stackrel{\text{def}}{=} \begin{cases} h_1 + \sqrt{\frac{x-a_1}{2\cdot(a_2-a_1)}} \cdot (h_2 - h_1) & \text{falls } x \in [a_1, \frac{a_1+a_2}{2}]\ ,\ a_1 \neq a_2 \\ h_2 - \sqrt{\frac{a_2-x}{2\cdot(a_2-a_1)}} \cdot (h_2 - h_1) & \text{falls } x \in [\frac{a_1+a_2}{2}, a_2]\ ,\ a_1 \neq a_2 \\ h_2 & \text{falls } x \in [a_2, a_3] \\ h_2 - \sqrt{\frac{x-a_3}{2\cdot(a_4-a_3)}} \cdot (h_2 - h_3) & \text{falls } x \in [a_3, \frac{a_3+a_4}{2}]\ ,\ a_3 \neq a_4 \\ h_3 + \sqrt{\frac{a_4-x}{2\cdot(a_4-a_3)}} \cdot (h_2 - h_3) & \text{falls } x \in [\frac{a_3+a_4}{2}, a_4]\ ,\ a_3 \neq a_4 \end{cases} \tag{6.3}$$

$$\mu_4(x) \stackrel{\text{def}}{=} \begin{cases} h_2 - \sqrt{1 - \left(\frac{x-a_1}{a_2-a_1}\right)^2} \cdot (h_2 - h_1) & \text{falls } x \in [a_1, a_2]\ ,\ a_1 \neq a_2 \\ h_2 & \text{falls } x \in [a_2, a_3] \\ h_2 - \sqrt{1 - \left(\frac{a_4-x}{a_4-a_3}\right)^2} \cdot (h_2 - h_3) & \text{falls } x \in [a_3, a_4]\ ,\ a_3 \neq a_4 \end{cases} \tag{6.4}$$

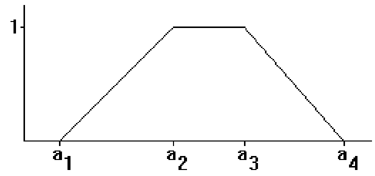
$$\mu_5(x) \stackrel{\text{def}}{=} \begin{cases} h_1 + \sqrt{1 - \left(\frac{a_2-x}{a_2-a_1}\right)^2} \cdot (h_2 - h_1) & \text{falls } x \in [a_1, a_2]\ ,\ a_1 \neq a_2 \\ h_2 & \text{falls } x \in [a_2, a_3] \\ h_3 + \sqrt{1 - \left(\frac{x-a_3}{a_4-a_3}\right)^2} \cdot (h_2 - h_3) & \text{falls } x \in [a_3, a_4]\ ,\ a_3 \neq a_4 \end{cases} \tag{6.5}$$

□

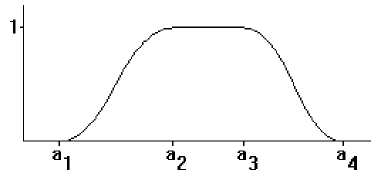
Bemerkung 6.2 Die Einschränkung $h_1 \leq h_2$ und $h_3 \leq h_2$ in Definition 6.1 garantiert dabei, daß die Fuzzy-Mengen konvex (vgl. Definition 3.7) sind. □

Festlegung 6.1 (Schreibweise für Fuzzy-Mengen-Definitionen) Um einen linguistischen Term $\mathcal{F}$ einfach zu charakterisieren, kann er dabei über folgendes Tupel definiert werden (vgl. Definition 6.1).

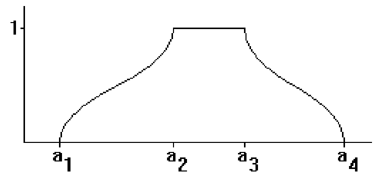
$$\mathcal{F} = (\theta, a_1, a_2, a_3, a_4, h_1, h_2, h_3, \triangle) \tag{6.6}$$



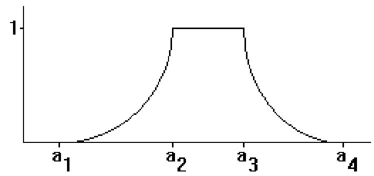
(a) Linear



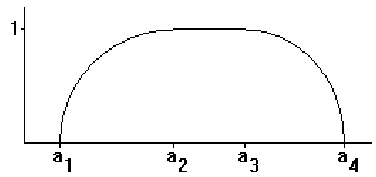
(b) Sigmoidal



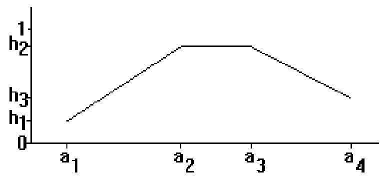
(c) Invers sigmoidal



(d) Radial

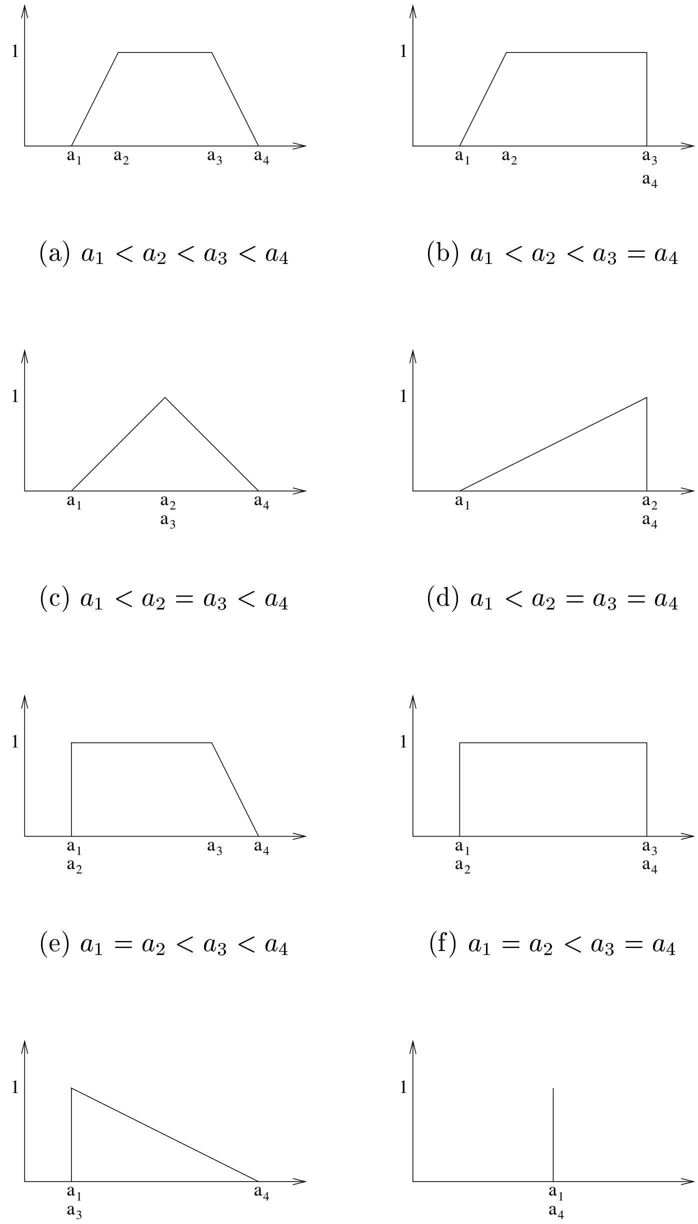


(e) Invers radial



(f) $h_1 \neq 0$, $h_2 \neq 1$, $h_3 \neq 0$

Abbildung 6.4: Darstellung der möglichen Fuzzy-Mengen aus den Gleichungen (6.1) bis (6.5) mit $a_1 < a_2 \leq a_3 < a_4$. 6.4(f) zeigt die Verwendung unterschiedlicher Höhen, während für 6.4(a) bis 6.4(e) $h_1 = h_3 = 0$, $h_2 = 1$ gilt.



(a) $a_1 < a_2 < a_3 < a_4$ (b) $a_1 < a_2 < a_3 = a_4$

(c) $a_1 < a_2 = a_3 < a_4$ (d) $a_1 < a_2 = a_3 = a_4$

(e) $a_1 = a_2 < a_3 < a_4$ (f) $a_1 = a_2 < a_3 = a_4$

(g) $a_1 = a_2 = a_3 < a_4$ (h) $a_1 = a_2 = a_3 = a_4$

Abbildung 6.5: Die exemplarische Darstellung möglicher Fuzzy-Mengen des Typs $\theta = 1$ (linear) über einem Universum $\mathbb{U}$.

Um eine kürzere Notation erreichen zu können, sind die folgenden Schreibweisen verwendbar. Soll die Höhe nicht variabel sein, können die Parameter h_1, h_2, h_3 zu einem Schlüsselwort `no` zusammengefaßt werden, was einem $h_1 = 0.0$, $h_2 = 1.0$ und $h_3 = 0.0$ entspricht. Eine Fuzzy-Menge kann dann wie folgt dargestellt werden:

$$\mathcal{F} = (\theta, a_1, a_2, a_3, a_4, \texttt{no}, \triangle) \tag{6.7}$$

Der Parameter $\triangle$ definiert, ob $a_3 \stackrel{\text{def}}{=} a_2$ gilt und kann mit `yes` oder `no` angegeben werden. Um eine weitere Verkürzung in der Schreibweise zu erreichen, können die Parameter h_1, h_2, h_3 und $\triangle$ auch entfernt werden. Dann wird die Höhe auf Eins fixiert und keine dreieckige Fuzzy-Menge gefordert, so daß vereinfacht ein 5-Tupel verwendet werden kann.

$$\mathcal{F} = (\theta, a_1, a_2, a_3, a_4) \tag{6.8}$$

Nur das Weglassen des Parameters $\triangle$ entspricht ebenfalls der Verwendung eines `no` und führt zu einem 8-Tupel

$$\mathcal{F} = (\theta, a_1, a_2, a_3, a_4, h_1, h_2, h_3) \tag{6.9}$$

bzw. einem 6-Tupel

$$\mathcal{F} = (\theta, a_1, a_2, a_3, a_4, \texttt{no}) \tag{6.10}$$

Da die Parameteranzahl für jede dieser Konventionen eindeutig ist, wird eine korrekte Interpretation der Schreibweise gewährleistet. Die Differenzierung der Schreibweisen ist teilweise notwendig, damit bei einer Optimierung der Fuzzy-Mengen mittels evolutionärer Algorithmen (vgl. Abschnitt 5.4.1) die Randbedingungen korrekt berücksichtigt werden können. Ein weiterer Grund ist die vereinfachte (verkürzte) Schreibweise. □

6.1.2.2 Interpretation unscharfer Variablen und Terme

Ausgehend von den in Definition 6.1 definierten möglichen Fuzzy-Mengen und der in Festlegung 6.1 benutzten Schreibweise werden die Interpretationen in einer Basis von Fuzzy-Mengen (FM-Basis) verwaltet. Eine einzelne Variable $\mathcal{F}_\diamond$ und die dazugehörigen Terme werden dabei, wie in Abbildung 6.6 gezeigt, gespeichert.

Die Variablendefinition beinhaltet das Universum, über das sie definiert ist und die Definition, ob es sich hierbei um ein ganzzahliges oder um ein reelwertiges Universum handelt. Bei dem Schlüsselwort `FUNCTION` handelt es sich um

Definition: Fuzzy-Variable

```
DEFINE_FUZZY_VARIABLE [context.|*]name
  UNIVERSE [lowest:highest] TYPE {real|integer}
  FUNCTION functionname

  TERM name_1 = (type_1,
                value_a11, value_a12, value_a13, value_a14[*|,
                [value_h11, value_h12, value_h13|no],
                [yes|no]])

  TERM name_2 = (type_2,
                value_a21, value_a22, value_a23, value_a24[*|,
                [value_h21, value_h22, value_h23|no],
                [yes|no]])

  TERM name_3 SEE a_context.a_fuzzy_variable.a_term_name

DEFINE_FUZZY_VARIABLE_END
```

Abbildung 6.6: Definition einer Fuzzy-Variablen „name“ im Kontext „context“ mit 3 unterschiedlichen Termen (vgl. Festlegung 6.1).

eine Anweisung, welche die Variable an eine der Berechnungsvorschriften bindet, die in Abschnitt 5.2.3 definiert werden. Die in Abschnitt 5.2.1 erwähnten Synonyme werden in der FM-Basis mittels Referenzen (Verweise) auf andere Terme realisiert, wie dies in Abbildung 6.6 durch die `SEE`-Anweisung dargestellt wird. Dadurch ist die Sprache, die durch die linguistischen Terme und Variablen definiert wird, erweiterbar[1], wobei trotzdem sichergestellt ist, daß die verschiedenen Synonyme gleich interpretiert werden. Ein solcher Verweis ist sowohl für einzelne Terme wie auch für ganze Variablen anwendbar. Bei der Verwendung eines Verweises für einen einzelnen Term müssen die Universen übereinstimmen, in denen die Terme definiert sind. Ein Beispiel für eine FM-Basis ist in Abbildung B.16 und Abbildung B.17 dargestellt.

Die Definition und Verwaltung unscharfer Variablen und Mengen kann im Fuzzy-Mengen-Werkzeug, wie es in Abbildung 6.7 dargestellt ist, erfolgen. Der Anwender erfährt hier die Möglichkeit, Variablen (mit Kontext, Universum und Datentyp) zu definieren, die zugehörigen Terme mit Träger und Kern festzulegen, ebenso wie den Typ des Terms. Als Ergebnis wird eine Fuzzy-Mengen-Basis erzeugt, die im Aufbau den Beschreibungen aus Abbildung 6.6 entspricht. Mittels einer Konsistenzprüfung läßt sich feststellen, ob die Terme einer Variablen sich bspw. im Kern überlappen oder ob Terme andere Terme überdecken. Gemäß des Prüfungsergebnisses wird eine entsprechende Meldung generiert.

[1]Erweiterbar in dem Sinne, daß intuitivere Begriffe für denselben Term verwendet werden können, etwa wenn die Universen, über die die Variablen definiert werden, normiert sind und semantisch äquivalente Terme verwendet werden, bspw. (klein, mittel, groß) und (kurz, mittel, lang) über einem Universum $\mathbb{U}$.

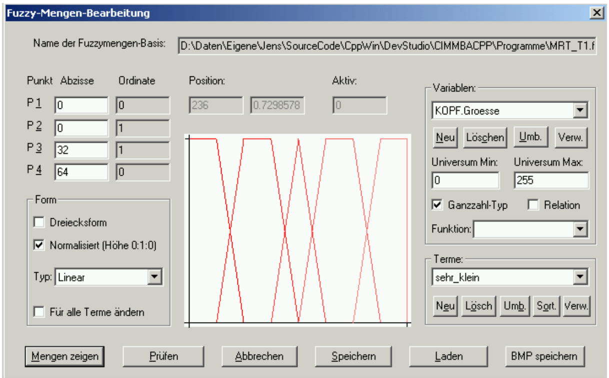


Abbildung 6.7: Werkzeug zur Bearbeitung der Fuzzy-Mengen.

Definition: FOM

```
DEFINE_FOM_VARIABLE [context|*].name
  DIMX dx
  DIMY dy
  FSTYPE t
  CORE_WIDTH cw
  SUPPORT_WIDTH sw
  POINTLIST_BEGIN
    P:x1 y1
    P:x2 y2
    :
  POINTLIST_END
DEFINE_FOM_VARIABLE_END
```

Abbildung 6.8: Definition eines Fuzzy-Objekt-Modells „name" im Kontext „context".

6.1.2.3 Fuzzy-Objekt-Modelle

Fuzzy-Objekt-Modelle werden in CIM²BA/P als Punktliste verwaltet, wie dies in Abbildung 6.8 dargestellt wird. Zudem wird für die Betrachtung weiterer Eigenschaften die horizontale und vertikale Ausdehnung eines Bildpunktes mit dem Schlüsselwort `DIMY` (respektive `DIMX`) gespeichert. Die zu verwendende Fuzzy-Menge bei der Interpretation des FOM wird mit dem Schlüsselwort `FSTYPE` hinterlegt (vgl. Definition 6.2), die Breite des Kerns und des Trägers mit dem Schlüsselwort `CORE_WIDTH` bzw. `SUPPORT_WIDTH`. Die Kernpunktmenge selbst wird in der Umgebung mit `POINTLIST_BEGIN` und `POINTLIST_END` aufgelistet.

Die Eingabe von Fuzzy-Objekt-Modellen erfolgt im FOM-Editor. Dieser erlaubt die Zeichnung eines Modells mit Hilfe eines geeigneten Eingabegerätes, welches sinnvoller Weise ein Grafiktablett ist. Das Einblenden eines Referenzmodells zur Größenkontrolle kann direkt im Zeichenfeld erfolgen. Art und Breite der Unschärfe können an dieser Stelle ebenfalls festgelegt werden. Schließlich kann hier

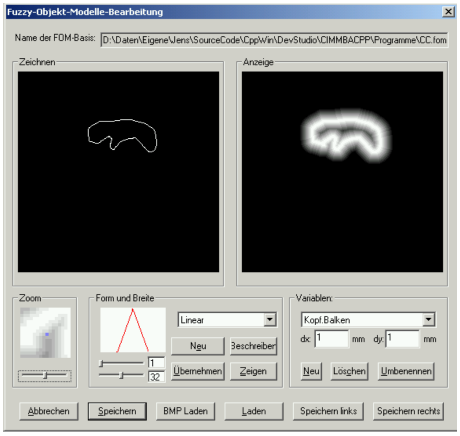


Abbildung 6.9: Werkzeug zur Bearbeitung der FOM.

auch eine direkte grafische Kontrolle erfolgen, wie dies beispielhaft in Abbildung 6.9 gezeigt wird.

6.1.2.4 Konzepte in CIM^{2}BA/P

Die Abbildung 6.10 zeigt die Definition eines Konzeptes namens „concept“ in einem Kontext. Auch hier ist die Verwendung eines Kontextes erlaubt, um eine Umgebung zu definieren, in der genutzte Begriffe (Variablen, Terme, Relationen) geeignet interpretiert werden können.

Die Angabe einer Eigenschaft dieses Konzeptes erfolgt mittels eines Schlüsselwortes `FEATURE`, dem ein Variablenname folgt. Der über den Namen referenzierte Variable kann ein unscharfer Wert zugewiesen werden, der dem Sollwert entspricht. Durch das Schlüsselwort `SUBFEATURE` kann ein Subsegment (vgl. Abschnitt 5.2.3.3) beschrieben werden. Neben den Angaben, wie sie bei einer `FEATURE`-Beschreibung erfolgen, wird zusätzlich noch ein Lagebeschreibung mittels einer Variablen und einem Term festgelegt. Bspw. kann so die Angabe `SUBFEATURE Rundheit=Hirnstamm.Oben,hoch` sich auf den oberen Teil eines Segmentes beziehen. Relationen zu anderen Konzepten werden mittels des Schlüsselwortes `RELATION` definiert, wobei die Relation zu einem Konzept[2] angegeben wird, der anschließende Term gibt einen Erfüllungsgrad an, in dem diese Relation erfüllt ist. Mit den Schlüsselwörtern `PREVFUNCTION` und `POSTFUNCTION` können vor bzw. nach der Betrachtung des Konzeptes durch die Funktion `COMPARE` (vgl. Abschnitt 6.1.2.7) Funktionen aufgerufen werden. Dies kann zum Beispiel dazu genutzt werden, daß zu dem Konzept gehörende Aufgaben wie Vorsegmentierung o.ä. durchgeführt werden können. Eine Anwendung für eine Nachverarbeitung kann bspw.

[2]Dieses Konzept darf zur eindeutigen Berechnung der Relation nur eine einzige Instanz besitzen.

Definition: Konzept

```
DEFINE_CONCEPT [context.|*]concept [IS_A [context2.|*]concept2|*]
               [PRIO p|*][MUSTEXIST|*]

  FEATURE [f_context1.|*]fname_1 = fterm_1
          [MOD|*][PRIO (p1,t1)|*]
  FEATURE [f_context2.|*]fname_2 = fterm_2
          [MOD|*][PRIO (p2,t2)|*]
  :
  SUBFEATURE [f_context3.|*]sfname_3 = [sfcontext.|*]var.term,fterm_3
             [MOD|*][PRIO (p3,t3)|*]
  :
  RELATION rname=[r_context.|*]concept,term
           [MOD|*][PRIO (p3,t3)|*][OntroubleIgnore|*]
  :
  PREVFUNCTION prevfuncname_1
  PREVFUNCTION prevfuncname_2
  :
  POSTFUNCTION postfunctname_1
  POSTFUNCTION postfunctname_2
  :
  FOM = fom_context.a_fom
        [MOD|*][PRIO (p4,t4)|*]
DEFINE_CONCEPT_END
```

Abbildung 6.10: Definition eines Konzeptes „concept“ im Kontext „context“.

die Speicherung von Ergebnissen sein. Diese Funktionen stellen aber keine echten Dämonen dar, wie sie in Abschnitt 3.2.4.3 beschrieben werden, da sie nicht abhängig von internen Belegungen der Instanzen des Konzeptes sind. Schließlich gibt die letzte Anweisung `FOM` ein Fuzzy-Objekt-Modell für das Konzept an (vgl. Abschnitt 5.2.3.4). Hierbei handelt es sich um ein grafisches Modell des Konzeptes in unscharfer Form.

Der optionale Parameter `MOD` legt bei allen Anweisungen fest, daß bei einem Iterationsprozeß zur Modifikation der verwendeten Parameter die angegebenen Fuzzy-Mengen verändert werden dürfen, wobei die Änderung nur temporär während des Programmlaufs gültig ist. Zudem besteht die Möglichkeit, Prioritäten `PRIO` $(\mathtt{p_i}, \mathtt{t_i})$ für die Berechnung der Eigenschaftswerte festzulegen, wodurch eine Reihenfolge der Berechnung definiert wird. Mit Hilfe eines Abbruchkriteriums kann somit ggf. die Berechnung für die Zugehörigkeitswerte eines Segmentes S schon dann beendet werden, wenn aufgrund von Zugehörigkeitswerten, die unterhalb eines Schwellenwertes t_i liegen, entscheidbar ist, daß das Segment S nicht eine Instanz des betrachteten Konzeptes ist (vgl. Abschnitt 5.2.4). Bei Relationen ist als zusätzliche Option ein Parameter `OnTroubleIgnore` erlaubt, der bewirkt, daß die Relation nicht weiter betrachtet wird, falls das dort referenzierte Objekt unbekannt ist. Zudem kann zum gesamten Konzept[3] eine Priorität

[3]Wann es sinnvoll ist, für ein Konzept eine Priorität anzugeben, wird in Abschnitt 5.2.4 diskutiert.

angegeben werden, die eine Sortierung auf die Konzepte definiert. Konzepte mit hoher Priorität (1 entspricht der höchsten, Werte kleiner 1 entsprechen keiner Priorität) werden dabei zuerst betrachtet. Ein Beispiel für die Definition von Konzepten findet sich etwa in Abbildung B.18.

Die Verwaltung einer Wissensbasis erfolgt konzeptbasiert. Neben dem Namen der Wissensbasis, der zur Identifikation dient, muß an erster Stelle eine *Sprache* gewählt werden. Hierbei handelt es sich um eine Basis von Fuzzy-Mengen. Dadurch ist festgelegt, welche Variablen und Terme für die Konzepte erlaubt sind. Das entsprechende Auswahlfenster ist in Abbildung 6.11(a) dargestellt.

Im nächsten Schritt können die einzelnen Konzepte definiert werden. Neben den Namen und dem Kontext, in dem sie interpretiert werden, können hier auch Einschränkungen auf die Ansicht (sagittal, coronar, transversal) des beschriebenen Objektes gemacht werden. Die ggf. zwingende Existenz und eine Priorität wird ebenfalls an dieser Stelle definiert. Zudem können hier Funktionsnamen angegeben werden, die in der entsprechenden Reihenfolge ausgeführt werden, wenn für das Konzept die `COMPARE`-Funktion (vgl. Abschnitt 6.1.2.7) ausgeführt wird. Bedingung ist natürlich, daß im ausführenden Programm die genannten Funktionen auch existieren. Falls dies nicht der Fall ist, wird eine entsprechende Fehlermeldung generiert. Die Eingabemaske des Werkzeuges ist in Abbildung 6.11(b) zu sehen.

Anschließend kann zu jedem Konzept die Liste der zu betrachtenden Eigenschaften festgelegt werden. Dazu wird für ein Konzept die Eigenschaft als Variable aus der Sprache und dessen unscharfer Sollwert definiert. Auch die Priorität und der Schwellenwert für die Priorität werden hier bestimmt (vgl. Abbildung 6.11(c)).

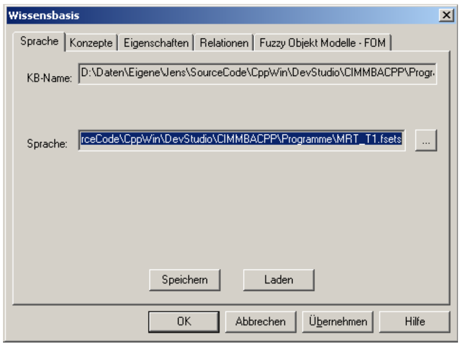
Die Definition der Relationen erfolgt in einem weiteren Fenster des Wissensbasis-Editors (vgl. Abbildung 6.11(d)). Dort wird zu einem Konzept ein weiteres Konzept ausgewählt, zu dem es in Relation steht. Welche Relationen möglich sind, ist wiederum von der Fuzzy-Mengen-Basis als Sprache abhängig.

Als letztes kann ein Fuzzy-Objekt-Modell zu einem Konzept gewählt werden. Dazu wird, wie in Abbildung 6.11(e) gezeigt, aus einer FOM-Wissensbasis ein FOM ausgewählt.

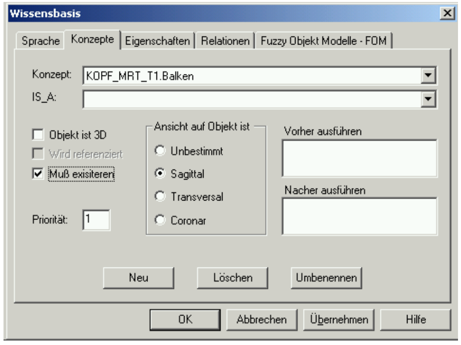
Der Wissensbasis-Editor erzeugt eine dem in Abbildung 6.10 beschriebenen Format konforme Wissensbasis, die in der Programmierumgebung (Abschnitt 6.2) verwendet werden kann. Eine konkrete Umsetzung und Anwendung einer Wissensbasis wird in Kapitel 7 vorgestellt und in Abbildung B.18 gezeigt.

6.1.2.5 Repräsentation von Handlungswissen

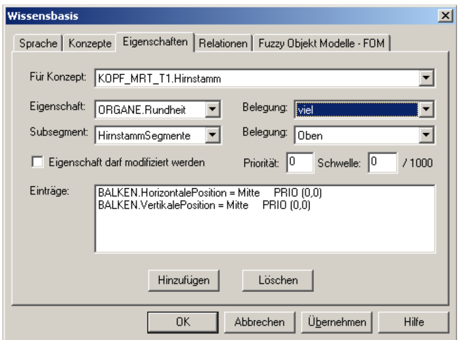
CIM^{2}BA/P gestattet die Verwendung von Regeln zur Formulierung von Handlungswissen in einer Form, wie sie in Abbildung 6.12 gezeigt und in Abschnitt 3.2.4.1 beschrieben wird. Die Prämisse kann dabei aus mehreren Teilbedingungen, die per fuzzy-logischem *und* (mit dem Schlüsselwort `AND`) verknüpft werden,



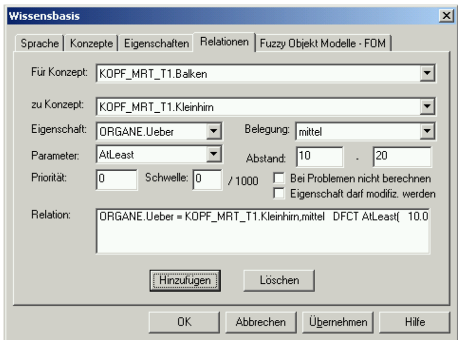
(a) Werkzeug zur Bearbeitung der Wissensbasis.



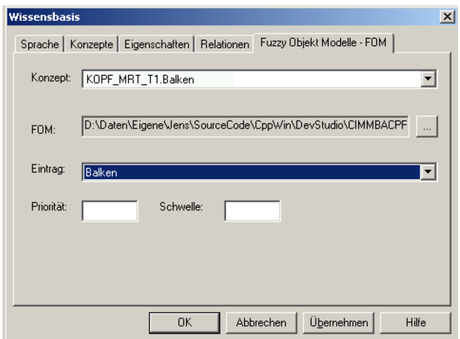
(b) Werkzeug zur Bearbeitung eines Konzeptes.



(c) Werkzeug zur Bearbeitung der Eigenschaften.



(d) Werkzeug zur Bearbeitung der Relationen.



(e) Werkzeug zur Bearbeitung eines FOM-Eintrages.

Abbildung 6.11: Werkzeuge zur Definition eines Konzeptes.

Definition: Regelbasis

```
DEFINE_RULES
  IF (premise_1) THEN conclusion_1
  IF (premise_2) THEN conclusion_2
  :
DEFINE_RULES_END
```

Abbildung 6.12: Definition von Regeln in CIM²BA/P.

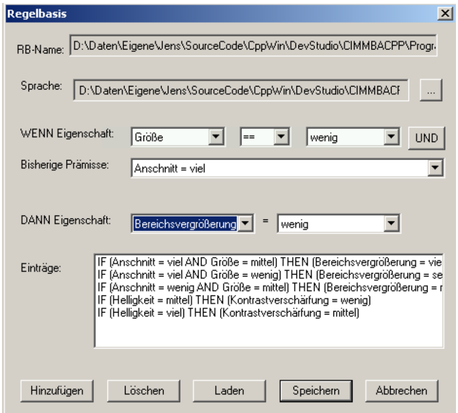


Abbildung 6.13: Werkzeug zur Bearbeitung einer Regelbasis.

bestehen, die Konklusion ist einstellig. Genutzt werden dürfen sowohl scharfe als auch unscharfe Variablen und Terme.

Die benötigten Regeln können mit dem in Abbildung 6.13 gezeigten Regelbasis-Werkzeug definiert und verwaltet werden. Die schon in Abschnitt 6.1.2.4 beschriebene Sprachauswahl erfolgt ebenfalls im Regelbasis-Editor. Eine Anwendung der Regeln erfolgt in der Programmierumgebung (vgl. Abschnitt 6.2).

6.1.2.6 Repräsentation von temporärem Wissen

Gefundene Segmente müssen in einem Analyseprozeß klassifiziert werden. Dazu sind sie geeignet zu repräsentieren. Das Wissen ist *temporär*, da die einzelnen Segmente nur während der Laufzeit einer Analyse bekannt sind.

Jedes Segment wird in einer Instanz eines speziellen Konzeptes *Segment* gespeichert (Abbildung 6.14). Neben den Segment-Daten (`SEGMENT_DATA`), welche die Bildinformationen des Segmentes (vgl. Abschnitt 2.3.3) und den Zugehörigkeitsgrad beinhalten (aufgrund der großen Anzahl möglicher Daten ist an dieser Stelle nur ein Verweis auf eine weitere, interne Segmentstruktur angegeben, was durch das `@` angedeutet werden soll.), besitzt das Konzept als Eigenschaftsbeschreibungen *alle* Informationen, die für ein Segment berechenbar sind. In den einzelnen Eigenschaften werden allerdings sowohl berechnete (scharfe) Werte ge-

Definition: Segment

```
DEFINE_CONCEPT_SEGMENT concept
  SEGMENT_DATA @data
  FEATURE fname_1 = fval1
  FEATURE fname_2 = fval2
  :
  RELATION rname_1(pc1,pt1) = rval1
  RELATION rname_2(pc2,pt2) = rval2
  :
DEFINE_CONCEPT_SEGMENT_END
```

Abbildung 6.14: Definition des Konzeptes „segment".

speichert (vgl. Abschnitt 5.2.3), wie auch die Zugehörigkeit zu jedem Term, der zu der in der Eigenschaft beschriebenen unscharfen Variablen gehört. Entsprechend sind die $\texttt{fval}_i$ (und $\texttt{rval}_j$) als $(n+1)$-Tupel aufzufassen, falls die Variable n Terme hat[4].

Das Konzept erlaubt zudem eine parametrisierte Abfrage von Relationen zu anderen Konzepten. Als Parameter dienen dabei das Konzept und ein Erfüllungsgrad. Schließlich weist das Konzept noch eine weitere Besonderheit auf. Es existiert eine spezielle parametrisierte Relation *Compare*, welche das Konzept mit einem weiteren Konzept vergleicht und einen Ähnlichkeitswert berechnet.

6.1.2.7 Vergleichsfunktion `Compare`

Ein Vergleich zwischen einem gefundenen Segment und einem in einem Konzept beschriebenen Objekt ist notwendig, um eine Klassifikation durchführen zu können. Deshalb wird im folgenden davon ausgegangen, daß für eine Segmentinstanz S, wie sie in Abschnitt 6.1.2.6 beschrieben ist, die Klassenzugehörigkeit[5] bestimmt werden soll. Zudem sei eine Wissensbasis ω wie in Abschnitt 6.1.2.4 gegeben, $e_{j,C}$ ist die j-te Eigenschaft[6] des Konzeptes C.

Ein Segment S wird mit einem in ω bekanntem Konzept C mittels der in Abschnitt 6.1.2.6 genannten Relation *Compare* verglichen. Entsprechend der in C verwendeten Eigenschaften $e_{j,C}$ werden diese (auf Anforderungen, wenn noch nicht geschehen) für das Segment S_i berechnet. Die Berechnung der Eigenschaften erfolgt in der Reihenfolge der angegebenen Prioritäten für die Eigenschaften (vgl. Abschnitt 6.1.2.4). Falls für eine Eigenschaft $e_{j,C}$ keine Priorität (bzw. kleiner Eins) angegeben wird, erhält diese automatische die (höchste) Priorität 1 und wird im ersten Durchlauf mit berechnet. Die Priorität ist eine natürliche

[4] Implementierungstechnisch erfolgt die eigentliche Berechnung allerdings aus Performanzgründen erst bei der ersten Abfrage einer Variablen.

[5] Ein Ähnlichkeitsmaß, das angibt, zu welchem Grad die gefundenen Eigenschaften mit den Eigenschaften im Konzept übereinstimmen.

[6] Wobei eine Ordnung an dieser Stelle für j unerheblich ist, es dient lediglich zur Differenzierung der Eigenschaften.

Zahl, die frei vom Wissensingenieur gewählt werden kann. Wird die zur Priorität definierten Schwelle unterschritten, wird eine weitere Berechnung abgebrochen und die Zugehörigkeit auf Null gesetzt. Falls alle Schwellenwerte einer Prioritätsstufe überschritten (bzw. erreicht) werden, wird mit der nächsten Prioritätsstufe fortgefahren, bis alle Eigenschaften berechnet sind oder eine schwellenbedingte Terminierung erfolgt.

Analog wird mit den Relationen im Konzept C verfahren. Dabei wird berechnet, mit welchem Grad die zum Segment S gehörenden Bildeigenschaften in Relation zu anderen gefundenen Objekten gilt. Voraussetzung hierfür ist, daß das referenzierte Objekt auch schon bekannt ist. Ist dies nicht der Fall, kann eine Klassifikation des aktuellen Segmentes zurückgestellt werden und mit der Klassifikation der noch nicht betrachteten Segmente fortgefahren werden. Ist die Liste der Segmente abgearbeitet, wird in einer zweiten Stufe versucht, die Relationen zu berechnen. Dieses Verfahren iteriert solange, bis kein Segment mehr klassifiziert werden kann. Alternativ kann auf die Berechnung der Relation verzichtet werden (falls der Parameter `OnTroubleIgnore` gesetzt ist). Die nach allen Iterationen noch unklassifizierten Segmente werden als *unbekannt* eingestuft. Umgekehrt kann es Konzepte in der Wissensbasis mit der Einstufung `MustExist` geben, denen noch kein Segment zugeordnet wurde. An dieser Stelle wird bei Bedarf durch Neuberechnung verwendeter Parameter (etwa Suchbereiche, Grauwertbereiche) oder Verwendung anderer (ggf. aufwendigerer) Methoden oder Operatoren eine weitere Iteration mit erneuerter Segmentierung gestartet. Entsprechende Parameteranpassungen sind anwendungsabhängig zu entscheiden.

Sofern alle (berechenbaren) Zugehörigkeitswerte der in C angegebenen Eigenschaften $e_{j,C}$ berechnet wurden, wird mittels eines T-Quantors TQ_τ ein Gesamtzugehörigkeitswert für die betrachtete Bildpunktmenge ermittelt. Entsprechende *Don't-Care*-Belegungen werden mit der Zugehörigkeit 1 (als neutrales Element, (vgl. Definition 3.11)) belegt. Nicht berechnete Relationen werden bei `OnTroubleIgnore` ebenfalls mit 1 berücksichtigt, ansonsten mit $\frac{1}{2}$ als maximale Unschärfe.

6.1.3 Neuronale Netze

6.1.3.1 Vorwärtsgerichte Netze

Vorwärtsgerichtete neuronale Netze können in CIM^{2}BA/P zur Klassifikation von einzelnen Segmenten genutzt werden. Dazu werden sie in einem Format gespeichert, wie in Abbildung 6.15 dargestellt ist.

Mit dem Schlüsselwort `SHORTCUTS` wird zunächst definiert, ob Verbindungen über eine Schicht hinweg erlaubt sind. Anschließend werden Definitions- und Wertebereich der Neuronen definiert und festgelegt, ob es sich um Ganzzahlwerte handelt (`yes`) oder nicht (`no`). Zudem wird der Wertebereich für die Gewichte definiert (Schlüsselwort `WEIGHTRANGE`). Dann wird mit dem Schlüssel-

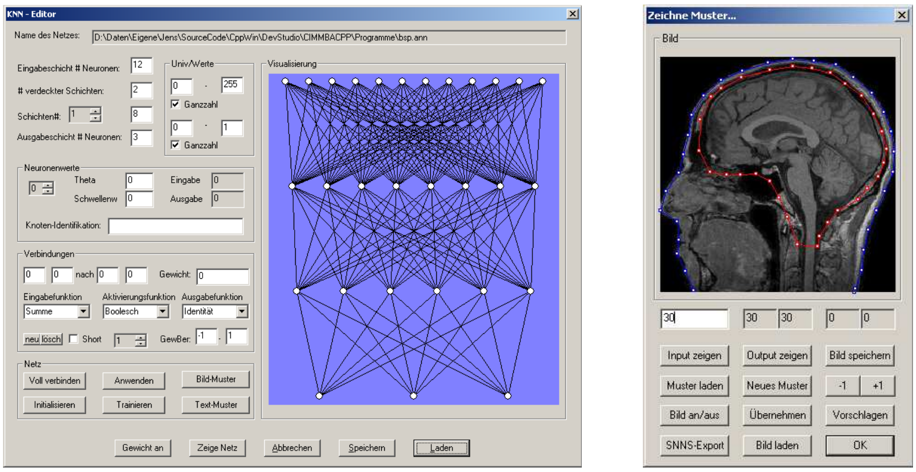
Definition: FF-Netz

```
DEFINE_FF_NET name
SHORTCUTS [yes|no]
UNITDOMAIN min max [yes|no]
UNITRANGE min max [yes|no]
WEIGHTRANGE min max [yes|no]

UNITS_IN_LAYER l1 m1
UNITS_IN_LAYER l2 m2
:
NEURON l1 n1 theta1 threshold1 [add|prod] [binary|linear|logistic|sigmoidal] [indentity] [ident1|*]
NEURON l2 n2 theta2 threshold2 [add|prod] [binary|linear|logistic|sigmoidal] [indentity] [ident2|*]
:
CONNECTION [srclayer1 srcneuron1|identx] destlayer1 destneuron1 weight1
CONNECTION [srclayer2 srcneuron2|identy] destlayer2 destneuron2 weight2
:
DEFINE_FF_NET_END
```

Abbildung 6.15: Definition eines Feed-forward-Netzes „name“ mit Angabe der Neuronen und Gewichte.

wort `UNITS_IN_LAYER l` die Anzahl der Neuronen in der Schicht l angegeben, wobei $l \in \mathbb{N}_0$ von der Eingabe- zur Ausgabeschicht indiziert ist. Das Schlüsselwort `NEURON l n` (das n-te Neuron in Schicht l) definiert die Eigenschaften des entsprechenden Neurons, indem der Bias Θ, ein Schwellenwert für das „Feuern“ des Neurons und die verwendeten Funktionen bestimmt werden. Auch kann eine optionale Zeichenkette als Bezeichner für das Neuron angegeben werden, über den es identifizierbar ist. Dadurch besteht die Möglichkeit, einzelnen Neuronen (hauptsächlich den Ein- und Ausgabeneuronen) eine natürlichsprachliche Semantik zu geben. Bspw. kann ein Ausgabeneuron mit dem Namen der Struktur benannt werden, der es bei der Klassifikation entspricht. Umgekehrt kann ein Eingabeneuron mit dem Namen des Eingabewertes (bspw. *Größe*) bezeichnet werden. Mittels dieses Bezeichners kann dann ebenfalls auf ein Neuron zugegriffen und der Eingabewert gesetzt bzw. der Ausgabewert gelesen werden. Schließlich wird mit dem Schlüsselwort `CONNECTION` eine Verbindung zwischen den definierten Neuronen mit dem entsprechenden Gewicht definiert. *Wie* ein solches Netz verwendet werden kann, wird in Abschnitt 6.2.3.6 beschrieben, *wofür* in Abschnitt 5.3.1.

Die Definition eines FF-Netzes erfolgt mit dem in Abbildung 6.16(a) dargestellten Werkzeug. Die Struktur des Netzes wird durch Eingabe der Anzahl der Schichten und der jeweiligen Anzahl der Neuronen in der Schicht definiert. Die Verbindungen der Neuronen untereinander erfolgt mit Hilfe des grafischen Eingabegerätes oder durch manuelle Eingabe. Die Eigenschaften der Neuronen sowie die Auswahl der Aktivierungsfunktion oder eines Schwellenwertes findet ebenfalls hier statt. Das Lernen erfolgt überwacht mittels einer Vorgabe der Eingabe- und Soll-Ausgabedaten und dem Standard-Backpropagation-Algorithmus. Die Über-



(a) Netzeingabe (b) Mustereingabe

Abbildung 6.16: Werkzeug zur Bearbeitung künstlicher neuronaler Netze und zur Eingabe von Trainingsmustern für Bilddaten.

nahme einer bekannten bzw. schon trainierten Netzstruktur (bspw. aus SNNS, [Zel94]) ist durch Eingabe der Netzgewichte zusätzlich möglich. Die Eingaben von Trainingsmustern können mit dem Werkzeug aus Abbildung 6.16(b) erfolgen.

6.1.3.2 Zellulare neuronale Netze

CIM^{2}BA/P erlaubt auch die Verwendung zellularer neuronaler Netze (vgl. Abschnitt 3.1.3.2), die aufgrund ihrer Struktur besonders für die Bildvorverarbeitung geeignet sind [HAMM98], da sie je Bildpunkt eine Zelle (Neuron) besitzen, welche mit genau seinen Nachbarn verbunden ist.

Die Verwendung zellularer neuronaler Netze nach Aizenberg [Aiz99] ist dabei in Grundfunktionen (zur Rauschreduktion und Kantenfilterung) implementiert. Die Eingabe einer Netzstruktur erfolgt, wie in Abbildung 6.17 zu sehen ist, durch die Festlegung der Größe der Nachbarschaft und durch Auswahl einer Funktion, die auf ein Bild angewendet werden soll. Die Nachbarschaft wird als rechteckiges Fenster betrachtet und ist in der Regel von der verwendeten Funktion abhängig.

Durch diese Grundimplementierung ((vgl. Abbildung 6.18) ist eine Schnittstelle geschaffen worden, die weiter ausgebaut werden kann, sofern weitere Funktionen mit Hilfe der CNN gelernt wurden oder aber eine Lernkomponente integriert ist. Eine Anwendung eines CNN ist in Abbildung 4.3(c) dargestellt.

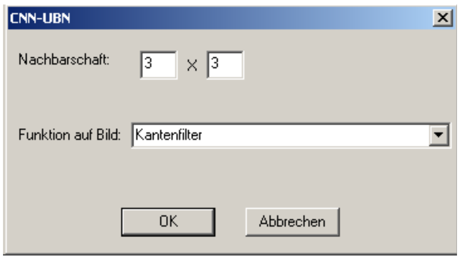


Abbildung 6.17: Werkzeug zur Bearbeitung zellularer neuronaler Netze.

Definition: CNN-Netz

```
DEFINE_CNN_NET name
  NEIGHBORHOOD y x
  TYPE [UBN|MVN]
  FUNCTION functionname
DEFINE_CNN_NET_END
```

Abbildung 6.18: Definition eines zellularen neuronalen Netzes „name“ mit Auswahl der zu berechnenenden Funktion.

6.1.4 Evolutionäre Algorithmen

6.1.4.1 Evolutionsstrategien

Für die Optimierung von Parametern können in CIM²BA/P evolutionäre Algorithmen eingesetzt werden, wobei auf die Verwendung von Soll-/Ist-Bildern zurückgegriffen wird. Die Definition notwendiger Einstellungen wird in Abbildung 6.19 gezeigt. Zunächst wird ein Programm gewählt, welches in der Programmiersprache CIM²BA/P geschrieben ist. Dieses Programm dient zur Berechnung des Zielbildes und verwendet dazu die zu optimierenden Parameter, die an dieser Stelle ebenfalls festgelegt werden.

Neben diesen Einstellungen für die Anzahl der Iterationen, also der Anzahl der zu erzeugenden Generationen, kann die Anzahl der Eltern für die Rekombination bzw. Selektion sowie die Anzahl der Nachkommen festgelegt werden. Wie die Nachfolgegeneration zustande kommt, kann mit dem Strategieparameter bestimmt werden. Für eine frühzeitige Beendigung (das heißt, daß nicht die maximale Anzahl vorgegebener Generationen berechnet wird) ist ein Terminierungskriterium mit einem Schwellenwert anzugeben. Das Eingabefenster für die Optimierungsparameter zeigt Abbildung 6.20. In dem zu optimierenden CIM²BA/P-Programm müssen geeignete Schlüsselwörter verwendet werden. Diese ersetzen die Ladebefehle für die Bilddaten und definieren die zu optimierenden Parameter. Außerdem muß ein Ergebnisbild definiert werden, mit dem das jeweilige Zielbild aus Abbildung 6.20 verglichen werden kann. Die zu vergleichenden Bilder

Definition: ES-Optimierungsparameter

```
DEFINE_OPTIMIZER name
  CIMMBAPROGRAMME pname
  FSETBASE fname
  SAVEPATH spname
  ISADMIN [no|yes]
  FITNESSFCT [difference1|difference2|quotient]
  COMPAREIMAGE sourceimage1 destimage1
  COMPAREIMAGE sourceimage2 destimage2
  :
  RECOMBINEFCT [mean|swap]
  PARENTS n
  MUTATIONFCT [fix|decrease|adaptive] init
  SELECTIONSTRATEGY [comma|plus]
  MU m
  LAMBDA l
  GENERATIONS g
  TERMINATION t
  FSCHECK UNIV=[no|yes] FSETS=[no|yes] NEIGHB=[no|yes] MODIFY=[no|yes]
DEFINE_OPTIMIZER_END
```

Abbildung 6.19: Definition der Rahmendaten für eine Parameteroptimierung mit Hilfe von Evolutionsstrategien.

werden jeweils mit dem Schlüsselwort `COMPAREIMAGE` festgelegt, wobei hier die Dateinamen des Eingabe- und des Zielbildes anzugeben sind.

Bei der Rekombination kann zwischen den Funktionen *Durchschnitt* und *Austasch der Parameter* gewählt werden, die auf jede Komponente $a_1, \ldots, a_4$ einer Fuzzy-Menge angewendet werden kann, wobei die dazu notwendige Auswahl der beteiligten Eltern *gleichverteilt* durchgeführt wird. Die Mutation erlaubt eine *fixe* Schrittweite oder *sinkende* oder *adaptive* Veränderung, wobei der Startwert (Standardabweichung der verwendeten Normalverteilung mit Erwartungswert Null) initialisiert wird und sich auf die Breite des Universums bezieht. Abbildung 6.21 zeigt Beispiele für die Rekombination zweier Fuzzy-Mengen.

Schließlich können noch einige Restriktionen für die Erzeugung der Fuzzy-Mengen angegeben werden, wobei die Verwendung mit `no` (nicht prüfen) oder `yes` (prüfen) kodiert wird. Bei einer Verletzung der Restriktionen wird eine Korrektur der Fuzzy-Mengen derart durchgeführt, daß die geforderten Bedingungen wieder erfüllt sind, wobei nach jeder Anpassung eine erneute Prüfung erfolgt. Zu beachten ist allerdings, daß eine festgelegte Schrittweite durch die Korrektur wieder übergangen werden kann. Folgende Anforderungen an die Fuzzy-Mengen stehen zur Auswahl, wobei sich die Notation $\mathcal{F}[a_i]$ auf den Parameter a_i der Menge $\mathcal{F}$ gemäß Definition 6.1 bezieht:

- *Universum überdecken*
  Wird diese Option gewählt, wird bei der Mutation (oder Rekombination) der Fuzzy-Mengen geprüft, ob das gesamte Universum überdeckt wird. Ist dies für eine Fuzzy-Variable $\mathcal{F}_\diamond$ nicht der Fall, so werden die jeweils am

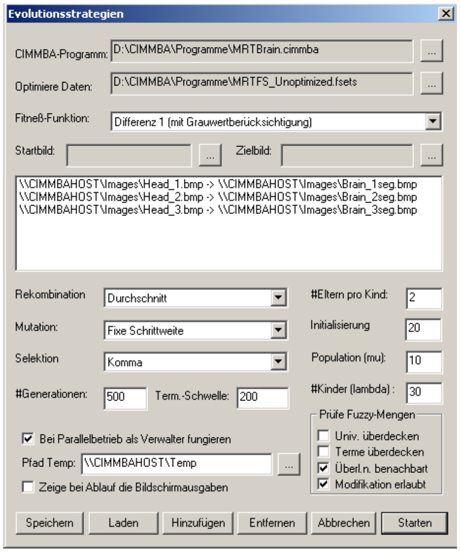


Abbildung 6.20: Werkzeug zur Verwendung von Evolutionsstrategien.

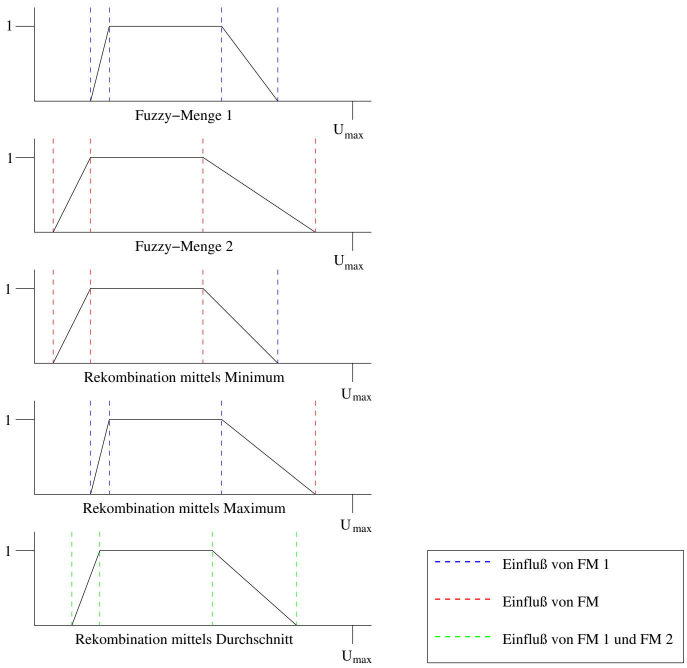


Abbildung 6.21: Rekombination zweier Fuzzy-Mengen mit unterschiedlichen Rekombinationsfunktionen.

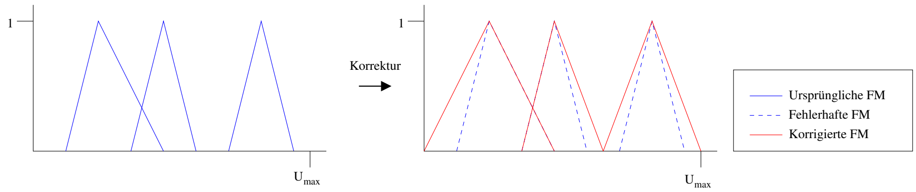
weitesten außen liegenden Mengen $\mathcal{F}_{links}$ und $\mathcal{F}_{rechts}$ so modifiziert, daß das gesamte Universum $\mathbb{U}_{\mathcal{F}_\diamond}$ nach rechts bzw. links überdeckt wird. Dazu wird der Parameter a_1 (vgl. Definition 6.1) auf $\min(\mathbb{U}_{\mathcal{F}_\diamond})$ gesetzt, falls $a_1 > \min(\mathbb{U}_{\mathcal{F}_\diamond})$ gilt. Entsprechendes gilt für a_4 und $a_4 < \max(\mathbb{U}_{\mathcal{F}_\diamond})$. Liegen Überdeckungslücken *innerhalb* des Bereiches von $\mathbb{U}_{\mathcal{F}_\diamond}$ vor, werden die jeweils links und rechts davon liegenden Fuzzy-Mengen so angepaßt, daß der Parameter a_4 der links von der Lücke liegenden Fuzzy-Menge $\mathcal{F}_i$ und der Parameter a_1 der rechts von der Lücke liegenden Fuzzy-Menge $\mathcal{F}_{i+1}$ auf die Mitte der Lücke gesetzt werden. Dies bedeutet, daß für $\mathcal{F}_i[a_1] < \mathcal{F}_{i+1}[a_1]$ folgende Modifikation vorgenommen wird: $\mathcal{F}_{i+1}[a_1] \stackrel{\text{def}}{=} \mathcal{F}_i[a_4] \stackrel{\text{def}}{=} \frac{\mathcal{F}_i[a_4]+\mathcal{F}_{i+1}[a_1]}{2}$. Ein Beispiel ist in Abbildung 6.22(a) gezeigt.

- *Terme überdecken*
  Mit dieser Option wird gewährleistet, daß der Träger einer Fuzzy-Menge $\mathcal{F}_i$ nicht vom Träger einer $\mathcal{F}_j,\ i \neq j$ überdeckt wird. Es wird also sichergestellt, daß $support(\mathcal{F}_i) \not\subset support(\mathcal{F}_j)$. Falls $support(\mathcal{F}_i) \subset support(\mathcal{F}_j)$ gilt, wird folgende Modifikation vorgenommen: Falls der Kern (bzw. der Bereich der Maximalwerte bei nicht normalen Fuzzy-Mengen) der überlappten Fuzzy-Menge $\mathcal{F}_i$ links vom Kern (bzw. wiederum des Maximalbereichs) der überlappenden Fuzzy-Menge $\mathcal{F}_j$ liegt (also $\mathcal{F}_i[a_3] < \mathcal{F}_j[a_2]$), dann erfolgt die Anpassung $\mathcal{F}_i[a_1] \stackrel{\text{def}}{=} \mathcal{F}_j[a_1]$, liegt der Kern (bzw. Maximalbereich) rechts (also $\mathcal{F}_i[a_2] > \mathcal{F}_j[a_3]$), erfolgt $\mathcal{F}_i[a_4] \stackrel{\text{def}}{=} \mathcal{F}_j[a_4]$. Abbildung 6.22(b) zeigt ein Beispiel für den ersten Fall. Die Gleichheit der Fuzzy-Mengen bzw. die Überlappung der Kerne wird gesondert behandelt (s.u.).

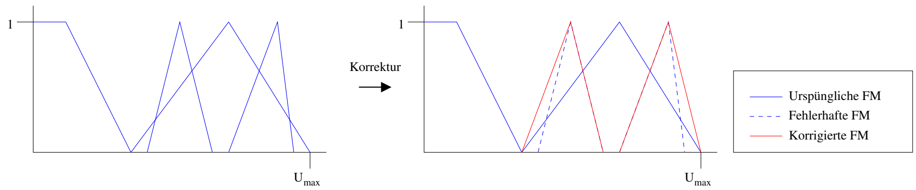
- *Überlappung nicht benachbarter Fuzzy-Mengen*
  Diese Option verhindert, daß sich zwei nicht direkte benachbarte Fuzzy-Mengen $\mathcal{F}_i$ und $\mathcal{F}_{i+j},\ j > 1$ nicht überlappen. Es muß also $\mathcal{F}_i[a_4] < \mathcal{F}_{i+j}[a_1],\ j > 1$ gelten. Gilt dies nicht, wird folgende Modifikation durchgeführt: $\mathcal{F}_{i+j}[a_1] \stackrel{\text{def}}{=} \mathcal{F}_i[a_4] \stackrel{\text{def}}{=} \frac{\mathcal{F}_i[a_4]+\mathcal{F}_{i+j}[a_1]}{2},\ j > 1$. In Abbildung 6.22(c) wird eine Beispielanpassung gezeigt.

Die angegebenen Modifikationen werden nur dann durchgeführt, wenn der entsprechende Schalter `MODIFY` (bzw. in der Eingabemaske der Punkt *Modifikation erlaubt*) gesetzt ist, ansonsten wird die für die Fuzzy-Menge berechnete Mutation nicht durchgeführt und der ursprüngliche Inhalt der Parameter beibehalten.

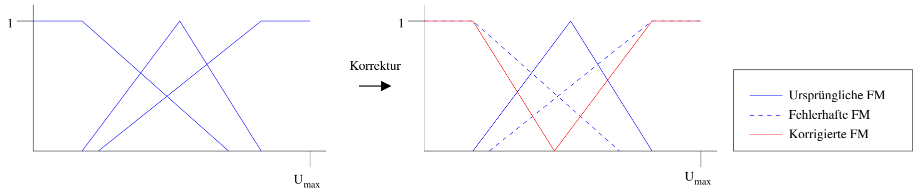
Implizit wird überprüft, daß Kerne (bzw. die Bereiche der Maximalwerte bei nicht normalen Fuzzy-Mengen) sich nicht überdecken dürfen. Ausnahme ist ein einziger gemeinsamer Punkt. Falls eine Überdeckung vorliegt, wird diese Veränderung (Rekombination oder Mutation) nicht durchgeführt. Auch wird sichergestellt, daß die Fuzzy-Mengen nicht über dem Universum herausragen (durch eine eventuelle Neudefinition der Fuzzy-Mengen-Parameter $a_1, \ldots, a_4$ auf die Universumsgrenzen). Zudem wird die Ordnung auf den Fuzzy-Mengen beibehalten, so



(a) Korrektur Universumsüberdeckung



(b) Korrektur Termüberdeckung



(c) Korrektur Nicht-Nachbarschafts-Überdeckung

Abbildung 6.22: Optionale Korrekturen von Fuzzy-Mengen bei der Anwendung von evolutionären Algorithmen. Gezeigt werden die ursprünglichen Fuzzy-Mengen (blau) und die korrigierten Fuzzy-Mengen (rot). Die gestrichelte Darstellung zeigt die fehlerhaften Fuzzy-Mengen vor der Korrektur.

daß nicht eine Fuzzy-Menge $\mathcal{F}_i$ rechts von einer Fuzzy-Menge $\mathcal{F}_{i+j},\ j > 0$ positioniert sein darf. Weiterhin muß die Bedingung $a_1 \leq a_2 \leq a_3 \leq a_4$ gelten.

Parallelisierung evolutionärer Algorithmen

Evolutionäre Algorithmen eignen sich hervorragend für eine parallele Anwendung. Aus diesem Grund verfügt CIM²BA/P über die Möglichkeit, die Optimierung der Fuzzy-Mengen parallel auf mehreren Rechnern durchführen zu lassen. Dazu dient das Schlüsselwort `SAVEPATH` (bzw. das Feld *Pfad Temp* in der Eingabemaske). Dieses erhält als Parameter einen Pfad für die gemeinsame Datenhaltung zum Speichern der (Zwischen-)Ergebnisse. Anhand eines Eintrages an dieser Stelle erkennt ein Rechner, ob parallel mehrere Computer beteiligt sind. Bei der parallelen Anwendung muß *genau ein* Rechner als *Verwalter* fungieren (Schlüsselwort `ISADMIN`), der die *Selektion* der Individuen vornimmt. Entsprechend seiner

Einstellungen berechnet jeder Rechner die ihm vorgebenen Daten und Bildmaterialien. Nachdem er die lokale Selektion durchgeführt hat, werden die $\hat{\mu}$ besten Individuen im angebenen Pfad gespeichert und der Rechner solange in Wartestellung geschaltet, bis der Verwalter alle Individuen betrachtet und daraus wieder $\hat{\mu}$ beste Individuen selektiert hat. Nachdem diese ermittelt wurden, informiert der Verwalter die anderen Rechner darüber, ob das Terminierungskriterium erreicht ist oder ob eine weitere Generation berechnet werden muß. Für den letzten Fall können die Rechner die besten Individuen einlesen und auf diesem eine neue Iteration starten. Der Informationsaustausch erfolgt über Dateien in einem gemeinsamen Speicher. Jeder Rechner i meldet sich beim Verwalter an. Daran kann der Verwalter erkennen, auf wieviele Rechner gewartet werden muß, bis die Selektion erfolgen kann. Funktionen wie das Sperren von Dateien wird derzeit nicht unterstützt, so daß sichergestellt sein muß, das zwei Rechner nicht gleichzeitig gestartet werden oder aber differenzierbar sind (unterschiedliche Rechnernamen). Abbildung 6.23 zeigt schematisch die Realisierung der parallelen Verwendung der Optimierung.

Die Verteilung der Berechnungsaufgaben kann sowohl durch Verteilung der zu berechnenden Bildmaterialien als auch durch Verteilung der einzelnen Individuen einer Population erfolgen. Bei der Verteilung der Bildmaterialien ist darauf zu achten, daß alle beteiligten Rechner eine gleiche Variabilität von Datensätzen bearbeitet, da ansonsten die Fuzzy-Mengen ggf. auf einen speziellen Typus von Datensatz optimiert werden (etwa nur auf dunkle Bilder). Auf allen beteiligten Rechnern muß die Elternanzahl $\hat{\mu}$ identisch sein. Die Anzahl der betrachteten Individiduen einer Generation λ ergibt sich aus der Summe aller λ_i, wenn auf dem Rechner i λ_i Individuen berechnet werden.

Im Gegensatz zum *Migrationsmodell* ist die vorliegende Parallelisierung synchronisiert. Vom *Insel-Modell* unterscheidet es sich dadurch, daß regelmäßig kommuniziert und Informationen ausgetauscht werden. Insofern handelt es sich um ein globales Modell [Poh00], welches nur parallel eine Population berechnet. Allerdings ist auch der Austausch zwischen den einzelnen Strängen erst nach mehr als einer Generation möglich, so daß sich dies wieder mehr dem Migrationsmodell annähert. Dadurch besteht die Möglichkeit, verschiedene Modelle einzusetzen. Für die Parallelisierung von evolutionären Algorithmen bestehen vielfältige Methoden. In [CG99] stellen Cantú-Paz und Goldberg ein eigenes Modell vor und diskutieren den Geschwindigkeitsvorteil des parallelen Ansatzes. Dort wird aber im Gegensatz zu dem in der vorliegenden Arbeit gewählten Ansatz auf $\mathcal{P}$ Prozessoren zurückgegriffen, die untereinander kommunizieren ($\mathcal{P} \cdot (\mathcal{P} - 1)$ Kommunikationen) und somit einen deutlich höheren Kommunikationsaufwand benötigen als der Ansatz eines gemeinsamen Speichers, auf den beteiligten Rechner zugreifen können ($2 \cdot \mathcal{P}$ Kommunikationen). Ein weiterer Grund ist die sehr einfache Implementierung, die kaum Erweiterungen zum sequentiellen Ansatz erfordert.

Die Bilddaten sind – in Anhabhängigkeit von ihrer Größe – ggf. auf den lokalen Rechnern zu halten. Ist dies erfüllt, ist der Kommunikationsaufwand relativ zur

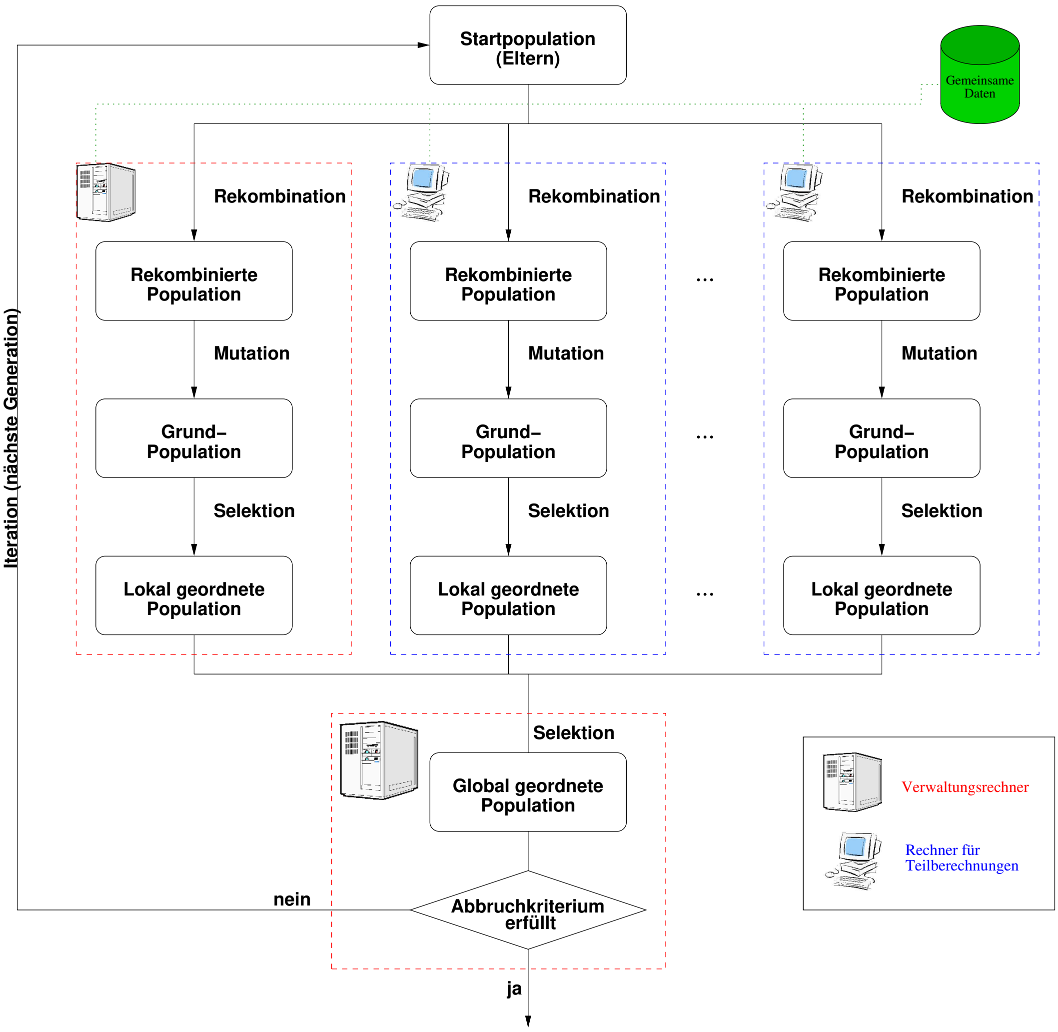


Abbildung 6.23: Parallele Realisierung der Optimierungskomponente. Der Verwaltungsrechner ist ebenfalls in den Optimierungsprozeß mit eingebunden.

Definition: GA-Optimierungsparameter

```
DEFINE_OPTIMIZER name
  CIMMBAPROGRAMME pname
  FSETBASE fname
  SAVEPATH spname
  ISADMIN [no|yes]
  FITNESSFCT [difference1|difference2|quotient]
  COMPAREIMAGE sourceimage1 destimage1
  COMPAREIMAGE sourceimage2 destimage2
  :

  CODING [gray|standard]
  RECOMBINEFCT [1pcrossover|2pcrossover|rand]
  PARENTS p
  CHILDS  c
  MUTATIONFCT [flip|set] init
  SELECTIONSTRATEGY [replace|elite]
  MU m
  LAMBDA l
  GENERATIONS g
  TERMINATION t
  FSCHECK UNIV=[no|yes] FSETS=[no|yes] NEIGHB=[no|yes] MODIFY=[no|yes]
DEFINE_OPTIMIZER_END
```

Abbildung 6.24: Definition der Rahmendaten für eine Parameteroptimierung mit Hilfe von genetischen Algorithmen.

Ausführung eines CIM^{2}BA/P-Programms als gering einzuschätzen ($\ll 1\%$), da lediglich die Individuen mit ihrer Fitneß zentral gelesen und geschrieben werden müssen.

6.1.4.2 Genetische Algorithmen

Die Nutzung von genetischen Algorithmen erfolgt in gleicher Weise wie bei den Evolutionsstrategien, es sind lediglich einige andere Einstellungmöglichkeiten bezüglich der verwendeten Strategien zu wählen, wie in Abbildung 6.25 zu sehen ist. Zudem ist hier die Art der binären Kodierung zu wählen, wobei zwischen dem Gray-Code und dem einfachen Binär-Code entschieden werden kann. Zu den Unterschieden wird hier auf Abschnitt 3.1.4.2 und auf [SHF94] verwiesen. Abbildung 6.24 zeigt das Speicherformat für eine Optimierung mittels genetischer Algorithmen, Abbildung 6.25 die Eingabemaske für die Parameter. Für die Kodierung wird die rechnerinterne Darstellung der Variablen als Basis verwendet. Die Auswahl der Eltern erfolgt nach dem *Roulette-Prinzip* [SHF94, Poh00], die Selektion erfolgt aufgrund der verwendeten Roulette-Methode nur an globaler Stelle, nicht auf den lokalen Rechnern (sofern das globale Modell verwendet wird).

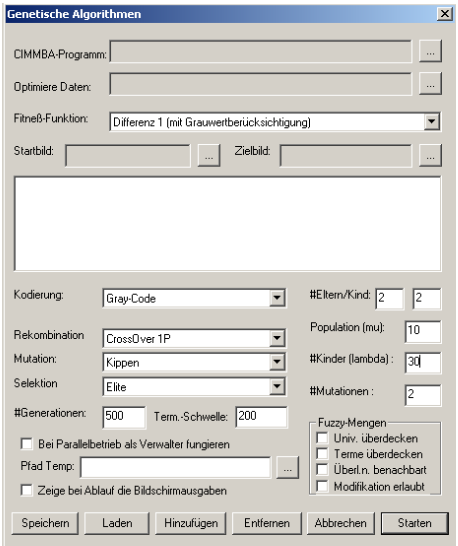


Abbildung 6.25: Werkzeug zur Verwendung genetischer Algorithmen.

6.2 Programmierumgebung

Die Hauptkomponente von CIM²BA/P stellt sicherlich das Werkzeug der Programmierumgebung dar (vgl. Abbildung 6.26). In ihr können Programme zur Analyse von Bilddaten definiert werden, die die zuvor beschriebenen Werkzeuge bzw. die damit generierten Wissensinhalte nutzen können. Dazu stehen spezielle Zugriffsfunktionen auf die Inhalte zur Verfügung. Um das zu realisieren, sind spezielle Datentypen definiert worden.

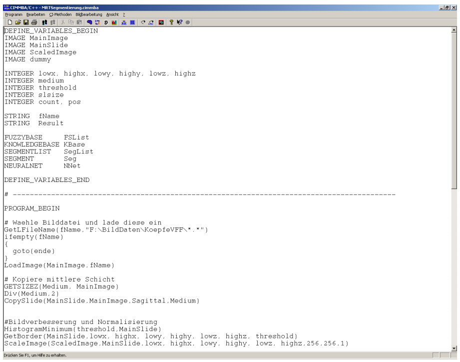
6.2.1 Struktur eines CIM²BA/P-Programmes

Ein CIM²BA/P-Programm besteht im wesentlichen aus zwei Teilen, der Definition von Variablen und dem eigentlichen Programmteil. Diese Trennung erfolgt aus Gründen der Übersichtlichkeit und leichten Lesbarkeit des Programms. Nachdem ein Programm gestartet ist, werden intern nicht relevante Teile wie bspw. Kommentare gelöscht, anschließend erfolgt die interpretierte Abarbeitung des Programms. Es sind bedingte Ausführungen oder Wiederholungen ebenso möglich wie die Verwendung von Sprungbefehlen oder Funktionsaufrufen. Grundsätzlich wird zwischen Groß- und Kleinschreibung im Programm nicht unterschieden.

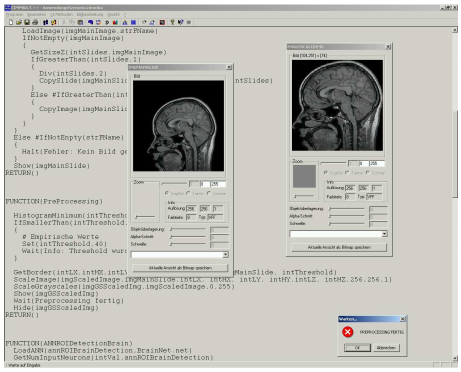
6.2.2 Datentypen

Als Standard-Datentypen erlaubt CIM²BA/P u.a. die Verwendung von

- Ganzzahlen (Schlüsselwort `INTEGER`)



(a) Programmeingabe



(b) Programmlauf

Abbildung 6.26: Die Programmierumgebung in CIM²BA/P mit der Darstellung eines Beispielprogramms.

- Realzahlen (Schlüsselwort `REAL`)
- Booleschen Variablen (Schlüsselwort `BOOLEAN`)
- Zeichenketten (Schlüsselwort `STRING`)

Spezielle Datentypen erlauben den Zugriff oder die Verwendung auf die mit den oben vorgestellen Werkzeugen erstellten Daten:

- Eine Fuzzy-Mengen-Basis (Schlüsselwort `FUZZYBASE`) ist eine Menge von unscharfen Variablen. Diese sind Fuzzy-Variablen (Schlüsselwort `FUZZYVAR`) und beinhalten die schon genannten Informationen wie Universum, Zahlenbereich und die einzelnen Terme des Typs Fuzzy-Menge (Schlüsselwort `FUZZYSET`) (vgl. Abbildung 6.6).
- Eine Wissensbasis (Schlüsselwort `KNOWLEDGEBASE`) beinhaltet eine Menge von Variablen des Typs `CONCEPT`.
- Ein vorwärtsgerichtetes neuronales Netz kann mit dem Schlüsselwort `NEURALNET` als Variable definiert werden. Die Anwendung erfolgt mit speziellen Zugriffsfunktionen für Netze. Ein zellulares neuronales Netz wird mit dem Schlüsselwort `CNN` als Variable definiert. Die Anwendung erfolgt ebenfalls mit speziellen Zugriffsfunktionen für Netze.
- Eine Segmentliste (Schlüsselwort `SEGMENTLIST`) beinhaltet eine Menge von Variablen des Typs `SEGMENT`. Operationen können auf jedes Element der Segmentliste ausgeführt werden.

- Das Schlüsselwort `IMAGE` beschreibt ein Bild. Mittels spezieller Zugriffsfunktionen kann auf den Bildinhalt oder auf Informationen über das Bild zugegriffen werden.

Diese und weitere Definitionen von Datentypen (vgl. Anhang C) werden zwischen `DEFINE_VARIABLES_BEGIN` und `DEFINE_VARIABLES_END` geschachtelt. Eine Variablendefinition erfolgt immer in der Form `TYP va, vb,...`. Die Namen der Variablen müssen eindeutig gewählt werden. Nicht mehr benötigte Variablen *var* können mit dem Befehl `DELETE(var)` gelöscht werden.

6.2.3 Befehlsbeschreibung

An dieser Stelle wird keine vollständige Beschreibung aller Befehle erfolgen, da es sich hier nicht um ein Benutzerhandbuch handelt (eine Liste der Befehle findet sich in Anhang C). Die Befehlsmenge wird zwischen dem Schlüsselwort `PROGRAM_BEGIN` und dem Schlüsselwort `PROGRAM_END` geschachtelt. Es wird der grundsätzliche Aufbau eines Befehls vorgestellt, zudem einige Befehle, etwa die Verwendung von Standard-Bildoperatoren, von Fuzzy-Mengen oder die Nutzung von neuronalen Netzen, um deren Anwendung zu verdeutlichen. Die Realisierung eines ganzen Programms wird in Kapitel 7 beschrieben. Grundsätzlich sind die Befehle in Präfixnotation und prozedural aufgebaut. Dies bedeutet, daß der Befehl und dann geklammert eine Liste der benötigten Parameter folgt. Bspw. bewirkt der Befehl `LOADIMAGE(head,name)` das Laden eines Bilddatensatzes mit dem Namen *Name*, welches sowohl ein Dateiname als auch eine Variable vom Typ `STRING` sein kann, in die Variable *Head* ein. Ergebnisse eines Prozeduraufrufs werden immer in die am weitesten links stehenden Parameter geschrieben, die dann Variablen sein müssen.

6.2.3.1 Generelle Informationen

Um das Lesen bzw. Beschreiben eines Programmes zu erleichtern, können Kommentare eingefügt werden, diese beginnen mit `#`. Bedingte Anweisungen sind in geschweiften Klammern {} gefaßt. Mit Hilfe des Prozentzeichens können Variablen (deren Wert) in Zeichenketten aufgelöst werden, etwa wird aus `Vertikale Größe=%sy Bildpunkte` und dem Wert 256 in *sy* die Zeichenkette `Vertikale Größe=256 Bildpunkte`.

6.2.3.2 Beispiele zu Standarddatentypen

Das Setzen von Datentypen auf einen Wert erfolgt mit dem Befehl `SET(var,val)`, wobei die Variable *var* auf den Wert *val* gesetzt wird. Ist *val* ebenfalls eine Variable, wird ihr Wert der Variablen *var* zugewiesen. Es existieren Konstanten wie etwa `SAGITTAL`, `TRANSVERSAL` oder `CORONAR`, `TRUE` oder `FALSE`, die direkt benutzt oder abgefragt werden können und auch eine feste Semantik haben. Die Darstellung von Variableninhalten (dazu zählen auch Bilder) können mit dem Befehl `SHOW(var)` erreicht werden.

6.2.3.3 Beispiele für Bildbefehle

Der oben schon vorgestellte Befehl `LOADIMAGE(img,name)` stellt einen wichtigen Befehl für ein Bild dar. Anhand des Suffix im Namen wird der Typ erkannt und die Datenmenge interpretiert, wobei bei einem RAW-Format von einem quadratischen Bildformat ausgegangen wird und die Dateigröße dann über das Bildformat entscheidet. Dabei wird als Heuristik[7] angenommen, daß die Bilder sagittale 8-Bit Graustufenbilder sind, die eine Auflösung von M^2 haben und eine Schichtanzahl von Z $\approx \frac{M}{2}$ Schichten besitzen. Wenn der Bilddatensatz aus n Bytes besteht, so wird zunächst ein M' aus $n \approx \frac{(M')^3}{2}$ berechnet und auf die nächste Zweiterpotenz aufgerundet, die als das gesuchte M interpretiert wird. Anschließend wird dann die Schichtanzahl Z aus dem Ergebnis bestimmt. Die Berechnung wird in Gleichung (6.11) dargestellt.

$$\begin{aligned} n &\approx (M')^2 \cdot \frac{M'}{2} \\ M' &\approx \sqrt[3]{2 \cdot n} \\ M &= 2^{\lceil \log_2 M' \rceil} \\ Z &= \frac{n}{M^2} \end{aligned} \tag{6.11}$$

Als Pendant existiert der Befehl `SAVEIMAGE(name,img)`, welcher ein Bild im aktuellen oder im optional angegebenen Format speichert. Es existiert eine Reihe von Standardoperationen auf Bilddaten, etwa die Anwendung von Filtern `SOBELH(imga,imgb)`. Hier wird der horizontale Sobeloperator auf das Bild *imgb* angewendet und das Ergebnis in *imga* gespeichert (vgl. Abbildung 2.8).

6.2.3.4 Bedingte Ausführung

Zur wiederholten Ausführung von Programmteilen kann auf eine Schleifenfunktion `FOR(var,za,zb,zc)` zurückgegriffen werden, wobei der folgende in {} eingeschlossene Block solange ausgeführt wird, bis die mit *za* initialisierte Variable *var* den Wert *zb* erreicht hat, wobei bei jeder Iteration die Variable um den Wert *zc* inkrementiert wird. Änderungen der beteiligten Parameter sind im Block möglich, sofern es sich um Variablen handelt, bspw. kann *za* auf *zb* gesetzt werden, um eine weitere Iteration zu verhindern. Vergleiche zwischen Variablen sind mit Befehlen der Art `IFEQUAL(vara,varb){BLOCKa} ELSE {BLOCKb}` möglich, wobei der Programmteil *BLOCKa* ausgeführt wird, wenn die Bedingung erfüllt ist, ansonsten

[7] Diese Heuristik ergibt sich aus den in der Arbeit behandelten Bilddatensätzen.

der *BLOCKb*. Sprünge sind mit dem Befehl `GOTO(lbl)` zu einer mit dem Befehl `LABEL(lbl)` gesetzten Zieladresse möglich.

6.2.3.5 Funktionen

Um eine weitere Strukturierungsmöglichkeit zu erhalten, sind Funktionsdefinionen im Programmanweisungsblock erlaubt. Funktionen werden mit dem Schlüsselwort `FUNCTION`, gefolgt vom Funktionsnamen, definiert und mit `RETURN` beendet. Als Rückgabewert wird der Inhalt einer implizit definierten Variablen namens *ReturnVal* verwendet. Diese Variable muß explizit vor den Rücksprung gesetzt werden und kann von jedem beliebigen (bekannten) Typ sein. Explizite Übergabeparameter gibt es nicht. Der Aufruf einer Funktion erfolgt mit `CALL(Funktionsname)`. Der Unterschied zu einem Sprungbefehl (vgl. Abschnitt 6.2.3.4) liegt in der Möglichkeit der übersichtlicheren Strukturierung des Programmtextes und der Existenz eines Rückgabewertes. Verschachtelungen von Funktionen sind nicht erlaubt.

6.2.3.6 Beispiele für CI-Befehle

Die wichtigste Komponente im Vergleich zu anderen Programmiersystemen ist die umfassende Einbindung von CI-Methoden. Demzufolge wird deren Verwendung im folgenden beschrieben.

Befehle für die Verwendung von unscharfen Methoden
Für den Zugriff auf Fuzzy-Mengen gibt es verschiedene Befehle. Das Setzen einer Fuzzy-Menge erfolgt mit dem Befehl `SET(var,params)`, wobei die Liste der Parameter mit Bezug auf Festlegung 6.1 variabel sein kann. Weiterhin existieren Befehle, um einzelne Parameter der Fuzzy-Menge zu ändern, bspw. definiert `SETCORE(var,left,right)` den Kern der Fuzzy-Menge *var* neu. Analog existieren Befehle, um einzelne Werte der Fuzzy-Menge auszulesen, z. B. liest `GETCORE(left,right,var)` den Kern der Fuzzy-Menge *var* in die Variablen *left* und *right*. Der Zugehörigkeitswert einer Fuzzy-Menge *var* an der Stelle x kann mit dem Befehl `GETDEGREE(d,var,x)` in die Variable d geschrieben werden.

Befehle für die Verwendung von neuronalen Netzen
Die Anwendung der neuronalen Netze erfolgt in der Art, daß den Neuronen Werte zugewiesen werden können bzw. deren Ausgabe abgefragt werden kann. Ersteres erfolgt durch den Befehl `SET(net,l,n,val)`. Dieser weist dem n-ten Neuron in Schicht l des Netzes *net* den Wert *val* zu. Mit `GET(val,net,l,n)` wird der Ausgabewert des entsprechenden Neurons in die Variable *val* geschrieben. Zur Modifikation des Netzes kann das Gewicht mit den Befehl `SETWEIGHT(net,l1,n1,l2,n2,val)` zwischen den angegebenen Neuronen gesetzt bzw. mit `GETWEIGHT(val,net,l1,n1,l2,n2)` gelesen werden. Die Parameter (z. B. der Schwellenwert oder

die Aktivierungsfunktion) eines einzelnen Neurons können mit weiteren Befehlen gesetzt oder gelesen werden. Statt ein Neuron mit der Schicht und der Position in der Schicht ($layer, neuron$) zu identifizieren, kann auch der Name des Neurons, der im sogenannten *Identifier* (vgl. Abschnitt 6.1.3.1) festgelegt ist, verwendet werden. Fehler beim Zugriff auf das Netz, etwa der Zugriff auf ein nicht vorhandenes Neuron, werden wie andere Zugriffe auf nicht existente Elemente mit einer *Fehlermeldung* angezeigt. Die Berechnung des Netzes *net* erfolgt mit dem Befehl `APPLYNET(net)`, wobei die Reihenfolge der Neuronen und Schichten die Ordnung auf die Auswertungsreihenfolge definiert.

Befehle für die Verwendung von evolutionären Algorithmen

Zur Optimierung verwendeter Parameter werden diese in einem Programm markiert. Dazu stehen verschiedene Parameter zur Verfügung, deren Anwendung – wie in Abschnitt 6.1.4 beschrieben – erfolgt. Der zu optimierende Parameter wird mit dem Befehl `EAPARAM(var)` beschrieben, dazu muß dieser im Variablendefinitionsteil deklariert sein. Bei *var* kann es sich um eine einzelne Fuzzy-Variable oder aber um eine gesamte Fuzzy-Mengen-Basis handeln. Die zu verwendenden Bilddaten können mit den Befehlen `EASOURCEIMAGE(img)` und `EADESTIMAGE(img)` festgelegt werden. An diesen Stellen werden genau die Bilddaten verwendet, die zuvor mit dem Befehl `COMPAREIMAGE` ausgewählt wurden (vgl. Abschnitt 6.1.4). Das im Programm berechnete Ergebnisbild, welches schließlich mit dem Zielbild verglichen werden soll, wird durch `EAFINALIMAGE(img)` festgelegt. Wie der Vergleich zu erfolgen hat, wird im Optimierungsteil (vgl. Abbildung 6.19) festgelegt. Sobald das Schlüsselwort `EAREADY()` gelesen wird, wird die Programmausführung für die Optimierung an dieser Stelle abgebrochen. Dadurch kann der Anwender steuern, bis wann im Programmteil für die Optimierung relevante Befehle folgen. `EADESTIMAGE(img)` kann angegeben werden, um das Zielbild eventuell zu modifizieren, ansonsten wäre auch die Beendigung mit `EAREADY()` ausreichend.

7 Prototyp für die Segmentierung von MRT-Kopfbildern

Nachdem im vorangegangenen Kapitel eine Vielzahl einzelner Bausteine des Rahmenmodells vorgestellt wurden, werden diese nun in einem Anwendungsbeispiel verwendet, um ihren Nutzen herauszustellen. Aufgabe ist die automatische Segmentierung des Gehirns in MRT-Datensätzen. Das Bildmaterial hat eine Größe von 256×256 Bildpunkten und besteht aus bis zu 128 Schichten, wovon die jeweils mittlere sagittale Schicht verarbeitet wird. Im Bild werden maximal 256 Grauwerte (Intensitätswerte) unterschieden. Die zur Verarbeitung notwendigen Schritte werden im folgenden explizit gemacht. Einige kleinere voneinander unabhängige Beispiele für die Anwendung der vorgestellten Methoden finden sich zudem in Anhang B. In diesem Kapitel werden vier Beispieldatensätze verwendet. Die Auswahl erfolgte derart, daß diese Datensätze das „normale“ Spektrum an Variabilitäten abdecken. Extreme Abweichungen (Artefakte, die beispielsweise große Bildregionen überdecken oder Translationen beinhalten), wurden nicht verwendet, da hier eine gesonderte Behandlung notwendig ist, etwa durch zusätzliches Spezialwissen oder weitere Verarbeitungsmethoden.

7.1 Algorithmische Vorgehensweise

Abbildung 7.1 zeigt die grobe Vorgehensweise zur Segmentierung des Bildes und der Klassifizierung der Segmente. Im Laufe dieses Kapitels werden die einzelnen Teile verfeinert und näher betrachtet. Jeweilige Zwischenergebnisse der Bilddaten bei der Anwendung der einzelnen Schritte ergänzen die Abschnitte. Auf den Definitionsteil verwendeter Variablen wird hier bewußt verzichtet, vielmehr erfolgt die Namenswahl derart, daß der Typ der Variablen aus der Bezeichnung erkennbar ist. In Anlehnung an die ungarische Notation steht der Typ dem eigentlichen Bezeichner voran.

Nach der Auswahl des zu verarbeitenden Bildes wird zunächst untersucht, ob der Datensatz ein 3D-Datensatz ist und dann entsprechend die mittlere Schicht ausgewählt. Diese Schicht dient als Basis für die Weiterverarbeitung und wird der Vorverarbeitung zugeführt (vgl. Abbildung 7.2). Die Vorverarbeitung normiert das Bild so weit wie möglich auf einen rechteckigen Bereich, um die weitere

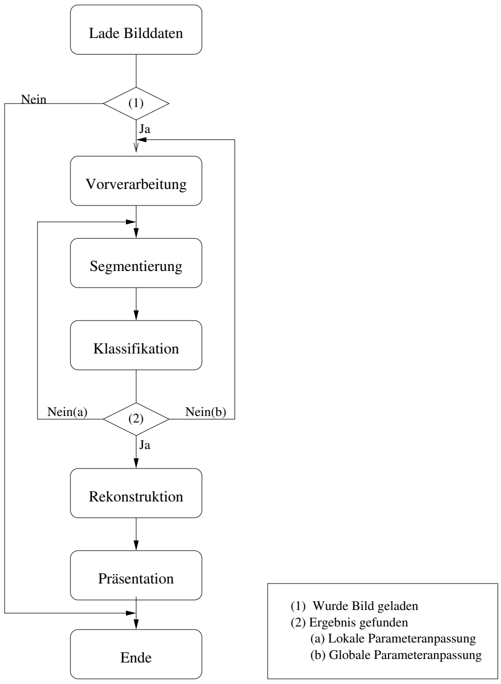


Abbildung 7.1: Vorgehensweise bei der Segmentierung und Klassifikation von MRT-Daten des Kopfes.

Arbeit zu erleichtern. Anschließend wird das Bild segmentiert und versucht, aus der Liste der Segmente diejenigen zu finden, die zu den Beschreibungen in der Wissensbasis den insgesamt höchsten Ähnlichkeitsgrad (Zugehörigkeitsgrad) besitzen. Kann für ein in der Wissensbasis beschriebenes Konzept kein passendes Segment gefunden werden, erfolgt regelbasiert eine Parameteranpassung. Andernfalls werden die Segmente mit den Konzeptnamen versehen und die Originalgrauwerte wiederhergestellt, bevor eine Ausgabe der klassifizierten Segmente erfolgt.

7.2 Vorverarbeitung der Bilddaten

Das Ziel der Vorverarbeitung ist die Verbesserung der Bilddaten in der Art, daß eine weitere Verarbeitung des Bildes dadurch erleichtert wird. Im vorliegenden Beispiel wird eine Normierung des Bildmaterials bezüglich der Größe und der Grauwertverteilung durchgeführt (vgl. Abbildung 7.3).

Die dazu notwendigen Befehle finden sich in Abbildung 7.4, die vier genannten Bildbeispiele der Anwendung der Funktion zeigt Abbildung 7.5. Die Berechnung der Normierung erfolgt ohne merkliche Verzögerung.

Funktion: GetImage

```
FUNCTION(GetImage)
  GetLFileName(strFName,\ImageData\*.*)
  IfNotEmpty(strFName)
  {
    LoadImage(imgMainImage,strFName)
    IfNotEmpty(imgMainImage)
    {
      GetSizeZ(intSlides,imgMainImage)
      IfGreaterThan(intSlides,1)
      {
        Div(intSlides,intSlides,2)
        CopySlide(imgMainSlide,imgMainImage,SAGITTAL,intSlides)
      }
      Else #IfGreaterThan(intSlides,1)
      {
        CopyImage(imgMainSlide,imgMainImage)
      }
    }
  }
  Else #IfNotEnpty(strFName)
  {
    Halt(Fehler: Kein Bild gewählt)
  }
RETURN()
```

Abbildung 7.2: Quelltext zum Einladen der Bilddaten

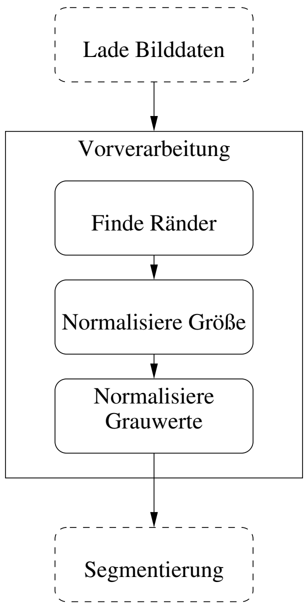
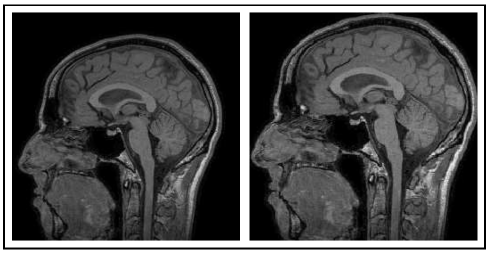


Abbildung 7.3: Vorgehensweise bei der Vorverarbeitung von MRT-Daten des Kopf-es.

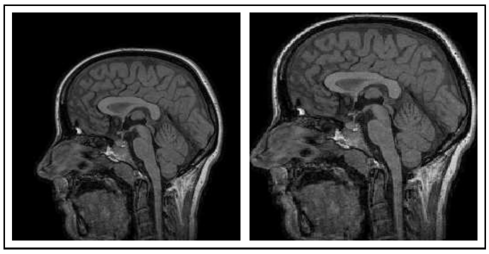
Funktion: PreProcessing

```
FUNCTION(PreProcessing)
  HistogramMinimum(intThreshold,imgMainSlide)
  IfSmallerThan(intThreshold,20)
  {
    # Empirischer Wert
    Set(intThreshold,40)
    Wait(Info: Threshold wurde fix gesetzt)
  }

  GetBorder(intLX,intHX,intLY ,intHY,intLZ, intHZ, imgMainSlide, intThreshold)
  ScaleImage(imgScaledImage,imgMainSlide,intLX, intHX, intLY, intHY,intLZ, intHZ,256,256,1)
  ScaleGrayscales(imgGSScaledImage,imgScaledImage,0,255)
RETURN()
```

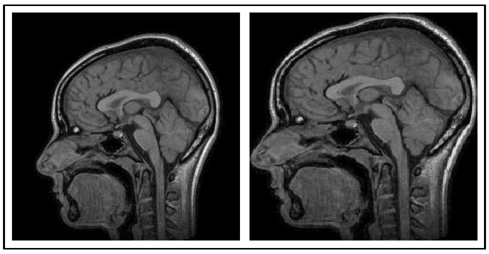
Abbildung 7.4: Quelltext zur Vorverarbeitung der Bilddaten



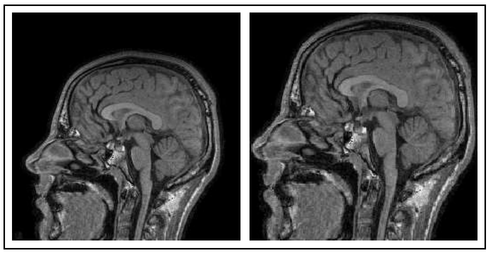
(a) Beispiel_1



(b) Beispiel_2



(c) Beispiel_3



(d) Beispiel_4

Abbildung 7.5: Ergebnis der Vorverarbeitung von MRT-Bilddaten jeweils vor und nach der Normierung von Größe und Grauwertbereich.

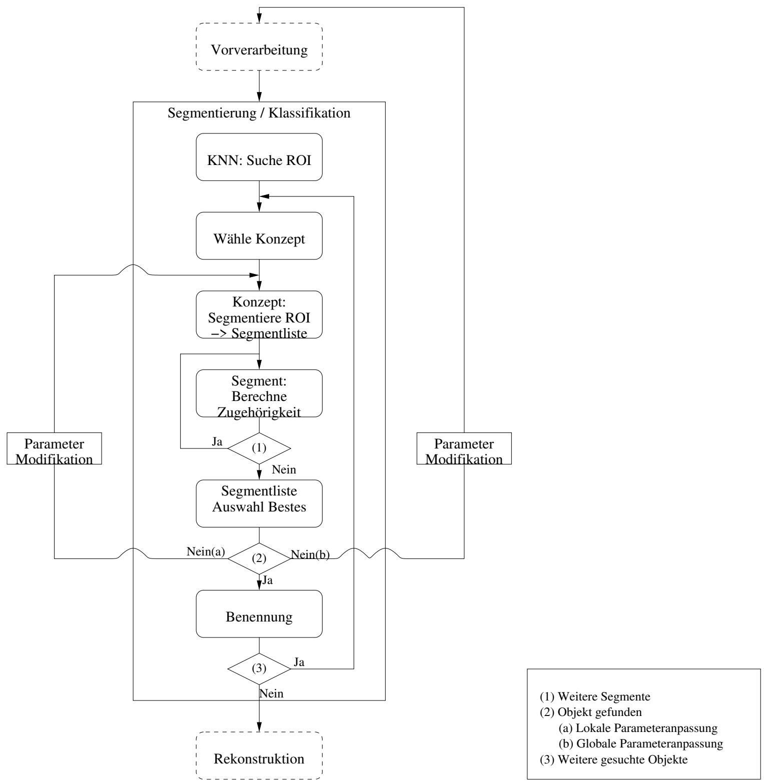


Abbildung 7.6: Segmentierung und Klassifikation nach der Vorverarbeitung.

7.3 Segmentierung und Klassifikation

Die Segmentierung stellt den wichtigsten Teil in der Prozeßkette dar und bildet die Grundlage für eine anschließende Klassifikation. Das im Anwendungsbeispiel verwendete Vorgehen wird in Abbildung 7.6 gezeigt und im folgenden weiter erläutert.

In einem ersten Schritt wird mit Hilfe eines vortrainierten neuronalen Netzes der Suchbereich eingeschränkt, um alle Bildpunkte auszuschließen, die nicht zum Bereich des Gehirns gehören. Anders als bei [Rit98] wird hier allerdings die Kopfkontur nicht manuell, sondern automatisch generiert und auf die angegebene Anzahl von Punkten äquidistant reduziert. Die dazu notwendigen Befehle werden in Abbildung 7.7 für die Vorsegmentierung des gesamten Gehirns dargestellt, anschließend zeigt Abbildung 7.8 die berechneten Bilder, wobei als Eingabebilder die Ergebnisbilder aus Abbildung 7.5 verwendet wurden. Auch diese Berechnung erfolgt ohne merkliche Verzögerung im Bereich $\ll 1$ Sekunde.

Bemerkung 7.1 Alternativ besteht die Möglichkeit, Funktionen(namen) an die Konzepte zu binden und somit das Bild für die Vorsegmentierung auf kleinere, dem Konzept entsprechende, Regionen einzuschränken. Dazu muß ein zum

Funktion: ANNROIDetectionBrain

```
FUNCTION(ANNROIDetectionBrain)
  LoadANN(annROIBrainDetection,BrainNet.net)
  GetNumInputNeurons(intVal,annROIBrainDetection)
  GetNumLayers(intLastLayer,annROIBrainDetection)
  GetContour(plContour,imgGSSCaledImage,intVal)
  Set(intCounter,0)
  # Setze Eingabewerte
  for(intI,1,intVal)
  {
    GetPointX(intX,plContour,intI)
    GetPointY(intY,plContour,intI)
    Add(intCounter,intCounter,1)
    SetNet(annROIBrainDetection,intCounter,1,intX)
    Add(intCounter,intCounter,1)
    SetNet(annROIBrainDetection,intCounter,1,intY)
  }

  ApplyNet(annROIBrainDetection)

  Set(intCounter,0)
  # Lese Ausgabewerte
  for(intI,1,intVal)
  {
    Add(intCounter,intCounter,1)
    GetNet(intX,annROIBrainDetection,intCounter,intLastLayer)
    Add(intCounter,intCounter,1)
    GetNet(intY,annROIBrainDetection,intCounter,intLastLayer)
    SetPointListX(plContour,intX, intI)
    SetPointListY(plContour,intY, intI)
  }
  ClipImage(imgClippedImage,imgGSScaledImage,plContour,INSIDE)
RETURN()
```

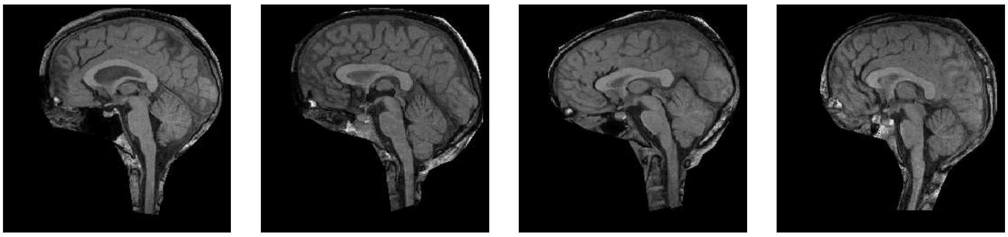
Abbildung 7.7: Quelltext zur ROI-Bestimmung des Gehirns mit Hilfe eines neuronalen Netzes.

Konzept passendes neuronales Netz[1] in einer im Konzept angegebenen Funktion existieren. Die Reihenfolge *Wähle Konzept* und *Suche ROI* in Abbildung 7.6 dreht sich dann um. □

An dieser Stelle wird unter Berücksichtigung der in der Wissensbasis gespeicherten Konzepte gezielt nach den dort beschriebenen Strukturen gesucht. Dazu werden weitere lokale unscharfe Filter (vgl. Abbildung 5.7) genutzt, die den Suchbereich neben der ROI-Detektion mit Hilfe des neuronalen Netzes zusätzlich einschränken. Eine Vorsegmentierung mit Hilfe von neuronalen Netzen ist dann sinnvoll, wenn dadurch eine genauere Einschränkung auf den gesuchten Bereich im Bild erfolgt als dies durch unscharfe Positionsfilter möglich ist.

Das Ergebnis der ROI-Bestimmung wird in einem weiteren Schritt segmentiert, wie in Abschnitt 5.2.2 beschrieben. Jedes Segment wird durch eine ikonische

[1] Die Anwendung eines Netzes in einer solchen Funktion ist nur ein zum Kontext passendes Beispiel, prinzipiell besteht für die Funktion keine Einschränkung, außer daß sie existieren muß.



(a) Beispiel_1 (b) Beispiel_2 (c) Beispiel_3 (d) Beispiel_4

Abbildung 7.8: Ergebnis nach der Bestimmung der ROI mit Hilfe von neuronalen Netzen.

Fuzzy-Menge dargestellt. Aufgrund der aufwendigen Berechnung der Segmente unter Berücksichtigung der zugrundeliegenden Sprache und Interpretation existiert eine implementierte Funktion, die ein Bild segmentiert und in alle gefundenen Segmente in einer Segmentliste speichert. Diese können dann weiter bewertet werden. Die α-Segmente (vgl. Definition 5.9) der in der Liste gespeicherten Segmente werden anschließend mit Hilfe der Funktion *Compare* (vgl. Abschnitt 6.1.2.7) bewertet und anschließend, wie in Abschnitt 5.2.4.2 beschrieben, das beste ausgewählt. Abbildung 7.9 zeigt den passenden Quelltext der Segmentierung. Entsprechend werden in Abbildung 7.10 einige wenige Beispiele gefundener Segmente gezeigt, während Abbildung 7.11 das jeweils als bestes ausgewählte Segment darstellt, wobei hier noch nach der Segmentierung eine Schließung der Löcher als Nachverarbeitungsschritt erfolgte.

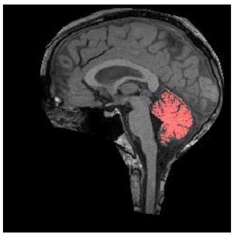
7.4 Rekonstruktion der Originalbilddaten und Präsentation

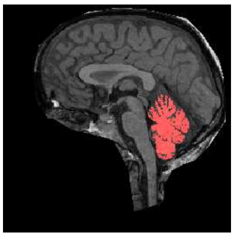
Da die Grauwerte während der Verarbeitung verändert (normiert) wurden, sind den gefundenen Segmenten wieder die Originalgrauwerte zuzuweisen, bevor das Ergebnis der Segmentierung präsentiert wird. Ebenso wird die Größenskalierung wieder rückgängig gemacht. Dieses ist notwendig, da die Ergebnisse in das Originalbild eingeblendet werden. Weil zu jedem Bild die Skalierung (in den d_x, d_y) gespeichert wird, ist diese Operation möglich. Abbildung 7.12 zeigt die Darstellung der Ablaufes der Rekonstruktion. Sofern kein Ergebis gefunden wurde, wird eine entsprechende Meldung ausgegeben. Anschließend endet die Programmausführung.

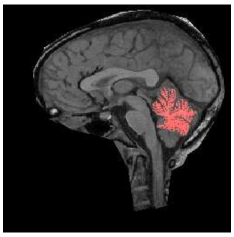
Funktion: Segmentation

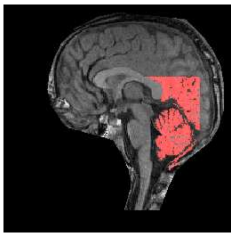
```
FUNCTION(Segmentation)
  Einladen der Fuzzymengenbasis, die notwendigen Interpretationen enthaelt
  LoadFuzzySets(fbFSList,MRTBrain.fsets)

  # Bestimme eine Liste aller Segmente mit einer Beschreibung aus aConcept (BWV)
  Segmentate(slSegList, imgClippedImage, conConcept, fbFSList, 64)

  GetCount(intSegCount,slSegList)

  for (intJ,1,intSegCount,1)
  {
     GetSegment(segSegment,slSegList,intJ)
     Compare(conConcept,segSegment,fbFSList,slResultList)
     UpdateList(slSegList,segSegment,intJ)
  }
RETURN()

FUNCTION(Classification)
# Einladen der Wissensbasis
LoadKnowledgeBase(kbKBase,MRTBrain.kbase)

# Anzahl der Konzepte
GetCount(intNumConcepts, kbKBase)

# Fuer alle Konzepte ...
For(intI,1,intNumConcepts,1)
{
  GetConcept(conConcept,kbKBase,intI)
  GetName(strConceptName,conConcept)
  CALL(Segmentation)
  GetBestSegment(segSegment,slSegList)
  SetName(segSegment,conConcept)
  AddListElement(slResultList,segSegment)
}
RETURN()
```

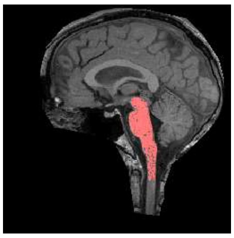
Abbildung 7.9: Quelltext zur Segmentierung des Bildes aus dem ROI-Bild mit Hilfe eines Bereichswachstumsverfahrens und Klassifikation.


(a) Bsp_1 Kleinhirn

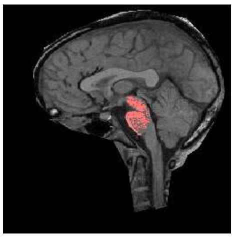

(b) Bsp_2 Kleinhirn

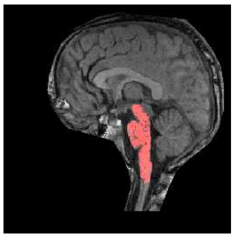

(c) Bsp_3 Kleinhirn

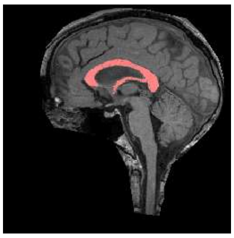

(d) Bsp_4 Kleinhirn

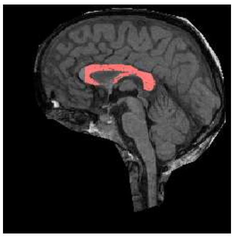

(e) Bsp_1 Hirnstamm

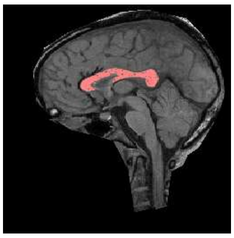

(f) Bsp_2 Hirnstamm

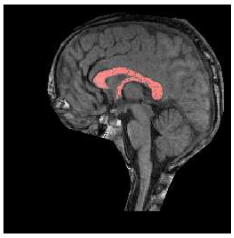

(g) Bsp_3 Hirnstamm

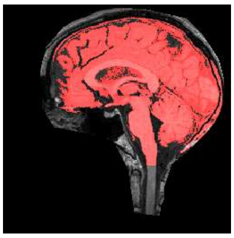

(h) Bsp_4 Hirnstamm


(i) Bsp_1 Balken


(j) Bsp_2 Balken


(k) Bsp_3 Balken


(l) Bsp_4 Balken


(m) Bsp_1 Gehirn


(n) Bsp_2 Gehirn


(o) Bsp_3 Gehirn

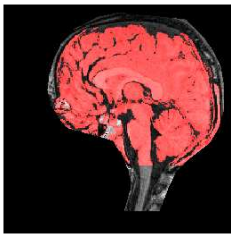
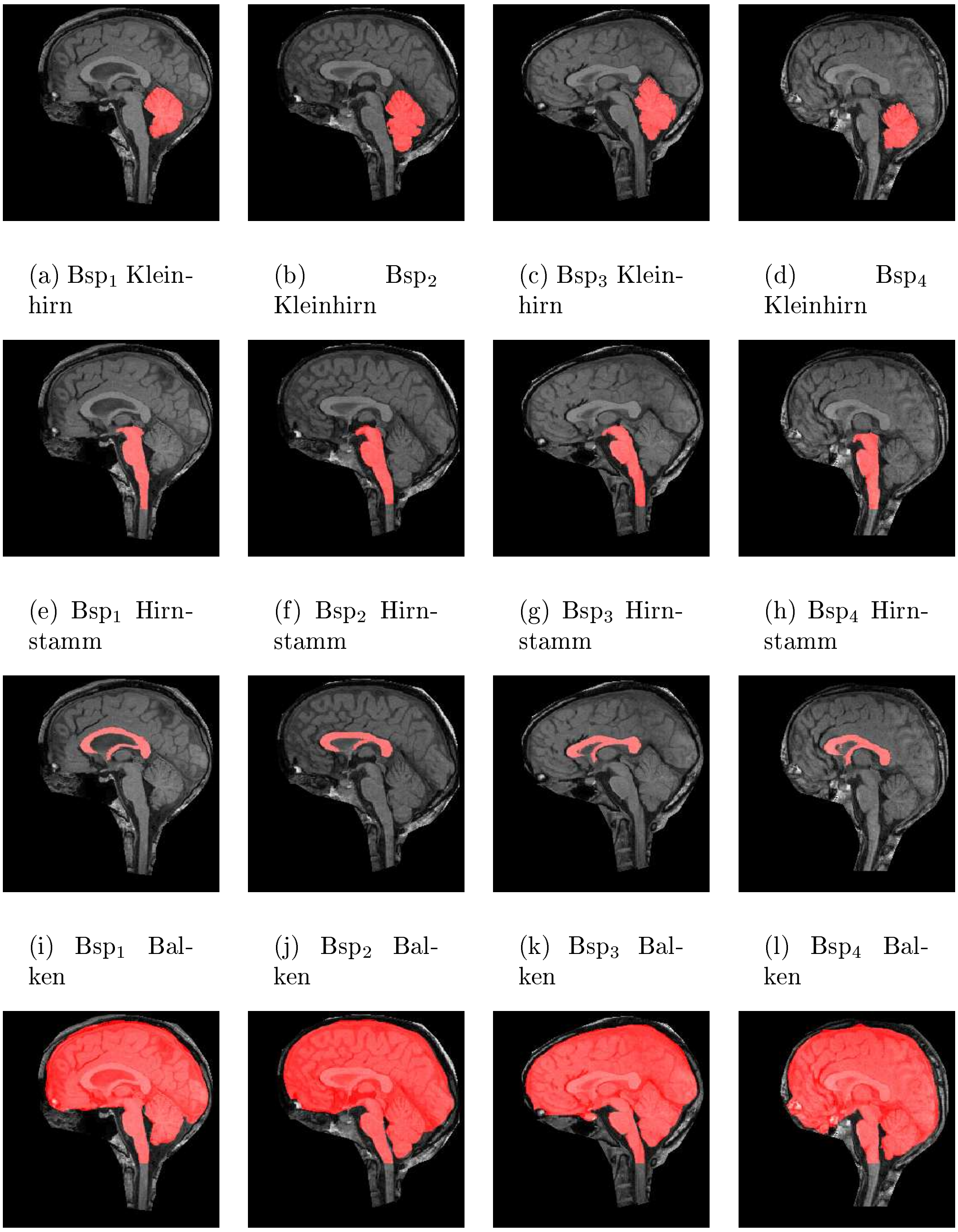

(p) Bsp_4 Gehirn

Abbildung 7.10: Darstellung einiger Zwischenergebnisse der Segmentierung (ohne Wertung).



(a) Bsp_1 Kleinhirn
(b) Bsp_2 Kleinhirn
(c) Bsp_3 Kleinhirn
(d) Bsp_4 Kleinhirn

(e) Bsp_1 Hirnstamm
(f) Bsp_2 Hirnstamm
(g) Bsp_3 Hirnstamm
(h) Bsp_4 Hirnstamm

(i) Bsp_1 Balken
(j) Bsp_2 Balken
(k) Bsp_3 Balken
(l) Bsp_4 Balken

(m) Bsp_1 Gehirn
(n) Bsp_2 Gehirn
(o) Bsp_3 Gehirn
(p) Bsp_4 Gehirn

Abbildung 7.11: Ergebnisse nach Auswahl der besten Segmente entsprechend der Vorgabe in der Wissensbasis.

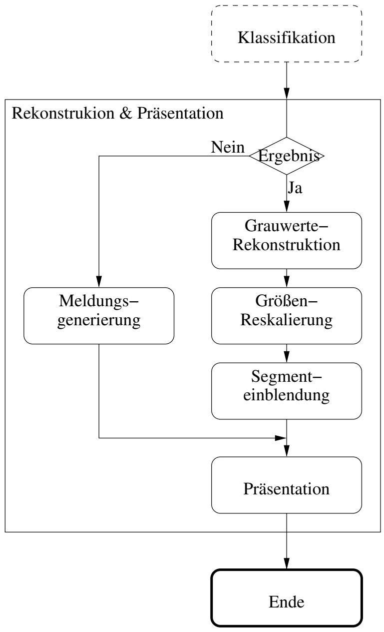


Abbildung 7.12: Rekonstruktion und Präsentation der Ergebnisse.

7.5 Parameteradaptionen

Falls in einem Durchlauf keines der gefundenen Segmente der Beschreibung eines Konzeptes ausreichend[2] entspricht, kann eine Parameteradaption erfolgen. Diese kann in einem ersten Schritt durch Änderung der im Konzept geforderten Eigenschaften oder der Modifikation der verwendeten Fuzzy-Mengen erfolgen. Die Änderung ist lokal zu betrachten und bezieht sich nur auf das aktuelle Konzept. Eine solche Modifikation ist beliebig oft wiederholbar und wird vom Anwender (beispielsweise durch eine Zählervariable für die Iterationen) gesteuert. Auch über den Trennbefehl `SPLIT` oder Vereinigungsbefehl `MERGE` kann eine zusätzliche Menge von Segmenten erzeugt werden.

Die Parameteranpassung erfolgt regelbasiert. Dazu werden die in einer Regelbasis hinterlegten Regeln zur Modifikation der in den Konzepten genutzen Fuzzy-Mengen verwendet, wobei sowohl der Typ als auch die weiteren Parameter (die 4 Stützstellen) der Fuzzy-Mengen verändert werden können.

Führt auch dies zu keinem akzeptablen Ergebnis (wieder in Bezug auf den Schwellenwert), muß eine (vom Anwender realisierte) globale Anpassung (Änderung von Schwellenbwerten oder eine aufwendigere Vorverarbeitung oder Verwendung von Spezial-Wissen) durchgeführt werden.

[2] Ausreichend in dem Sinne, daß der geforderte Schwellenwert erreicht wird.

7.6 Resümee

Wie sich gezeigt hat, ist die Segmentierung von den hier betrachteten Bilddaten mit Hilfe der genutzen CI-Methoden und der verwendeten Programmiersprache CIM^{2}BA/P einfach durchführbar. Der Umfang der benötigten Routinen ist leicht überschaubar und verständlich. Für die Durchführung der Segmentierung eines einzelnen Bilddatensatzes mit den vier gesuchten Strukturen *Gehirn* (als Ganzes), *Kleinhirn*, *Hirnstamm* und *Balken* benötigt auf einem (heutigen) Standardrechner (1 GHz Pentium-Klasse) ca. 15-25 Sekunden. In dieser Zeit werden (abhängig vom Bilddatensatz und der gewählten Schwelle für die α-Schnitte) mehrere hundert α-Segmente bewertet und selektiert. Insgesamt standen 50 Testdatensätze zur Verfügung. Bei 42 (84%) Datensätzen konnte eine korrekte Segmentierung durchgeführt werden. Die restlichen 8 (16%) Datensätze wiesen viele Artefakte auf, so daß schon die Segmentierung scheiterte. Hierbei lagen Fehler wie Translationen oder starke Verschattungen in Teilbereichen der Bilder vor. Von den 42 übrigen korrekt verarbeiteten Bildern wurden 34 ($\approx$ 81%) Datensätze im ersten Durchlauf (ohne Parameteränderung), 7 ($\approx$ 17%) Datensätze im zweiten Durchlauf (lokale Parameteränderung) und 1 ($\approx$ 2%) Datensatz im dritten Durchlauf (weitere lokale Parameteränderung) segmentiert und klassifiziert.

8 Abschlußbemerkungen und Ausblick

8.1 Abschlußbemerkungen

Das Ziel dieser Arbeit war (laut Abschnitt 1.1) die

> *Entwicklung eines Rahmenmodells, welches es Anwendern insbesondere aus dem medizinischen Umfeld erlaubt, neben den bekannten Bildverarbeitungsmethoden auf umfangreiche Bibliotheken von Methoden aus dem Bereich der Computational Intelligence zurückzugreifen.*

Es wurde im Rahmen dieser Arbeit untersucht, inwieweit sich existierende Bildverarbeitungsmethoden um Methoden aus dem Bereich der Computational Intelligence ergänzen lassen und welche Vorteile (bzw. auch Nachteile) das mit sich bringt.

Wie zu sehen war, existieren viele Arbeiten, die die Anwendung von CI-Methoden in der Mustererkennung und hier speziell der Bildverarbeitung untersuchen. Die Verarbeitung unsicherer und unscharfer Informationen wurde durch die in diesen Arbeiten entwickelten Ansätze erleichtert. Jedoch untersuchen diese Arbeiten jeweils nur kleine spezialisierte Teilgebiete der jeweiligen CI-Methoden, so daß hier ein ganzheitlicher Ansatz fehlt.

Deshalb wurde im Rahmen der vorliegenden Arbeit das neue, umfassende Modell CIM^{2}BA entwickelt. CIM^{2}BA ermöglicht die Verwendung aller relevanten CI-Methoden an den Stellen der Bildverarbeitungskette, an welchen besonders gute Ergebnisse bei solchen Bildtypen zu erzielen sind, die sich durch unscharfe Konzepte umgangssprachlich beschreiben lassen.

- CIM^{2}BA setzt voraus, daß eine Anzahl klassischer Bildverarbeitungsoperatoren fuzzifiziert werden mußte. Zudem war es erforderlich, neuartige Operatoren, insbesondere bei Segmentierung und Interpretation, zu entwickeln.

- Mit Hilfe dieser unscharfen Methoden in der Wissensbeschreibung wird dem Anwender die Möglichkeit geboten, die in den Bilddaten gesuchten Objekte zum einen natürlichsprachlich zu beschreiben und zum anderen die Interpretation der Beschreibung mit unscharfen Randbereichen zu formulieren.

- Durch die Nutzung neuronaler Netze kann erreicht werden, daß Objekte auch dann erkannt werden können, wenn ein unbekannter oder nicht beschreibbarer Zusammenhang zwischen Ein– und Ausgabedaten vorliegt und geeignete Trainingsdaten zur Verfügung stehen.

- Die aufwendige Optimierung derartiger Systeme und die Parametereinstellungen können mit Verwendung von evolutionären Algorithmen zu einem großen Teil automatisiert werden.

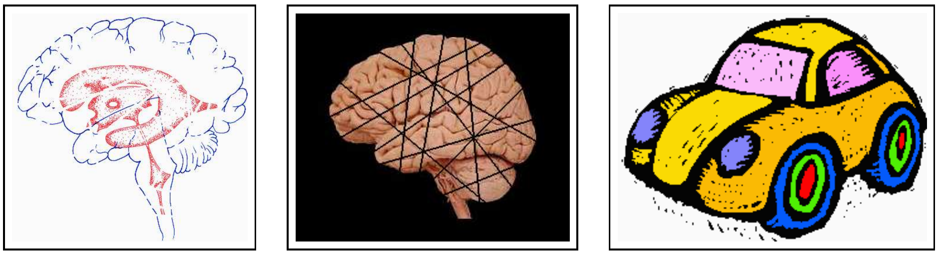
Das Modell wurde prototypenhaft als System CIM^{2}BA/P durch eine umfangreiche Menge an Befehlen implementiert, um die Methoden zu testen. Als Testfeld wurde die Segmentierung von MRT-Kopf-Bildern mit Hilfe von CIM^{2}BA/P durchgeführt und einige Strukturen des Gehirns, die in einer unscharfen Wissensbasis beschrieben waren, segmentiert und klassifiziert, wobei als Testdaten mehrere MRT-Bilddaten verwendet wurden.

Nicht behandelt wurde in dieser Arbeit die vollständige Berücksichtigung von 3D-Informationen. Die Akquisition medizinischer Bilddaten erfolgt meist in der Suche nach einer Diagnosebestätigung, so daß gezielt spezielle Bereiche aufgenommen werden. Die Schichtdicke oder der Schichtabstand zwischen den Aufnahmen wird dann in der Regel sehr dick gewählt und ist häufig auch nicht zusammenhängend. Dadurch bedingt sind die Schichtinformationen durch Partialvolumeneffekte beeinträchtigt. Unabhängig davon kann aber im Prototypen auf die verschiedenen Sichten der 3D-Datensätze (sagittal, transversal oder coronar) zugegriffen werden.

Auch nicht behandelt wurde die Berücksichtigung von mehrkanaligen Bildern, da sie in dem hier betrachteten medizinischen Gebiet der Radiologie nicht zu finden sind, Beispiele für unscharfe Farbbildverarbeitung aus einem technischen Umfeld finden sich z. B. in [VV00, HR00]. Der Umfang der implementierten Methoden umfaßt die vorgestellten Beispiele, stellt aber sicherlich nicht alle denkbaren Operatoren zur Verfügung, bspw. fehlen Operatoren zur Kantenverfolgung.

Insgesamt kann als Nachteil der Anwendung von CI-Methoden gesagt werden, daß ein erhöhter Rechenzeitbedarf besteht. Zudem kann es an Vertrauen des Nutzers in derartige Methoden mangeln. Speziell bei neuronalen Netzen fehlt die Nachvollziehbarkeit der Ergebnisse. Es darf nicht davon ausgegangen werden, daß CI-Methoden einen allgemeinen Problemlöser für alle Bildverarbeitungsprobleme dieser Welt liefern, vielmehr sollen sie als Unterstützung bestehender Methoden gesehen werden, mit denen sie in „friedlicher Koexistenz“ existieren.

Auf die Verwaltung und Verfügbarmachung von medizinischem Bildmaterial, was die sogenannte elektronische Patientenakte betrifft, wurde nicht näher eingegangen, da diese in der heutigen Zeit zwar eine noch offene und problembehaftete Fragestellung ist, was aber weniger durch technische als mehr durch rechtliche und datenschutzbedingte Aspekte bedingt ist.



(a) Schemazeichnung (b) Modell-Gehirn (c) Grafik

Abbildung 8.1: Beispiel für die Leistung des Menschen bei der Erkennung von Objekten. Das Erkennen der Gehirne ist trotz unterschiedlicher Darstellungsarten, Inhalten und starken Störungen (Unterbrechungen) problemlos möglich, ebenso wie das Erkennen „anderer“ Objekte.

8.2 Ausblick

Die Verwendung von CI-Methoden, die die einfache Verarbeitung von nicht exaktem Wissen erlauben (Fuzzy-Logik), anhand von Beispielen trainiert werden können (neuronale Netze) sowie gezielt nach Lösungen suchen können (evolutionäre Algorithmen), bringt das Erreichen der Zielsetzung nach einer automatisierten Bildverarbeitung weiter. Trotzdem ist die Leistungsfähigkeit dieser Systeme noch weit von der Bildverarbeitungsleistung des Menschen entfernt. Das Erkennen der in Abbildung 8.1 gezeigten Objekte ist für einen Menschen ohne großen Aufwand aufgrund seines apriori-Wissens möglich, obwohl sehr viele Einzelsegmente vorliegen, die für sich betrachtet nicht erkennbar sind. Für einen Menschen ist es auch kein Problem zu erkennen, daß das dritte Objekt kein Gehirn darstellt, obwohl die Kontur der Grafik auf ein Gehirn schließen lassen könnte.

Bildpunktbasierte Computersysteme sind bei solchen Beispielen aber sicherlich überfordert. Trotzdem kann an dieser Stelle mit Hilfe von apriori-Wissen die Leistungsfähigkeit gesteigert werden. Wenn dem System bekannt ist, daß nur Bilddaten von Gehirnen existieren, so wird zwar Abbildung 8.1(c) zu Problemen führen, aber die Formeigenschaften der anderen Bilder können auf Gehirne schließen lassen. Es gibt Ansätze, welche die Bilddaten in verschiedenen Auflösungsstufen betrachten, so daß zu Beginn die Details auf Bildpunktebene außer Acht gelassen werden. Dadurch ist bspw. eine Erkennung der Strukturen trotz vieler Einzelsegmente möglich.

Die folgenden Punkte nennen einige Erweiterungsmöglichkeiten für die implementierte Umsetzung des Rahmenmodells. Der hier vorgestellte Baukasten stellt eine Experimentierumgebung dar, welche die Auswertung von Bilddaten mit neuartigen Aufgabenstellungen erleichtert, aber auch mittels der Programmiersprache eine spätere Routinenutzung erlaubt, wobei jeweils auf Wissensbasen und neuronale Netze sowie evolutionäre Algorithmen zurückgegriffen werden kann. Trotzdem ist für weitere Anwendungen der Bedarf an neuen Operatoren vorhanden. Falls eine Verarbeitung von mehrkanalige Bilddaten oder von 3D-Bilddaten angestrebt wird, sind diese Operatoren zu integrieren.

Weiterhin kann die Erweiterung der verwendbaren Arten neuronaler Netze und der Lernverfahren sinnvoll sein. Zudem fehlt eine umfassende Realisierung der zellularen neuronalen Netze. Die implementierten evolutionären Algorithmen basieren ebenfalls auf Standardverfahren. Verbesserungen, wie sie etwa von [HRF99] vorgeschlagen werden, können bei der hier vorgestellten Anwendung ebenfalls untersucht werden. Die dort behandelten asymmetrischen Algorithmen erlauben eine zielgerichtetere Suche nach dem Optimum und lassen somit eine bessere Performanz erhoffen. Auch eine evolutionäre Optimierung der ebenfalls diskutierten Fuzzy-Objekt-Modelle ist denkbar. Dabei ist neben den Parametern des Unschärfegrades δ und des verwendeten Fuzzy-Mengen-Typs θ auch eine Modifikation der Form des FOMs möglich. Für eine FOM-Optimierung müssen neben Positivbeispielen auch Negativbeispiele verwendet werden, da ansonsten ein großer δ-Wert auch eine hohe Zugehörigkeit für beliebige Objekte berechnet. Die Parallelisierung bei der Anwendung evolutionärer Algorithmen ist implementiert, eine automatische Lastverteilung kann aber weitere Geschwindigkeitsvorteile bringen, dazu muß lediglich eine Untersuchung der Rechenleistung für einzelne Rechner berücksichtigt werden, indem etwa die Zeit für die Berechnung eines einzelnen Programmablaufs gemessen und verwendet wird.

Schließlich fehlt ein geeignetes Werkzeug, mit der eine benutzerkonforme ergonomische Oberfläche für ein CIM^{2}BA/P-Programm erstellt werden kann. Dieses ist für eine alltagstaugliche Anwendung notwendig.

Die Implementierung weist derzeit noch den Nachteil auf, daß die Sprache interpretiert wird und dadurch einen deutlichen Geschwindigkeitsnachteil gegenüber kompilierter Software aufweist. Neben einer Erweiterung des Befehlsumfanges der Programmiersprache kann somit auch über die Entwicklung eines Compilers für diese Sprache nachgedacht werden.

A Implementierung von CIM^{2}BA/P

In diesem Kapitel wird die benutzte Programmierumgebung vorgestellt, die für die Implementierung von CIM^{2}BA/P zur Verfügung stand. Zudem werden einige wichtige Klassen genannt. Anschließend wird gezeigt, wie ein CIM^{2}BA/P ausgeführt wird und wie neue Befehle hinzugefügt werden können.

A.1 Benutzte Umgebung

Die Implementierung des Prototypen CIM^{2}BA/P ist objektorientiert in Visual C++ der Version 6 unter dem Betriebssystem Windows™erfolgt. Zunächst wurde über eine Implementierung in JAVA nachgedacht, jedoch hat sich sehr schnell herausgestellt, daß diese Programmiersprache im Vergleich zu C++ für die vorgestellte Anwendung zu langsam ist. Vorteil wäre sicherlich die deutlich größere Plattformunabhängigkeit gewesen, allerdings dann zu Lasten der Antwortzeit des Systems. Für das Windows-Betriebssystem spricht zudem das Vorhandensein von einigen Routinen aus vorherigen Arbeiten, um den dann noch immer erheblichen Entwicklungsaufwand etwas zu reduzieren. Zudem konnte die Oberfläche mit Hilfe vom *Application Wizard* und *Class Wizard*, Werkzeugen zur Generierung von Programmgerüsten, realisiert werden.

A.2 Vorstellung der wichtigsten Klassen

Das System CIM^{2}BA/P stellt, wie in Kapitel 6 beschrieben, dem Anwender eine Arbeitsumgebung zur Verfügung, mit dem er verschiedene Wissensinhalte oder neuronale Netze verwalten, sowie in der Programmiersprache CIM^{2}BA/P geschriebene Programme entwickeln oder starten kann. An dieser Stelle werden die wichtigsten neu implementierten Klassen skizziert. Dabei teilen sich die Klassen in drei Bereiche: Daten(-typen), Sichten auf die Daten und die Programmabarbeitung.

A.2.1 Datenklassen

Die Datenklassen stellen im System verschiedene Datentypen zur Verfügung. Dabei leiten sich alle Klassen von einer allgemeinen Datenklasse *CDataObject* ab,

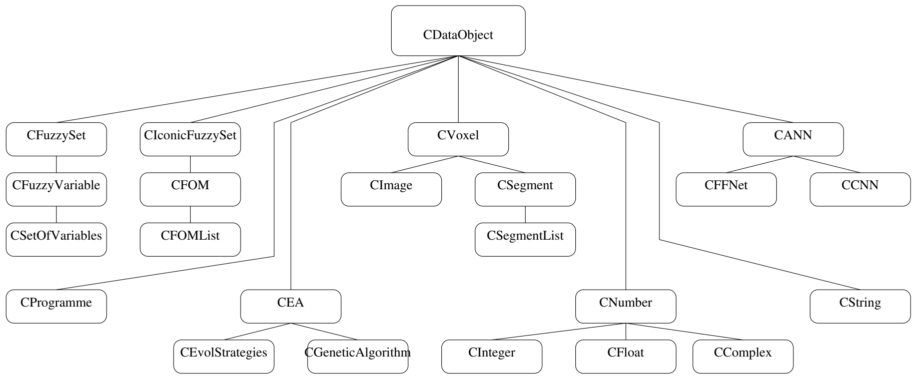


Abbildung A.1: Von CIM²BA/P zur Verfügung gestellte Datentypen.

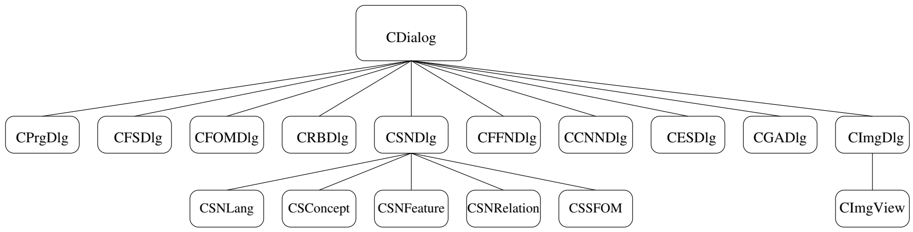


Abbildung A.2: Von CIM²BA/P zur Verfügung gestellte Eingabemasken.

die grundlegende – für alle Datentypen wichtige – Daten[1] bereitstellt. Einfache Datentypen sind Zahlen, Zeichenketten oder auch Bildpunkte. Weiterhin existieren Klassen für Datentypen wie Fuzzy-Mengen (*CFuzzySet*) oder Fuzzy-Variablen (*CFuzzyVariable*). Neben den Eigenschaften, welche die Klassen für die Datentypen speichern (etwa Name, Wertebereiche, o.ä.), stellen die Klassen bspw. Methoden zur Berechnung von Zugehörigkeitswerten zur Verfügung. Die Klasse CImage beinhaltet u. a. die Bildpunktmenge und bietet Methoden zur Bearbeitung des Bildes an. Eine Liste der Datentypen ist in Abbildung A.1 zu sehen.

A.2.2 Sichtenklassen

Zur Bearbeitung der Datenobjekte gibt es verschiedene Eingabemasken, wie sie in Abbildung A.2 aufgelistet werden. Zu jeder der in Abschnitt A.2.1 genannten Datenklassen existieren ein oder mehrere Sichtenklassen, die den Zugriff und die Manipulation auf die Eigenschaften erlauben. Die entsprechenden Masken wurden schon in Kapitel 6 vorgestellt.

[1] Also die *Eigenschaften* eines Objektes.

A.2.3 Programmabarbeitung

Die Abarbeitung eines in CIM^2BA/P realisierten Programmes erfolgt in einer eigenen Klasse, die einen *Programminterpreter* darstellt. Dadurch ist eine Trennung zwischen den Eingabemasken und der Programmausführung möglich. Die Klasse bindet dazu alle Datentypen ein und besitzt als Ein- und Ausgabemasken lediglich einen Teil der Sichten. Die eigentlichen Berechnungen oder Manipulationen von Objekteigenschaften erfolgen direkt bei den Objekten, so daß in der Programmabarbeitungsklasse lediglich die Schnittstelle zu den Objekten genutzt wird. Allerdings übernimmt diese Klasse die Speicherverwaltung und die bedingte Abarbeitung, falls Funktionsaufrufe, Schleifen oder Sprungbefehle benötigt werden.

A.2.4 Befehlserweiterung

Soll das System um weitere Befehle ergänzt werden, sind diese dem Programminterpreter mitzuteilen. Dieser verwendet die implementierten Klassen und steuert die Programmausführung, wie in Abschnitt A.2.3 beschrieben. Entsprechend muß zu jedem neuen Befehl eine Methode implementiert werden, welche die verwendeten Parameter auswertet und geeignet an die ausführenden Methoden weiterleitet.

Zu einem Befehl werden folgende Informationen verwaltet:

- Befehlsname
- Parameterliste
- Korrespondierende implementierte Methode

Die Parameterliste wird in der Form $[Typ_1|Typ_2|\ldots]$ für jeden Parameter abgelegt, wobei alle erlaubten Typen (einschließlich Konstanten) angegeben werden müssen. Mehrere mögliche Parameter P_i werden in der Form $[[P_1],[P_2],\ldots]$ hinterlegt. Zusätzlich wird ein Zeiger auf die Prozedur gesetzt, die mit dem Befehl identifiziert werden soll.

Bspw. kann der Befehl `LOADIMAGE` mit folgender Angabe eingefügt werden: `AddCommand("LoadImage","[img],[str|const]",@LoadImageMethod)`. Findet der Programm-Parser dann das Schlüsselwort `LOADIMAGE` (wobei zwischen Groß- und Kleinschreibung nicht unterschieden wird), wird die festgelegte Methode *LoadImageMethod* aufgerufen und versucht, die Parameter in angegebener Weise auszuwerten. Die Methode muß Bestandteil des Programminterpreters sein, so daß an dieser Stelle eine Neukompilierung des Quelltextes erfolgen muß, wenn eine neue Methode integriert wird.

B Tests und Ergebnisse

B.1 Fuzzy-Methoden

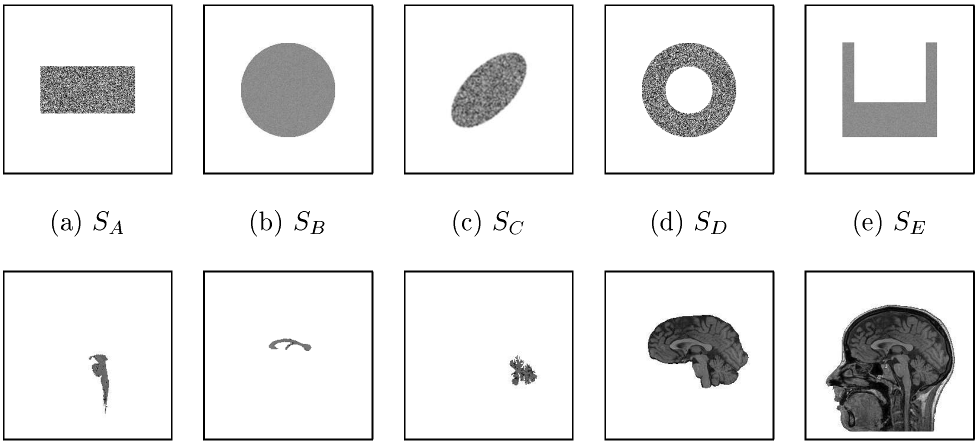
B.1.1 Beispiele für die Berechnung von Segmenteigenschaften

Die in Abschnitt 5.2.3 definierten Interpretationen für Segmenteigenschaften sollen anhand einiger Beispielsegmente (vgl. Abbildung B.1) vorgestellt werden. In diesen Beispielen finden aber keine unterschiedlichen Dimensionsmaße d_x und d_y Berücksichtigung. Tabelle B.1 zeigt die berechneten Werte der einzelnen Segmente S_i.

Schließlich wird in Abbildung B.2 die Berechnungsgrundlage für das Rundheitsmaß $roundness_C$ dargestellt, wobei die grau dargestellten Bildpunkte Punkte des Segmentes S sind, die nicht Teil des Referenzkreises C sind (also Punkte aus $S \cap \overline{C}$, (vgl. Gleichung (5.23))), während die schwarz dargestellten Bildpunkte Punkte des Referenzkreises C sind, die nicht zum Segment S gehören (also Punkte aus $\overline{S} \cap C$). Abbildung B.3 zeigt die Berechnung des Rundheitsmaßes $roundness_R$, welches den Abstand der äußeren Kontur zum Flächenschwerpunkt verwendet. In Abbildung B.4 wird die nach Definition 5.18 gegebene Berechnung der Hauptachse gezeigt. Die Abbildung B.5 stellt die konvexen Hüllen der einzelnen Segmente dar, die zur Berechnung des Maßes *convexity* herangezogen werden. Schließlich zeigt Abbildung B.6 die Berechnung der Eigenschaft *narrowness* als das Verhältnis eines maximal inliegenden Kreises zur Größe eines minimal umschließenden Kreises.

Die Abbildung B.7 zeigt einige Beispiele für erlaubte Relationen. Dabei stellt der Grauwert den Grad dar, wie sehr der Punkt die entsprechende Relation erfüllt. Weiße Bildpunkte erfüllen die Relation vollständig, schwarze Punkte erfüllen die Relation nicht. Entsprechend handelt es sich bei dieser Darstellung um ikonische Fuzzy-Mengen (vgl. Definition 3.17).

Die Abbildung B.8 zeigt die Verwendung von Subsegmenten bezüglich der Beschreibung *oben* mit dem 5-Tupel $(1, 0.4,\ 0.6,\ 1.0,\ 1.0)$.



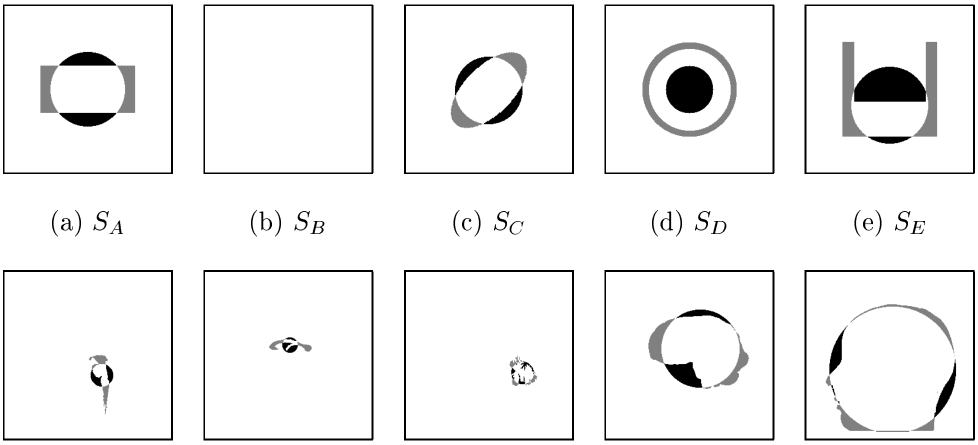
(a) S_A (b) S_B (c) S_C (d) S_D (e) S_E

(f) S_F (g) S_G (h) S_H (i) S_I (j) S_J

Abbildung B.1: Beispielsegmente für die Berechnung von Segmenteigenschaften.

Eigenschaft	S_A	S_B	S_C	S_D	S_E	S_F	S_G	S_H	S_I	S_J
Umfang(fast)	476.00	452.00	340.00	680.00	836.00	249.00	190.00	320.00	450.00	693.00
Umfang(exact)	476.00	527.00	434.85	794.15	836.36	270.41	219.65	343.39	516.69	773.98
Größe	12800	20156	10327	15044	13379	1253	582	1180	13976	36362
Helligkeit	99.04	100.08	128.47	99.42	139.98	100.58	113.99	86.78	73.30	69.39
Rundheit(C)	0.88	1.00	0.90	0.83	0.76	0.80	0.71	0.88	0.92	0.97
Rundheit(R)	0.77	1.00	0.74	1.00	0.57	0.50	0.40	0.81	0.73	0.89
Kompaktheit	0.71	0.91	0.69	0.30	0.24	0.22	0.15	0.13	0.66	0.76
Orientierung	90.00	90.00	135.00	90.00	92.46	11.76	85.95	124.26	38.87	8.79
Länglichkeit	0.75	0.00	0.75	0.00	0.40	0.93	0.95	0.45	0.61	0.27
Konvexität	1.00	1.00	0.98	0.75	0.52	0.64	0.53	0.72	0.91	0.96
Schmalheit	0.56	0.00	0.50	0.75	0.74	0.79	0.84	0.71	0.48	0.27
Homogenität(g)	0.55	0.55	0.60	0.55	0.94	0.85	0.88	0.81	0.79	0.71
Homogenität(l)	0.84	0.84	0.88	0.84	0.98	0.97	0.98	0.97	0.97	0.96

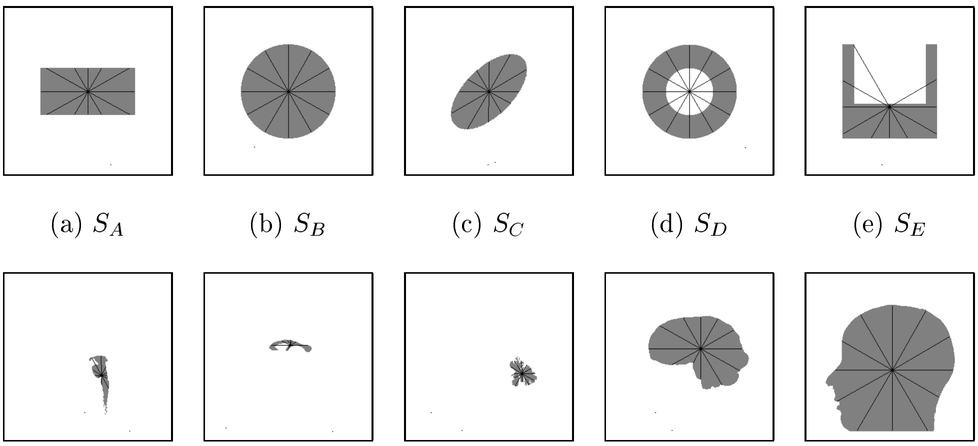
Tabelle B.1: Berechnung einiger Segmenteigenschaften.



(a) S_A (b) S_B (c) S_C (d) S_D (e) S_E

(f) S_F (g) S_G (h) S_H (i) S_I (j) S_J

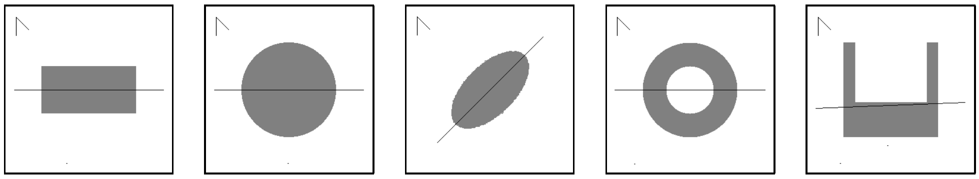
Abbildung B.2: Berechnung der Eigenschaft roundness$_C$ für die Objekte aus Abbildung B.1. Die grauen Bildpunkte sind Punkte des Objektes, die nicht Teil des Referenzkreises sind, während die schwarzen Bildpunkte zum Segment gehören, nicht aber zum Referenzkreis.



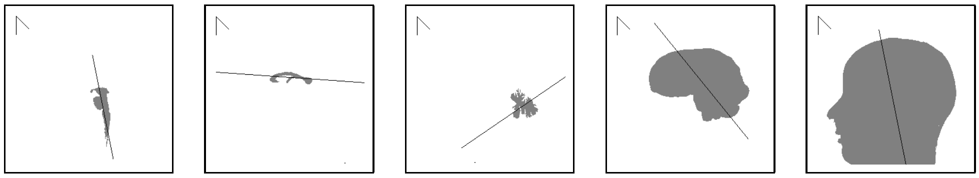
(a) S_A (b) S_B (c) S_C (d) S_D (e) S_E

(f) S_F (g) S_G (h) S_H (i) S_I (j) S_J

Abbildung B.3: Berechnung der Eigenschaft roundness$_R$ für die Objekte aus Abbildung B.1. Die Strahlen dienen zur Berechnung des Abstandes vom Flächenschwerpunkt zur äußeren Kontur des Objektes.

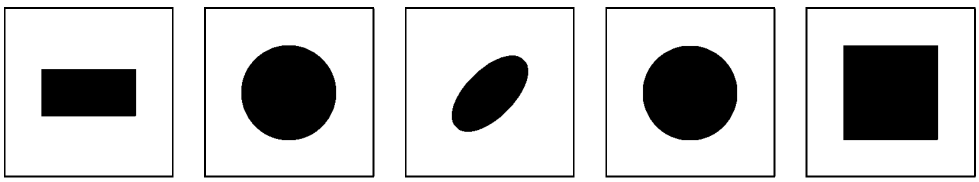


(a) S_A (b) S_B (c) S_C (d) S_D (e) S_E

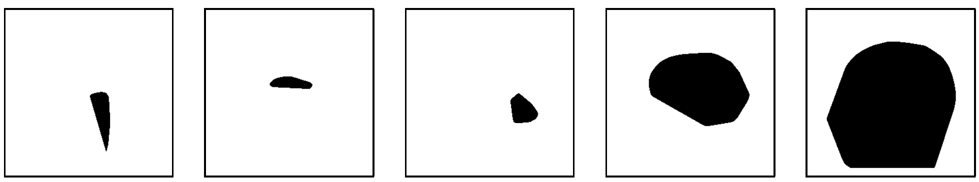


(f) S_F (g) S_G (h) S_H (i) S_I (j) S_J

Abbildung B.4: Berechnung der Eigenschaft orientation für die Objekte aus Abbildung B.1. Dargestellt wird die Hauptachse des jeweiligen Segmentes.

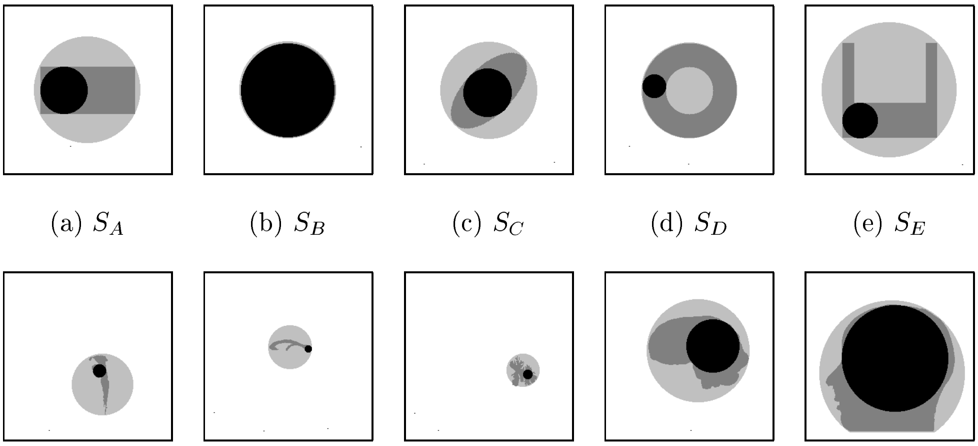


(a) S_A (b) S_B (c) S_C (d) S_D (e) S_E



(f) S_F (g) S_G (h) S_H (i) S_I (j) S_J

Abbildung B.5: Berechnung der Eigenschaft convexity für die Objekte aus Abbildung B.1. Dargestellt wird die konvexe Hülle des jeweiligen Segmentes.



(a) S_A (b) S_B (c) S_C (d) S_D (e) S_E

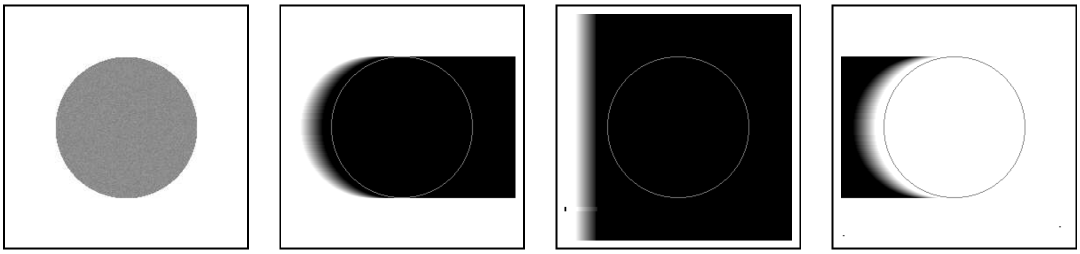
(f) S_F (g) S_G (h) S_H (i) S_I (j) S_J

Abbildung B.6: Berechnung der Eigenschaft narrowness für die Objekte aus Abbildung B.1. Dargestellt wird der maximal inliegende (schwarz) und minimal umschließende Kreis (hell) des jeweiligen Segmentes (mittel-hell).

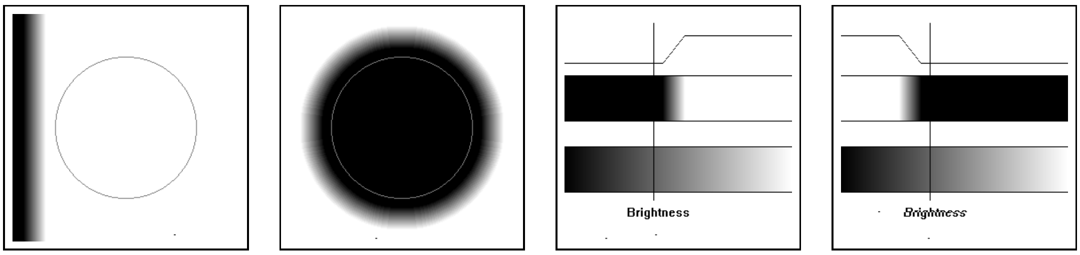
B.1.2 Segmentklassifikation mit Hilfe unscharfer Objekt-Modelle (FOM)

Für die in Abschnitt 5.2.3.4 beschriebenen Fuzzy-Objekt-Modelle werden an dieser Stelle einige Beispiele betrachtet. Als Daten dienen die schon in Abbildung 5.4 vorgestellten Strukturen. Neben unterschiedlichen Unschärfegraden δ werden auch zwei unterschiedliche Fuzzy-Mengen-Typen (vgl. Definition 6.2) betrachtet. Die Abbildung B.9 zeigt die Beispieldaten und die FOM für verschiedene Kleinhirne, Abbildung B.10 für verschiedene Hirnstämme und schließlich Abbildung B.11 für verschiedene Balken. Die entsprechenden Zugehörigkeitsgrade finden sich in Tabelle B.2 wieder.

Wie in der Tabelle zu sehen ist, stellt die Breite $\delta = 16$ einen guten Kompromiß für den Unschärfegrad dar. Die Berücksichtigung des Verhältnisses vom Umfang des betrachteten Objektes und des FOMs führt jedoch zu größeren Schwankungen. Entsprechend muß für jede Anwendung zunächst überprüft werden, inwieweit FOMs verwendbar sind und welche Bedeutung ihnen bei unterschiedlichen Bewertungen zugesprochen wird. Gegenproben, das heißt ein Vergleich von einem Objekt mit einem *nicht* dazu passenden FOM, ergaben für alle in der Tabelle betrachteten Fälle einen Zugehörigkeitswert kleiner 0.3, so daß eine Nichtübereinstimmung leicht erkennbar ist. Dieser kleine Wert ist durch die Berücksichtigung der Größenverhältnisse von FOM und Objekt zu erklären.

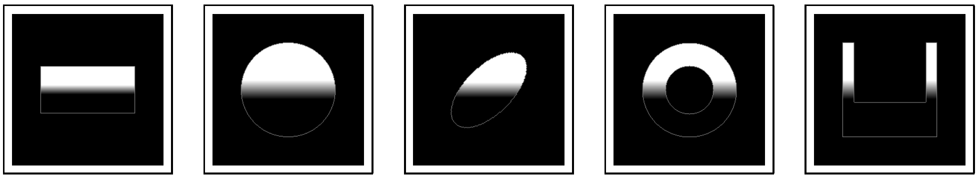


(a) Referenzobjekt (b) Liegt links von (c) Liegt total links von (d) Ragt nicht über

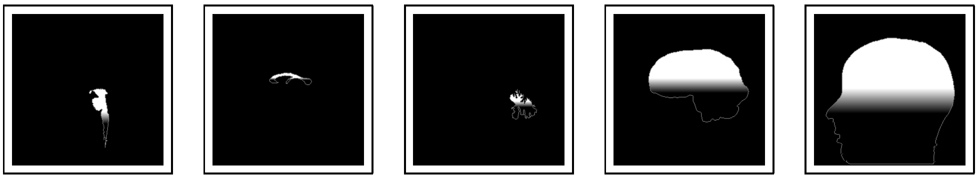


(e) Ragt nicht total links über (f) Liegt entfernt von (g) Ist heller als (h) Ist dunkler als

Abbildung B.7: Beispiele einiger Relationen, die zu einem Objekt möglich sind. Der Grauwert stellt den Grad dar, wie sehr der Punkt die Relation erfüllt. Für den unscharfen Bereich werden jeweils 10-35 Bildpunkte bzw. Grauwerte verwendet. Die Kontur des Referenzobjektes wird zur besseren Anschauung eingeblendet.

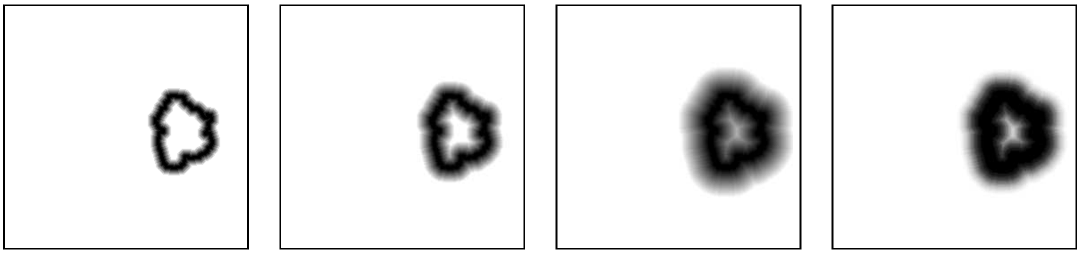


(a) S_A (b) S_B (c) S_C (d) S_D (e) S_E

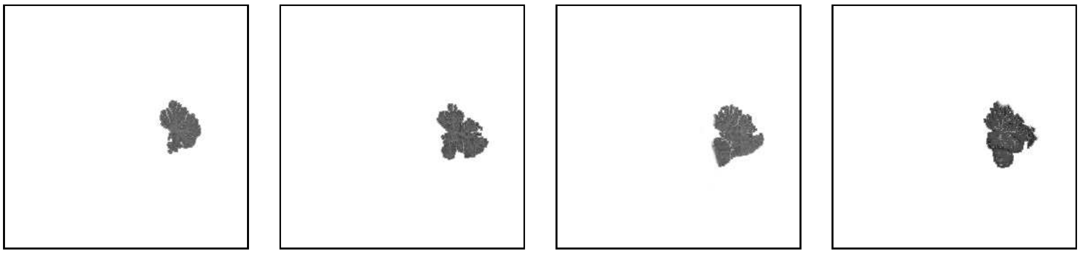


(f) S_F (g) S_G (h) S_H (i) S_I (j) S_J

Abbildung B.8: Berechnung der Beschreibung oben *für die Objekte aus Abbildung B.1, dargestellt als ikonische Fuzzy-Mengen. Die Beschreibung benutzt den Fuzzy-Mengen-Typ 1.*

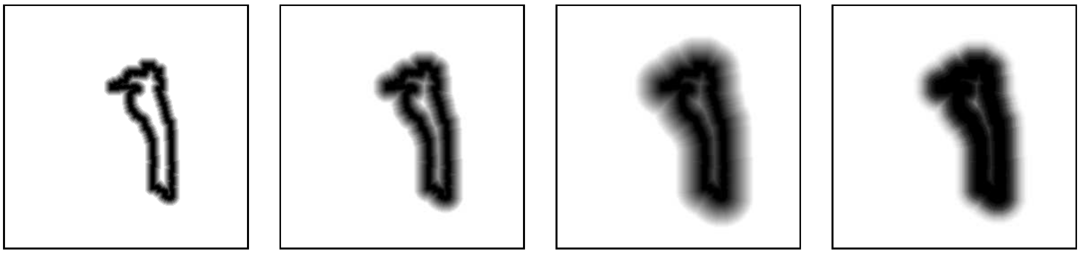


(a) $FOMKH(8,l)$ (b) $FOMKH(16,l)$ (c) $FOMKH(32,l)$ (d) $FOMKH(32,s)$

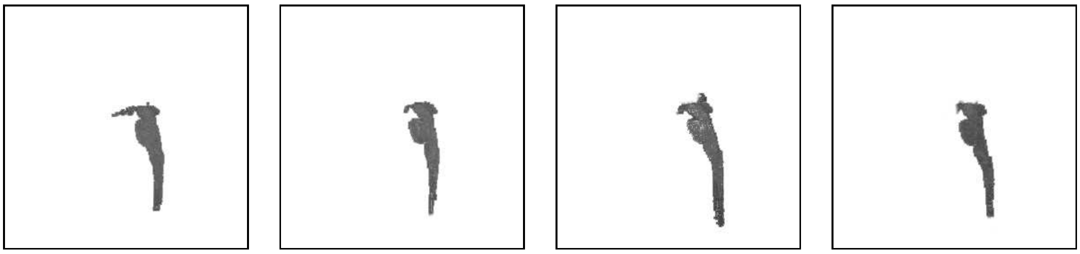


(e) $BspKH_1$ (f) $BspKH_2$ (g) $BspKH_3$ (h) $BspKH_4$

Abbildung B.9: Test von Fuzzy-Objekt-Modellen anhand des Beispiels Kleinhirn. Die Zahl in der Klammer gibt den gewählten Wert für δ an, l ist eine Fuzzy-Menge vom Typ 1 (linear), s vom Typ 2 (sigmoidal).

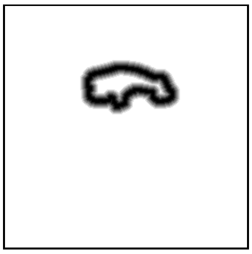


(a) $FOMHS(8, l)$ (b) $FOMHS(16, l)$ (c) $FOMHS(32, l)$ (d) $FOMHS(32, s)$

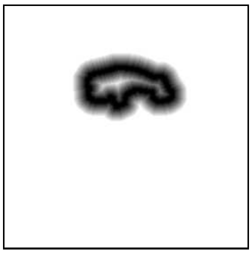


(e) $BspHS_1$ (f) $BspHS_2$ (g) $BspHS_3$ (h) $BspHS_4$

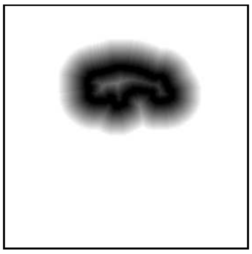
Abbildung B.10: Test von Fuzzy-Objekt-Modellen anhand des Beispiels Hirnstamm. *Die Zahl in der Klammer gibt den gewählten Wert für δ an, l ist eine Fuzzy-Menge vom Typ 1 (linear), s vom Typ 2 (sigmoidal).*



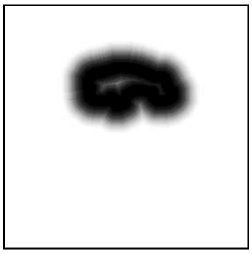
(a) $FOMCC(8, l)$



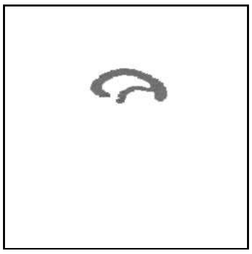
(b) $FOMCC(16, l)$



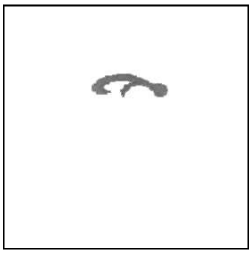
(c) $FOMCC(32, l)$



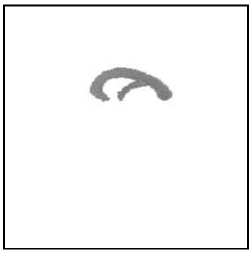
(d) $FOMCC(32, s)$



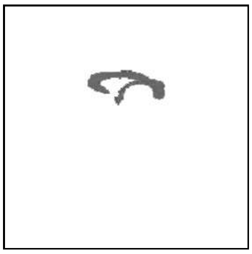
(e) $BspCC_1$



(f) $BspCC_2$



(g) $BspCC_3$



(h) $BspCC_4$

Abbildung B.11: Test von Fuzzy-Objekt-Modellen anhand des Beispiels Corpus Callosum. Die Zahl in der Klammer gibt den gewählten Wert für δ an, l ist eine Fuzzy-Menge vom Typ 1 (linear), s vom Typ 2 (sigmoidal).

FOM-Eigenschaft	Beispiel	Kleinhirn	Hirnstamm	Balken
$\delta = 8,\ \theta = 1$	Bsp_1	0.310	0.667	0.631
	Bsp_2	0.545	0.694	0.439
	Bsp_3	0.557	0.655	0.671
	Bsp_4	0.627	0.698	0.580
$\delta = 16,\ \theta = 1$	Bsp_1	0.616	0.847	0.796
	Bsp_2	0.769	0.851	0.706
	Bsp_3	0.780	0.839	0.812
	Bsp_4	0.808	0.863	0.765
$\delta = 32,\ \theta = 1$	Bsp_1	0.816	0.925	0.902
	Bsp_2	0.890	0.929	0.859
	Bsp_3	0.894	0.924	0.910
	Bsp_4	0.910	0.933	0.886
$\delta = 8,\ \theta = 2$	Bsp_1	0.290	0.682	0.667
	Bsp_2	0.553	0.710	0.431
	Bsp_3	0.569	0.647	0.698
	Bsp_4	0.629	0.694	0.608
$\delta = 16,\ \theta = 2$	Bsp_1	0.667	0.914	0.831
	Bsp_2	0.820	0.886	0.749
	Bsp_3	0.847	0.906	0.843
	Bsp_4	0.839	0.922	0.812
$\delta = 32,\ \theta = 2$	Bsp_1	0.902	0.984	0.957
	Bsp_2	0.957	0.976	0.933
	Bsp_3	0.965	0.980	0.952
	Bsp_4	0.961	0.983	0.949

Tabelle B.2: Beispiele für die Anwendung von Fuzzy-Objekt-Modellen. Gezeigt werden die Zugehörigkeitswerte der vier jeweiligen Beispielstrukturen zum entsprechenden FOM.

Bsp_1_In	0.020	0.020	0.980	0.020	0.980	0.980	0.500	0.980	0.500	0.500	0.020	0.500
Bsp_1_Out	0.617	0.617	0.852	0.617	0.852	0.734	0.734	0.734	0.734	0.852	0.617	0.852
Bsp_2_In	0.020	0.020	0.980	0.020	0.980	0.980	0.500	0.980	0.500	0.500	0.020	0.500
Bsp_2_Out	0.582	0.582	0.832	0.582	0.832	0.707	0.707	0.707	0.707	0.832	0.582	0.832
Bsp_3_In	0.039	0.039	0.961	0.039	0.961	0.961	0.500	0.961	0.500	0.500	0.039	0.500
Bsp_2_Out	0.613	0.613	0.840	0.613	0.840	0.727	0.727	0.727	0.727	0.840	0.613	0.840
Bsp_4_In	0.039	0.039	0.961	0.039	0.961	0.961	0.500	0.961	0.500	0.500	0.039	0.500
Bsp_4_Out	0.605	0.605	0.832	0.605	0.832	0.719	0.719	0.719	0.719	0.832	0.605	0.832
Bsp_5_In	0.059	0.059	0.941	0.059	0.941	0.941	0.500	0.941	0.500	0.500	0.059	0.500
Bsp_5_Out	0.609	0.609	0.828	0.609	0.828	0.719	0.719	0.719	0.719	0.828	0.609	0.828
Bsp_6_In	0.059	0.059	0.941	0.059	0.941	0.941	0.500	0.941	0.500	0.500	0.059	0.500
Bsp_6_Out	0.641	0.641	0.836	0.641	0.836	0.738	0.738	0.738	0.738	0.836	0.641	0.836
Bsp_7_In	0.078	0.078	0.922	0.078	0.922	0.922	0.500	0.922	0.500	0.500	0.078	0.500
Bsp_7_Out	0.605	0.605	0.816	0.605	0.816	0.711	0.711	0.711	0.711	0.816	0.605	0.816
Bsp_8_In	0.078	0.078	0.922	0.078	0.922	0.922	0.500	0.922	0.500	0.500	0.078	0.500
Bsp_8_Out	0.586	0.586	0.813	0.586	0.813	0.699	0.699	0.699	0.699	0.813	0.586	0.813
Bsp_9_In	0.098	0.098	0.902	0.098	0.902	0.902	0.500	0.902	0.500	0.500	0.098	0.500
Bsp_9_Out	0.598	0.598	0.793	0.598	0.793	0.695	0.695	0.695	0.695	0.793	0.598	0.793
Bsp_{10}_In	0.098	0.098	0.902	0.098	0.902	0.902	0.500	0.902	0.500	0.500	0.098	0.500
Bsp_{10}_Out	0.574	0.574	0.738	0.574	0.738	0.656	0.656	0.656	0.656	0.738	0.574	0.738
Bsp_{11}_In	0.117	0.117	0.883	0.117	0.883	0.883	0.500	0.883	0.500	0.500	0.117	0.500
Bsp_{11}_Out	0.594	0.594	0.781	0.594	0.781	0.688	0.688	0.688	0.688	0.781	0.594	0.781
Bsp_{12}_In	0.117	0.117	0.883	0.117	0.883	0.883	0.500	0.883	0.500	0.500	0.117	0.500
Bsp_{12}_Out	0.586	0.586	0.789	0.586	0.789	0.688	0.688	0.688	0.688	0.789	0.586	0.789
Bsp_{13}_In	0.137	0.137	0.863	0.137	0.863	0.863	0.500	0.863	0.500	0.500	0.137	0.500
Bsp_{13}_Out	0.590	0.590	0.770	0.590	0.770	0.680	0.680	0.680	0.680	0.770	0.590	0.770
Bsp_{14}_In	0.137	0.137	0.863	0.137	0.863	0.863	0.500	0.863	0.500	0.500	0.137	0.500
Bsp_{14}_Out	0.613	0.613	0.816	0.613	0.816	0.715	0.715	0.715	0.715	0.816	0.613	0.816
Bsp_{15}_In	0.156	0.156	0.844	0.156	0.844	0.844	0.500	0.844	0.500	0.500	0.156	0.500
Bsp_{15}_Out	0.586	0.586	0.758	0.586	0.758	0.672	0.672	0.672	0.672	0.758	0.586	0.758
Bsp_{16}_In	0.156	0.156	0.844	0.156	0.844	0.844	0.500	0.844	0.500	0.500	0.156	0.500
Bsp_{16}_Out	0.574	0.574	0.762	0.574	0.762	0.668	0.668	0.668	0.668	0.762	0.574	0.762
Bsp_{17}_In	0.176	0.176	0.824	0.176	0.824	0.824	0.500	0.824	0.500	0.500	0.176	0.500
Bsp_{17}_Out	0.578	0.578	0.734	0.578	0.734	0.656	0.656	0.656	0.656	0.734	0.578	0.734
Bsp_{18}_In	0.176	0.176	0.824	0.176	0.824	0.824	0.500	0.824	0.500	0.500	0.176	0.500
Bsp_{18}_Out	0.582	0.582	0.730	0.582	0.730	0.656	0.656	0.656	0.656	0.730	0.582	0.730

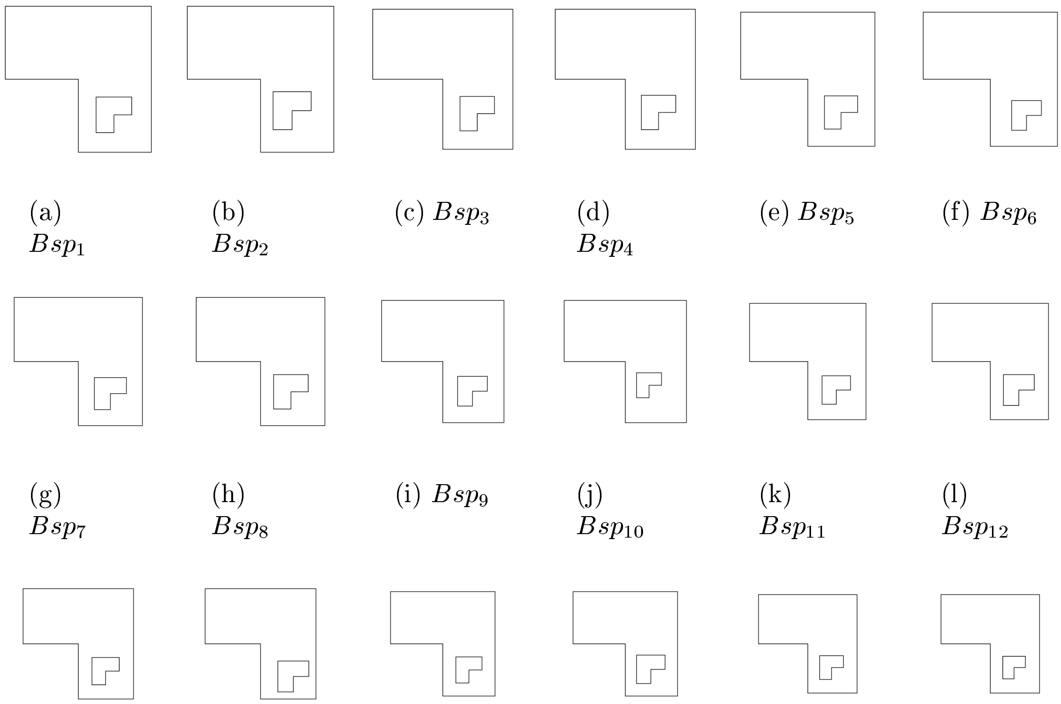
Tabelle B.3: Beispieltrainingsdaten für das neuronale Netz. Jede Struktur wird mit 6 Punkten repräsentiert.

B.2 Neuronale Netze

Die Anwendung neuronaler Netze soll an einer ähnlichen (aber sehr vereinfachten) Problemstellung gezeigt werden. Dazu werden dem Netz die Konturen von Beispielobjekten präsentiert. Derartige Objekte sind in Abbildung B.12 dargestellt. Das jeweils größere Objekt stellt die Eingabekontur dar, während das inliegende kleinere Objekt das gesuchte Objekt ist.

Grundsätzlich liegt das kleinere Objekt im Verhältnis immer im unteren rechten Abschnitt und hat 25% der Kantenlänge des größeren Objektes, wobei es zusätzlich um 90 Grad positiv gedreht ist. Die Hälfte der kleineren gesuchten Objekte besitzen zusätzlich eine Varianz bezüglich der Lage und der Größe. Beide Objektkonturen sind jeweils mit sechs Punkten definiert.

Das trainierte neuronale Netz besitzt 12 Eingabeneuronen und 12 Ausgabeneuronen. Dadurch können die Konturpunkte geeignet als Eingabe und Ausgabe verwendet werden. Zudem sind zwei verdeckte Schichten, ebenfalls jeweils mit 12 Neuronen, vorhanden. Das Netz ist voll verbunden, das heißt, daß jedes Neuron aus der Vorgängerschicht mit dem Neuron der Folgeschicht verbunden ist. Die Gewichte werden dabei mit Zusatzwerten aus dem Bereich $[-1, 1]$ initialisiert. Die Ergebnisse, die in Abbildung B.13 gezeigt werden, sind nach 3000 Lernzyklen erreicht. In dieser Abbildung werden 6 Beispieldaten präsentiert, die dem

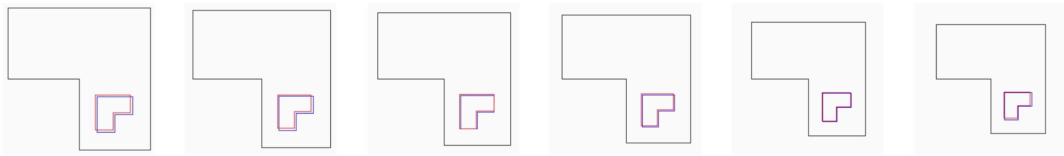


(a) Bsp_1 (b) Bsp_2 (c) Bsp_3 (d) Bsp_4 (e) Bsp_5 (f) Bsp_6

(g) Bsp_7 (h) Bsp_8 (i) Bsp_9 (j) Bsp_{10} (k) Bsp_{11} (l) Bsp_{12}

(m) Bsp_{13} (n) Bsp_{14} (o) Bsp_{15} (p) Bsp_{16} (q) Bsp_{17} (r) Bsp_{18}

Abbildung B.12: Beispieltrainingsdaten für das neuronale Netz.



(a) E_1 (b) E_2 (c) E_3 (d) E_4 (e) E_5 (f) E_6

Abbildung B.13: Ergebnisse nach der Anwendung des neuronalen Netzes. Die roten Objekte entsprechen den realen Daten, die blauen Objekte wurden vom Netz berechnet.

Netz nicht bekannt sind. Das rote Objekt stellt dabei den Idealfall des gesuchten Objektes dar. Dies bedeutet, daß es nach obiger Vorschrift konstruiert ist und keine Varianz besitzt. Das blau dargestellte Objekt entspricht der Objektkontur, die durch das Netz berechnet wird, wenn die Kontur des größeren (schwarz dargestellten) Objektes als Eingabe dient.

B.3 Evolutionsstrategien

B.3.1 Fuzzy-Mengen-Optimierung

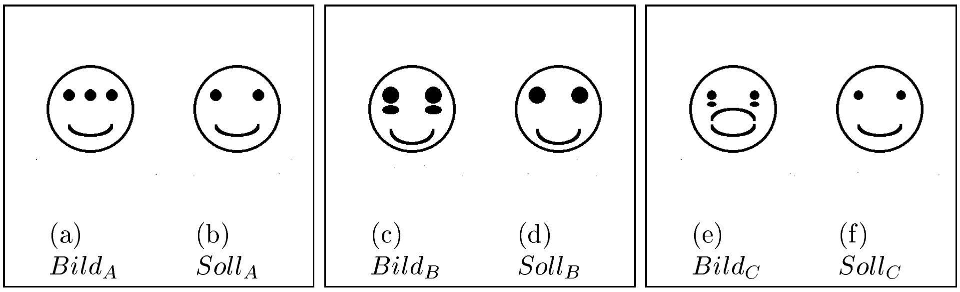
Exemplarisch soll hier die in Abschnitt 5.4.1 beschriebene Methode zur Optimierung von Fuzzy-Mengen mittels evolutionärer Algorithmen dargestellt werden.

Aufgabe ist die Erkennung des Objektes in einem Bild, welches folgende Eigenschaften besitzt:

1. Das gesuchte Objekt ist *sehr rund*.
2. Innerhalb des Objektes existieren drei Strukturen.
3. Die oberen Strukturen *liegen mittelweit* Bildpunkte auseinander, sind beide sehr rund und liegen ungefähr in der *oberen Mitte*. Sie sind sehr *klein*.
4. Die dritte Struktur *liegt unten*, hat einen Abstand von mittleren Bildpunkten von den oberen Strukturen, und ist *sehr länglich*.

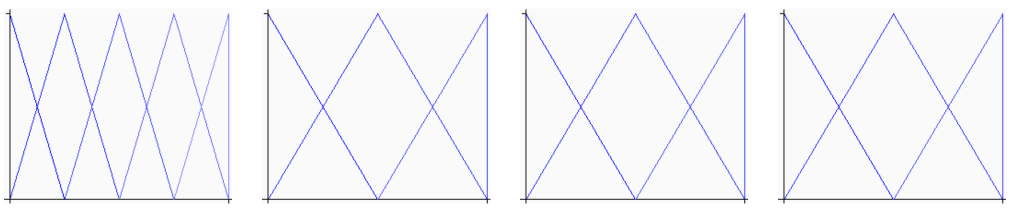
Die verwendete Wissensbasis ist in Abbildung B.18 dargestellt. Als manuell vom Experten segmentierte Objekte liegen die in Abbildung B.14 dargestellten Sollbilder und die Eigenschaften der einzelnen Segmente aus Tabelle B.4 vor. Diese sind jedoch als Sollvorgabe nicht einzeln bekannt. Zu optimieren sind die Interpretationen der Fuzzy-Mengen für die Lage, Größe, und Rundheit.

Die initialen Fuzzy-Mengen der verwendeten Variablen sind teilweise in Abbildung B.15 dargestellt und in Abbildung B.16 und Abbildung B.17 definiert. Der Einfachheit halber sind sie gleichmäßig verteilt.



(a) $Bild_A$ (b) $Soll_A$ (c) $Bild_B$ (d) $Soll_B$ (e) $Bild_C$ (f) $Soll_C$

Abbildung B.14: Darstellung der manuell segmentierten und als Referenz genutzten Beispielobjekte Soll zur Optimierung der hinterlegten Fuzzy-Mengen. Gezeigt werden jeweils die Eingabebilder und daneben die Bilder, die als Ergebnis geliefert werden.



(a) Größe (b) Rundheit (c) Horizontale Position (d) Vertikale Position

Abbildung B.15: Darstellung einiger initialen Fuzzy-Mengen vor der Optimierung für die verwendeten Variablen. Sie sind anfangs gleichmäßig über dem Universum verteilt. Der Definitionsbereich der Fuzzy-Variablen bestimmt die Beschriftung der Abszisse, die Fuzzy-Mengen sind normiert.

Dieses Beispiel sieht sehr einfach aus und ist es auch, damit die Übersichtlichkeit gewahrt bleibt. Trotzdem soll herausgestellt werden, daß nur die Gesamtvorgabe des gesuchten Objektes vom Experten vorliegt, die einzelnen Strukturen innerhalb des Objektes sind nicht bekannt, so daß hier nicht direkt deren Eigenschaften zur Optimierung der unscharfen Mengen verwendet werden können. In realen Systemen, etwa bei [RFH98], ist die Anzahl verwendeter Eigenschaften und zugehöriger Fuzzy-Mengen deutlich größer, so daß hier eine manuelle Optimierung der Mengen extrem aufwendig ist. In diesen Beispielbildern wurde die Größe der Augen, der Mundwinkel und die Lage der drei inneren Strukturen verändert. Für die Klassifikation von Objekten kann die Beschreibung aus Abbildung B.18 angegeben werden.

Die Fitneßfunktion eines Individuums V, also einer Belegung der Variablen aus Abbildung B.16 und Abbildung B.17 ist so definiert, daß die Anzahl der Bildpunkte, an denen sich Soll- und Ist-Bild unterscheiden, summiert wird (vgl.

Definition: Initiale Fuzzy-Variablen (1)

```
DEFINE_FUZZY_VARIABLE SMILEY.Size
UNIVERSE [0:3000] TYPE integer
FUNCTION Size
RELATION no
  TERM very_small = ( 1,      0,      0,      0,    750, no, no)
  TERM small      = ( 1,      0,    750,    750,   1500, no, no)
  TERM medium     = ( 1,    750,   1500,   1500,   2250, no, no)
  TERM big        = ( 1,   1500,   2250,   2250,   3000, no, no)
  TERM very_big   = ( 1,   2250,   3000,   3000,   3000, no, no)
DEFINE_FUZZY_VARIABLE_END

DEFINE_FUZZY_VARIABLE SMILEY.Roundness
UNIVERSE [0:1] TYPE real
FUNCTION RoundnessBeams
RELATION no
  TERM small  = ( 1,     0.0000,     0.0000,     0.0000,     0.5000, no, no)
  TERM medium = ( 1,     0.0000,     0.5000,     0.5000,     1.0000, no, no)
  TERM big    = ( 1,     0.5000,     1.0000,     1.0000,     1.0000, no, no)
DEFINE_FUZZY_VARIABLE_END

DEFINE_FUZZY_VARIABLE SMILEY.H_Position
UNIVERSE [0:255] TYPE integer
FUNCTION AbsoluteHorizontalPosition
RELATION no
  TERM left   = ( 1,     0,     0,     0,   128, no, no)
  TERM center = ( 1,     0,   128,   128,   255, no, no)
  TERM right  = ( 1,   128,   255,   255,   255, no, no)
DEFINE_FUZZY_VARIABLE_END

DEFINE_FUZZY_VARIABLE SMILEY.V_Position
UNIVERSE [0:255] TYPE integer
FUNCTION AbsoluteVerticalPosition
RELATION no
  TERM top    = ( 1,      0,      0,      0,    128, no, no)
  TERM center = ( 1,      0,    128,    128,    255, no, no)
  TERM bottom = ( 1,    128,    255,    255,    255, no, no)
DEFINE_FUZZY_VARIABLE_END
```

Abbildung B.16: Initiale Definition der verwendeten Fuzzy-Variablen durch eine über dem Universum gleichmäßige Verteilung der Terme.

Definition: Initiale Fuzzy-Variablen (2)

```
DEFINE_FUZZY_VARIABLE SMILEY.Contains
UNIVERSE [0:1] TYPE real
FUNCTION IsInside
RELATION yes
  TERM few    = ( 1,     0.0000,     0.0000,     0.0000,     0.5000, no, no)
  TERM medium = ( 1,     0.0000,     0.5000,     0.5000,     1.0000, no, no)
  TERM much   = ( 1,     0.5000,     1.0000,     1.0000,     1.0000, no, no)
DEFINE_FUZZY_VARIABLE_END

DEFINE_FUZZY_VARIABLE SMILEY.Is_Part_Of
UNIVERSE [0:1] TYPE real
FUNCTION PartOf
RELATION yes
  TERM few    = ( 1,     0.0000,     0.0000,     0.0000,     0.5000, no, no)
  TERM medium = ( 1,     0.0000,     0.5000,     0.5000,     1.0000, no, no)
  TERM much   = ( 1,     0.5000,     1.0000,     1.0000,     1.0000, no, no)
DEFINE_FUZZY_VARIABLE_END

DEFINE_FUZZY_VARIABLE SMILEY.Resides_totally_left_of
UNIVERSE [0:255] TYPE integer
FUNCTION TotallyLeftOf
RELATION yes
  TERM near    = ( 1,      0,      0,      0,    128, no, no)
  TERM medium  = ( 1,      0,    128,    128,    255, no, no
  TERM far     = ( 1,    128,    255,    255,    255, no, no)
DEFINE_FUZZY_VARIABLE_END

DEFINE_FUZZY_VARIABLE SMILEY.Resides_totally_right_of
UNIVERSE [0:255] TYPE integer
FUNCTION TotallyRightOf
RELATION yes
  TERM near    = ( 1,      0,      0,      0,    128, no, no)
  TERM medium  = ( 1,      0,    128,    128,    255, no, no
  TERM far     = ( 1,    128,    255,    255,    255, no, no)
DEFINE_FUZZY_VARIABLE_END

DEFINE_FUZZY_VARIABLE SMILEY.Resides_totally_top_of
UNIVERSE [0:255] TYPE integer
FUNCTION TotallyAboveOf
RELATION yes
  TERM near    = ( 1,      0,      0,      0,    128, no, no)
  TERM medium  = ( 1,      0,    128,    128,    255, no, no
  TERM far     = ( 1,    128,    255,    255,    255, no, no)
DEFINE_FUZZY_VARIABLE_END

DEFINE_FUZZY_VARIABLE SMILEY.Resides_totally_bottom_of
UNIVERSE [0:255] TYPE integer
FUNCTION TotallyBelowOf
RELATION yes
  TERM near    = ( 1,      0,      0,      0,    128, no, no)
  TERM medium  = ( 1,      0,    128,    128,    255, no, no
  TERM far     = ( 1,    128,    255,    255,    255, no, no)
DEFINE_FUZZY_VARIABLE_END
```

Abbildung B.17: Initiale Definition der verwendeten Fuzzy-Variablen durch eine über dem Universum gleichmäßige Verteilung der Terme.

Definition: Konzepte

```
DEFINE_CONCEPT SMILEY.Face
PRIO 0
  CONCEPT3D no
  CONCEPTVIEW undefined
  FEATURE SMILEY.Roundness = big   PRIO (0,0)
  FEATURE SMILEY.Size = big   PRIO (0,0)
  FEATURE IMAGE.V_Position = center   PRIO (0,0)
  FEATURE IMAGE.H_Position = center   PRIO (0,0)
  RELATION   SMILEY.Contains = SMILEY.Left_Eye,much  PRIO (0,0)
  RELATION   SMILEY.Contains = SMILEY.Right_Eye,much   PRIO (0,0)
  RELATION   SMILEY.Contains = SMILEY.Mouth,much  PRIO (0,0)
DEFINE_CONCEPT_END

DEFINE_CONCEPT SMILEY.Eye
PRIO 0
  CONCEPT3D no
  CONCEPTVIEW undefined
  FEATURE SMILEY.V_Position = top   PRIO (0,0)
  FEATURE SMILEY.Roundness = big    PRIO (0,0)
  FEATURE SMILEY.Size = small       PRIO (0,0)
  RELATION   SMILEY.Resides_totally_top_of = SMILEY.Mouth,medium   PRIO (0,0)
  RELATION   SMILEY.Is_Part_Of = SMILEY.Face,much   PRIO (0,0)
DEFINE_CONCEPT_END

DEFINE_CONCEPT SMILEY.Left_Eye IS_A SMILEY.Eye
PRIO 0
  CONCEPT3D no
  CONCEPTVIEW undefined
  FEATURE SMILEY.H_Position = left     PRIO (0,0)
  RELATION   SMILEY.Resides_totally_left_of = SMILEY.Right_Eye,medium   PRIO (0,0)
DEFINE_CONCEPT_END

DEFINE_CONCEPT SMILEY.Right_Eye IS_A SMILEY.Eye
PRIO 0
  CONCEPT3D no
  CONCEPTVIEW undefined
  FEATURE SMILEY.H_Position = right     PRIO (0,0)
  RELATION   SMILEY.Resides_totally_right_of = SMILEY.Left_Eye,medium   PRIO (0,0)
DEFINE_CONCEPT_END

DEFINE_CONCEPT SMILEY.Mouth
PRIO 0
  CONCEPT3D no
  CONCEPTVIEW undefined
  FEATURE SMILEY.V_Position = bottom PRIO (0,0)
  FEATURE SMILEY.H_Position = center PRIO (0,0)
  FEATURE SMILEY.Size = medium PRIO (0,0)
  FEATURE SMILEY.Roundness = small PRIO (0,0)
  RELATION   SMILEY.Resides_totally_below_of = SMILEY.Left_Eye,medium   PRIO (0,0)
  RELATION   SMILEY.Resides_totally_below_of = SMILEY.Right_Eye,medium   PRIO (0,0)
  RELATION   SMILEY.Is_Part_Of = SMILEY.Face,much   PRIO (0,0)
DEFINE_CONCEPT_END
```

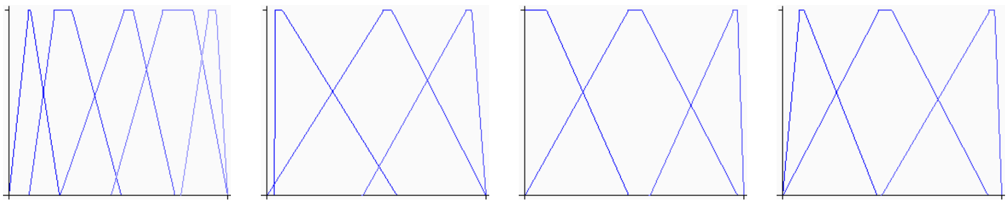
Abbildung B.18: Beschreibung der Konzepte der gesuchten Objekte.

Objekt	Bild *A*	Bild *B*	Bild *C*
Größe Kopf	2503	2503	2503
Rundheit Kopf	1.0	1.0	1.0
Größe Auge	201	375	147
Rundheit Auge	1.0	1.0	1.0
Schwerpunkt linkes Auge	(87,101)	(87,101)	(87,101)
Schwerpunkt rechtes Auge	(87,167)	(87,167)	(87,167)
Größe Mund	532	545	532
Rundheit Mund	0.60	0.62	0.60
Schwerpunkt Mund	(183,127)	(187,127)	(183,127)
Fehler Größe mittleres Auge	201	—	—
Fehler Schwerpunkt mittleres Auge	(87,127)	—	—
Fehler Größe Augenrand	—	205	83
Fehler Schwerpunkt Augenrand links	—	(121,101)	(113,101)
Fehler Schwerpunkt Augenrand rechts	—	(121,167)	(113,167)
Fehler Schwerpunkt oberer Mund	—	—	(139,127)

Tabelle B.4: Gemessene Eigenschaften der einzelnen Segmente.

Abschnitt 5.4.1). Dieser muß minimiert werden.

Abbildung B.19 zeigt die Fuzzy-Mengen nach 50 und nach 150 Generationen. μ ist 10, λ 30. Als Strategie wurde die Komma-Strategie gewählt. Schließlich zeigt Abbildung B.20 die finalen Ergebnisse nach 287 Generationen. Bei dieser Generation wurde das Terminierungskriterium erreicht.

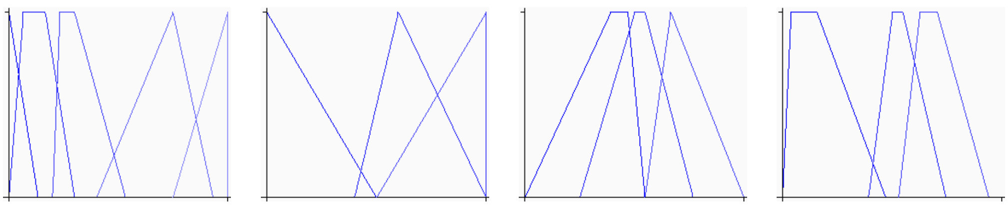


(a) Größe 50

(b) Rundheit 50

(c) Horizontale Position 50

(d) Vertikale Position 50



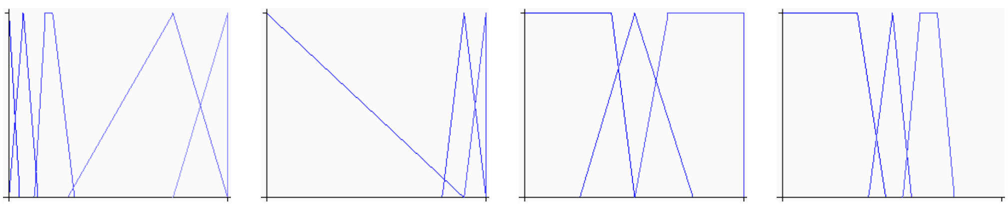
(e) Größe 150

(f) Rundheit 150

(g) Horizontale Position 150

(h) Vertikale Position 150

Abbildung B.19: Darstellung der Zwischenergebnisse bei der Optimierung der Fuzzy-Mengen vor der Optimierung nach 50 Generationen und nach 150 Generationen.



(a) Größe 287

(b) Rundheit 287

(c) Horizontale Position 287

(d) Vertikale Position 287

Abbildung B.20: Darstellung der Endergebnisse bei der Optimierung der Fuzzy-Mengen nach 287 Generationen, nachdem das Terminierungskriterium erreicht wurde.

C Befehlsübersicht von CIM^{2}BA/P

Im folgenden werden die im System CIM^{2}BA/P implementierten Befehle mit ihren Parametern beschrieben und größtenteils auch Beispiele gezeigt. Aus den Parametern können die entsprechenden Typen der Variablen abgelesen werden, bspw. *img* für den Typ *Bild* (vgl. Abschnitt C.2). Kann ein Befehl mehrere Typen als Parameter verarbeiten, wird stattdessen die allgemeine Bezeichnung *var* verwendet. Vordefinierte Konstanten werden mit *const* angegeben. Bei Typabfragen werden diese Konstanten als Ganzzahlen betrachtet. Werden mehrere Parameter des selben Typs verwendet, sind diese entsprechend ihrer Position durchnumeriert. Aufgrund des Umfangs wird auf die Angabe einer Grammatik verzichtet. Verwendet wird die Präfixnotation, wobei als Bezeichner für Variablen alle Symbole aus der Menge $\{A, \ldots Z, a, \ldots, z, 0, \ldots, 9\}$ mit der Einschränkung zugelassen sind, daß das erste Symbol keine Ziffer ist.

C.1 Sonderzeichen

`#`
Anfang eines Kommentars.

{
Beginn eines Blockes einer bedingten Anweisung.

}
Ende eines Blockes einer bedingten Anweisung.

`%`
Auflösung einer Variablen in einer Zeichenkette.

C.2 Datentypen

`INTEGER` $int_1, int_2, \ldots$
Definition von Ganzzahlen.

`BOOLEAN` $bool_1, bool_2, \ldots$
Definition von Booleschen Variablen.

`REAL` $real_1, real_2, \ldots$
Definition von Realzahlen.

`COMPLEX` $cpx_1, cpx_2, \ldots$
Definition von komplexen Zahlen.

`STRING` $str_1, str_2, \ldots$
Definition von Zeichenketten.

`POINT` $pt_1, pt_2, \ldots$
Definition von Punkten.

`POINTLIST` $pl_1, pl_2, \ldots$
Definition von Punktlisten.

`IMAGE` $img_1, img_2, \ldots$
Definition von Bilddatentypen.

`SEGMENT` $seg_1, var_2, \ldots$
Definition von Segmenten.

`SEGMENTLIST` $sl_1, sl_2, \ldots$
Definition von Segmentlisten.

`FUZZYSET` $fs_1, fs_2, \ldots$
Definition von Fuzzy-Mengen.

`FUZZYVAR` $fv_1, fv_2, \ldots$
Definition von Fuzzy-Variablen.

`FUZZYBASE` $fb_1, fb_2, \ldots$
Menge von Fuzzy-Variablen.

`FOM` $fom_1, fom_2, \ldots$
Definition von Fuzzy-Objekt-Modellen.

`FOMBASE` $fomb_1, fomb_2, \ldots$
Menge von Fuzzy-Objekt-Modellen.

`IFS` $ifs_1, ifs_2, \ldots$
Definition einer ikonischen Fuzzy-Menge.

`CONCEPT` $con_1, con_2, \ldots$
Definition von Konzepten.

`KNOWLEDGEBASE` $kb_1, kb_2, \ldots$
Menge von Konzepten.

`RULE` $rule_1, rule_2, \ldots$
Definition von Regeln.

`RULEBASE` $rb_1, rb_2, \ldots$
Menge von Regeln.

`NEURALNET` $ann_1, ann_2, \ldots$
Definition von neuronalen Netzen.

`CNN` $cnn_1, cnn_2, \ldots$
Definition von zellularen neuronalen Netzen.

Festlegung C.1 (Variablen) Statt der Verwendung von Variablen können im folgenden an den Stellen, wo kein Wert gesetzt wird, auch Konstanten (Zahlen, Texte) verwendet werden. So ist bspw. der Befehl `SET(Zaehler,1)` gleichbedeutend mit den Befehl `SET(Zaehler,Init)`, wenn die Variable *Init* den Wert 1 hat.

C.3 Allgemeine Befehle

`DEFINE_VARIABLES_BEGIN`
Parameter: keine
Zeigt den Beginn der Variablen-Definition an.

`DEFINE_VARIABLES_END`
Parameter: keine
Zeigt das Ende der Variablen-Definition an.

`PROGRAM_BEGIN`
Parameter: keine
Zeigt den Beginn des Programmteils an.

`PROGRAM_END`
Parameter: keine
Zeigt das Ende des Programmteils an.

`FUNCTION`
Parameter: *str*
Beginn einer Funktion mit dem Namen *str*.
Beispiel: `FUNCTION(BerechneRand)`

`RETURN`
Parameter: $[var|*]$
Ende einer Funktion und Rücksprung an die aufrufende Stelle. In die implizit definierte Variable $ReturnVal$ wird zunächst der Inhalt von *var* kopiert, der Parameter ist aber optional.
Beispiel: `RETURN(ErgBild)`

`CALL`
Parameter: *str*
Springt zu einer Funktion mit Namen *str*.
Beispiel: `CALL(Vorverarbeitung)`

`LABEL`
Parameter: *str*
Definiert eine Sprungadresse mit Namen *str*.
Beispiel: `LABEL(Segmentierung)`

`GOTO`
Parameter: *str*
Springt zu einer Sprungadresse mit Namen *str*.
Beispiel: `GOTO(Segmentierung)`

`DELETE`
Parameter: *var*
Löscht die Variable *var* und gibt den Speicher wieder frei.
Beispiel: `DELETE(BildTemp)`

`GETLFILENAME`
Parameter: str_1, str_2
Liest einen Dateinamen zum Laden einer Datei in die Zeichenkette str_1 ein, wobei als Verzeichnisfilter str_2 verwendet wird.
Beispiel: `GETLFILENAME(fName, BilderVerzeichnis)`

`GETSFILENAME`

Parameter: str_1, str_2
Liest einen Dateinamen zum Speichern einer Datei in die Zeichenkette str_1 ein, wobei als Verzeichnisfilter str_2 verwendet wird.
Beispiel: `GETSFILENAME(fName, BilderVerzeichnis)`

`SHOW`
Parameter: var
Zeigt den Inhalt der Variablen var an.
Beispiel: `SHOW(KopfBild)`

`SHOWAT`
Parameter: var, int_1, int_2
Zeigt den Inhalt der Variablen var an, wobei das Fenster an Position (int_1, int_2) gesetzt wird.
Beispiel: `SHOW(Kopf,100,100)`

`HIDE`
Parameter: var
Entfernt das jeweils letzte Fenster, welches den Inhalt der Variablen var anzeigt.
Beispiel: `HIDE(KopfBild)`

`SET`
Parameter: var_1, var_2
Setzt die Variable var_1 auf den Wert var_2.
Beispiel: `SET(Slides,128)`

`GET`
Parameter: var
Fragt einen Wert für die Variable var vom Benutzer ab.
Beispiel: `GET(AnzACuts)`

`GETAT`
Parameter: var, int_1, int_2
Fragt einen Wert für die Variable var vom Benutzer ab, wobei das Fenster an Position (int_1, int_2) gezeigt wird.
Beispiel: `GETAT(AnzACuts,100,100)`

`GETNAME`
Parameter: str, var
Fragt den Namen (Bezeichner) der Variablen var ab und schreibt das Ergebnis in str.
Beispiel: `GETNAME(name, AktuellesKonzept)`

`SETNAME`
Parameter: var, str
Setzt den Namen der Variablen var auf den Wert von str.
Beispiel: `SETNAME(AktuellesSegment, name)`

`WAIT`
Parameter: $[str|*]$
Wartet mit der Fortführung des Programms, bis das Fenster bestätigt wird, wobei die optionale Zeichenkette str als Meldung ausgegeben wird.
Beispiel: `WAIT(Fertig mit Segmentierung)`

`HALT`
Parameter: $[str|*]$
Wartet mit der Fortführung des Programms, bis das Fenster bestätigt wird, wobei die optionale Zeichenkette str als Meldung ausgegeben wird. Nach Betätigung einer Taste wird das Programm beendet.
Beispiel: `HALT(Fehler: Kein Bild geladen)`

`ADD`
Parameter: var_1, var_2, var_3
Addiert den Wert von var_3 zur Variablen var_2 hinzu und schreibt das Ergebnis in var_1. Zeichenketten werden aneinandergehängt.
Beispiel: `ADD(Zaehler,Zaehler,1)`

`SUB`
Parameter: var_1, var_2, var_3
Subtrahiert den Wert von var_3 von Variablen var_2 hinzu und schreibt das Ergebnis in var_1. Bei Zeichenketten wird ein Vorkommen von var_3 in var_2 gelöscht.
Beispiel: `SUB(Zaehler,Zaehler,1)`

`MUL`
Parameter: var_1, var_2, var_3
Multipliziert den Wert von var_2 mit der Variablen var_3 und schreibt das Ergebnis in var_1. Bei Zeichenketten wird var_2 genau $\lfloor var_3 \rfloor$ mal wiederholt.
Beispiel: `NeueGroesse,Groesse,2)`

`DIV`
Parameter: var_1, var_2, var_3
Dividiert den Wert von var_2 durch den Wert von var_3 und schreibt das Ergebnis in var_1.
Beispiel: `DIV(Mitte,Schichten,2)`

`MIN`

Parameter: var_1, var_2, var_3
Schreibt den kleineren der beiden Werte var_2 oder var_3 in var_1.
Beispiel: `MIN(minimum,ergebnis1,ergebnis2)`

`MAX`
Parameter: var_1, var_2, var_3
Schreibt den größeren der beiden Werte var_2 oder var_3 in var_1.
Beispiel: `MAX(maximum,ergebnis1,ergebnis2)`

`FOR`
Parameter: $var_1, var_2, var_3, var_4$
Beginnt eine Schleife mit der Zählervariablen var_1, die mit var_2 initialisiert wird, wobei als Abbruchkriterium das Erreichen von var_3 gilt und die Zählervariable um den Wert von var_4 inkrementiert wird. Der auszuführende Block danach wird in geschweiften Klammern gesetzt.
Beispiel: `FOR(Zaehler,1,AnzSchichten,1)`

`IFEMPTY`
Parameter: var
Prüft, ob die Variable leer bzw. für Zahlen gleich Null ist und führt im Wahrheitsfall den folgenden Block aus, ansonsten den ggf. existierenden `ELSE`-Teil.
Beispiel: `IFEMPTY(Bild){LOADIMAGE(Bild,Name)}`

`IFNOTEMPTY`
Parameter: var
Identisch mit `IFEMPTY` mit negierter Bedeutung.

`IFGREATERTHAN`
Parameter: var_1, var_2
Prüft, ob die Variable var_1 größer als die Variable var_2 ist und führt im Wahrheitsfall den folgenden Block aus, ansonsten den ggf. existierenden `ELSE`-Teil.
Beispiel: `IFGREATERTHAN(i,max) {GOTO(w)} ELSE {GOTO(z)}`

`IFGREATEREQUALTHAN`
Parameter: var_1, var_2
Identisch mit `IFGREATERTHAN` mit der Prüfung auf größer oder gleich ($\geq$).

`IFSMALLERTHAN`
Parameter: var_1, var_2
Identisch mit `IFGREATERTHAN` mit der Prüfung auf kleiner ($<$).

`IFSMALLEREQUALTHAN`
Parameter: var_1, var_2

Identisch mit IFGREATERTHAN mit der Prüfung auf kleiner oder gleich ($\leq$).

IFEQUAL
Parameter: var_1, var_2
Identisch mit IFGREATERTHAN mit der Prüfung auf Gleichheit ($=$).

IFNOTEQUAL
Parameter: var_1, var_2
Identisch mit IFGREATERTHAN mit der Prüfung auf Ungleichheit ($\neq$).

C.4 Befehle für Bilddaten

SHOW
Parameter: $img[*|, seg]$
Zeigt das Bild *img* an, wobei das Segment *seg* in das Bild eingeblendet wird. Die Angabe von *seg* darf auch fehlen.
Beispiel: `SHOW(Kopf,Hirn)`

LOADIMAGE
Parameter: img, str
Lädt einen Datensatz mit dem Dateinamen *str* in die Variable *img*.
Beispiel: `LOADIMAGE(Bild,MRTKopf.DCM)`

SAVEIMAGE
Parameter: str, img
Speichert ein Bild *str* mit dem Dateinamen *str* ab, wobei das Format durch das in *str* enthaltene Suffix bestimmt wird.
Beispiel: `SAVEIMAGE(MRTKopf_Segmentiert.DCM,Resultat)`

GETSIZEX
Parameter: int, img
Speichert die transversale Schichtanzahl vom Bild *img* in der Variablen *int*.
Beispiel: `GETSIZEX(Tiefe,Bild)`

GETSIZEY
Parameter: int, img
Identisch mit GETSIZEX für die coronare Schichtanzahl.

GETSIZEZ
Parameter: int, img
Identisch mit GETSIZEX für die sagittale Schichtanzahl.

`GETMETRICX`
Parameter: $real, img$
Speichert die Ausdehnung eines Bildpunktes in mm vom Bild img in der Variablen $real$ in transversaler Sichtweise.
Beispiel: `GETMETRICX(Tiefe,Bild)`

`GETMETRICY`
Parameter: $real, img$
Identisch mit `GETMETRICX` für die coronare Sichtweise.

`GETMETRICZ`
Parameter: $real, img$
Identisch mit `GETMETRICX` für die sagittale Sichtweise.

`SETSIZEX`
Parameter: img, int
Setzt die transversale Schichtanzahl vom Bild img.
Beispiel: `SETSIZEX(Kopf,AnzSchichtenBild)`

`SETSIZEY`
Parameter: img, int
Identisch mit `SETSIZEX` für die coronare Schichtanzahl.

`SETSIZEZ`
Parameter: img, int
Identisch mit `SETSIZEX` für die sagittale Schichtanzahl.

`GETMINCOLOR`
Parameter: int, img
Speichert den minimalen Grauwert vom Bild img in der Variablen int.
Beispiel: `GETMINCOLOR(GSMin,Bild)`

`GETMAXCOLOR`
Parameter: int, img
Identisch mit `GETMINCOLOR` für die maximale Graustufenanzahl.

`COPYIMAGE`
Parameter: img_1, img_2
Kopiert den Inhalt des Bildes img_2 nach img_1.
Beispiel: `COPYIMAGE(KopfKopie,Kopf)`

`COPYSLIDE`

Parameter: $img_1, img_2, const, int$
Kopiert die durch die Position *val* spezifizierte Schicht (in der Sichtweise *const*) von img_2 nach img_1.
Beispiel: `COPYSLIDE(Schicht,Kopf,SAGITTAL,mitte)`

`CONVERT`
Parameter: $img, seg, const$
Konvertiert ein Segment *seg* in ein Bild *img*, wobei abhängig von *const* nur die Punkte selbst oder auch deren Grauwert übernommen werden.
Beispiel: `CONVERT(KopfBild,HirnSegment,FALSE)`

`CONVERT`
Parameter: $seg, img, const$
Konvertiert ein Bild *img* in ein Segment *seg*, wobei abhängig von *const* nur die Punkte selbst oder auch deren Grauwert übernommen werden.
Beispiel: `CONVERT(HirnSegment,HirnBeispielBild,FALSE)`

`BINARIZE`
Parameter: img_1, img_2, int
Das Bild img_2 wird mit den Schwellenwert *int* binarisiert und das Ergebnis in img_1 geschrieben.
Beispiel: `BINARIZE(BinBild,Kopf,HistogrammMinimum)`

`GETCONTOUR`
Parameter: pl, img, int
Berechnet die Kontur des in *img* dargestellten Objektes und speichert die Punkte in *pl*, wobei maximal *int* Punkte erlaubt sind. Sind mehr Punkte auf der Kontur vorhanden, wird eine äquidistante Auswahl getroffen.
Beispiel: `GETCONTOUR(KonturListe,KopfBild,30)`

`MULTITHRESHOLD`
Parameter: $img_1, img_2, int_1, int_2, \ldots, int_n$
Das Bild img_2 wird mit dem Schwellenwerten int_i in $n+1$ Grauwerte segmentiert und das Ergebnis in img_1 geschrieben.
Beispiel: `MULTITHRESHOLD(SegmentBild,Kopf,Hintergrund,Kopf,Hirn)`

`FLOODFILL`
Parameter: $seg, img, int_1, int_2, int_3, int_4, int_5$
Anwendung eines einfachen Bereichswachstumsverfahrens auf das Bild *img* mit der Startposition (int_1, int_2, int_3) und einer unteren Grauwertabweichung von int_4 und einer oberen int_5. Die Punkte werden in eimem Segment *seg* gespeichert.
Beispiel: `FLOODFILL(Region,Kopf,centerx,centery,centerz,20,20)`

`FLOODFILLREGION`
Parameter: $seg, img, int_1, int_2, int_3, int_4, int_4, int_6, int_7, int_8, int_9, int_{10}, int_{11}$
Anwendung eines einfachen Bereichswachstumsverfahrens mit der Startposition (int_1, int_2, int_3) und einer unteren Grauwertabweichung von int_{10} und einer oberen int_{11}. Der Suchbereich wird dabei auf die durch die Punkte (int_4, int_5, int_6) und (int_7, int_8, int_9) beschriebene Region beschränkt. Das Ergebnis wird in *seg* geschrieben.
Beispiel: `FLOODFILLREGION(Region,Kopf,x,y,z,30,30,20,120,120,80,20,20)`

`INVERT`
Parameter: img_1, img_2
Das Bild img_2 wird invertiert in img_1 geschrieben.
Beispiel: `INVERT(Negativ, Kopf)`

`GRAYFILTER`
Parameter: $img_1, img_2, int_1, int_2$
Setzt die Grauwerte im Bild img_2 auf 0, die kleiner int_1 oder größer als int_2 sind und schreibt das Ergebnis in img_1.
Beispiel: `GRAYFILTER(Gefiltert,Kopf,HirnMin,HirnMax)`

`PREWITTH`
Parameter: $img_1, img_2, const$
Wendet den horizontalen Prewitt-Operator auf Bild img_2 (jede Schicht in Sichtweise *const*) an und schreibt das Ergebnis nach img_1.
Beispiel: `PREWITTH(Kanten,Kopf,SAGITTAL)`

`PREWITTV`
Parameter: $img_1, img_2, const$
Identisch mit `PREWITTH` für den vertikalen Operator.

`SOBELH`
Parameter: $img_1, img_2, const$
Wendet den horizontalen Sobel-Operator auf Bild img_2 (jede Schicht in Sichtweise *const*) an und schreibt das Ergebnis nach img_1.
Beispiel: `SOBELH(Kanten,Kopf,SAGITTAL)`

`SOBELV`
Parameter: $img_1, img_2, const$
Identisch mit `SOBELH` für den vertikalen Operator.

`LAPLACE4`
Parameter: $img_1, img_2, const$

Wendet den N_4-Laplace-Operator Bild img_2 (jede Schicht in Sichtweise $const$) an und schreibt das Ergebnis nach img_1.
Beispiel: `LAPLACE4(Kanten,Kopf,SAGITTAL)`

`LAPLACE8`
Parameter: $img_1, img_2, const$
Identisch mit `LAPLACE4` für den N_8-Operator.

`EDGEFILTER3x3`
Parameter: $img_1, img_2, const, real_1, \ldots, real_9$
Wendet eine 3×3-Maske, die in $real_1, \ldots, real_9$ definiert ist, auf das Bild img_2 an und schreibt das Ergebnis in img_1 (Sichtweise $const$).
Beispiel: `EDGEFILTER3x3(Kanten,Kopf,SAGITTAL,-1,-2,-1,0,0,0,1,2,1)`

`ENHANCECONTRAST`
Parameter: img_1, img_2
Erhöht den Kontrast in img_2, indem der Grauwertbereich mittels Sigmoidfunktion über den minimalen bis maximalen Grauwert skaliert wird, das Ergebnis wird in img_1 geschrieben.
Beispiel: `ENHANCECONTRAST(KopfNeu,Kopf)`

`ENHANCECONTRASTPARAM`
Parameter: $img_1, img_2, const, int_1, int_2$
Der gleiche Befehl wie `ENHANCECONTRAST`, wobei allerdings als Skalierungsfunktion zwischen den Konstanten *linear* oder *sigmoidal* gewählt werden kann und die Funktion über den Bereich von $[int_1, int_2]$ (statt über den gesamten Grauwertbereich) verwendet wird.
Beispiel: `ENHANCECONTRASTPARAM(KopfNeu,Kopf,LINEAR,64,128)`

`HISTOGRAM`
Parameter: img_1, img_2
Berechnet das Histogramm von img_2 und speichert das Ergebnis in img_1.
Beispiel: `HISTOGRAM(Histo,Kopf)`

`HISTOGRAMEQUALIZE`
Parameter: img_1, img_2
Berechnet für Bild img_2 das Histogramm so, daß die Grauwerte (approxmiert) gleichverteilt werden und schreibt das Ergebnis in img_1.
Beispiel: `HISTOGRAMEQUALIZE(Gleichverteilt,Kopf)`

`HISTOGRAMMINIMUM`
Parameter: int, img
Berechnet das Histogramm von img und speichert das erste (lokale) Minumum

in *int* ab.
Beispiel: `HISTOGRAMMINIMUM(MinGW,Kopf)`

`DIFFIMAGE`
Parameter: img_1, img_2, img_3
Berechnet die Unterschiede in Bild img_2 zu img_3 und schreibt das Ergebnis in img_1.
Beispiel: `DIFFIMAGE(Ergebnis,SollBild,Istbild)`

`COUNTPIXEL`
Parameter: int_1, img, int_2
Zählt die Bildpunkte im Bild img, die nicht der Hintergrundfarbe int_2 entsprechen und schreibt das Ergebnis in int_1. Der Befehl kann zur Fitneßberechnung verwendet werden.
Beispiel: `COUNTPIXEL(diff,ErgebnisBild,0)`

`COMBINEIMAGE`
Parameter: $img_1, img_2, img_3, const$
Kombiniert den Inhalt von Bild img_2 und img_3 und schreibt das Ergebnis in img_1. *const* gibt dabei an, ob der Hintergrund dunkel ist (const = TRUE); entsprechend werden die Bilder mit max oder min verknüpft.
Beispiel: `COMBINEIMAGE(Gesamt,Kleinhirn,Hirnstamm,TRUE)`

`SMOOTHLINEAR`
Parameter: $img_1, img_2, const$
Glättet das Bild img_2 und schreibt das Ergebnis in img_1, wobei eine lineare Glättungsmaske 3×3 verwendet wird. Die Schichtrichtung der Maske kann in *const* angegeben werden.
Beispiel: `SMOOTHLINEAR(KopfGeglaettet,Kopf,sagittal)`

`SMOOTHGAUSS`
Parameter: $img_1, img_2, const$
Identisch mit `SMOOTHLINEAR` mit einer Gauss-Funktion als Glättungsfunktion.

`SMOOTH3x3`
Parameter: $img_1, img_2, const, real_1, \ldots, real_9$
Identisch mit `SMOOTHLINEAR` mit einer frei definierbaren 3×3 Maske als Glättungsfunktion.
Beispiel: `SMOOTH3x3(Geglaettet,Kopf,sagittal,1,1,1,1,2,1,1,1,1)`

`SMOOTHMXNLINEAR`
Parameter: $img_1, img_2, const, int_1, int_2$
Identisch mit `SMOOTHLINEAR`, allerdings kann die Größe der Glättungsmaske mit

$int_1 \times int_2$ frei angegeben werden.
Beispiel: `SMOOTHMxNLINEAR(Geglaettet,Kopf,sagittal,5,5)`

`MEDIAN`
Parameter: $img_1, img_2, const$
Wendet den Medianfilter der Größe $3 times 3$ auf img_2 an und schreibt das Ergebnis in img_1. Die Sichtrichtung der Maske wird in $const$ angegeben.
Beispiel: `MEDIAN(Geglaettet,Kopf,sagittal)`

`MEDIAMMXN`
Parameter: $img_1, img_2, const, int_1, int_2$
Identisch mit `MEDIAN`, wobei die Maske eine Größe von $int_1 \times int_2$ hat.
Beispiel: `MEDIAN(Geglaettet,Kopf,sagittal,5,5)`

`GETBORDERS`
Parameter: $int_1, int_2, int_3, int_4, int_5, int_6, img, int_7$
Sucht die Grenzen des Bildes img und schreibt das Ergebnis in die entsprechenden Variablen int_1 bis int_6. Als Schwellenwert wird dabei int_7 verwendet.
Beispiel: `GETBORDERS(minx,maxx,miny,maxy,minz,maxz,Kopf,30)`

`SCALEIMAGE`
Parameter: $img_1, img_2, int_1, int_2, int_3, int_4, int_5, int_6, int_7, int_8, int_9$
Skaliert den in $[int_1, int_2][int_3, int_4][int_5, int_6]$ definierten Auschnitt des Bildes img_2 auf die Größe $[int_7, int_8, int_9]$ und schreibt das Ergebnis in img_1. Entsprechendene Größenänderungen der Bildpunkte werden in den dazugerhörigen Eingenschaften des Bildes d_x, d_y, d_z gespeichert (vgl. Festlegung 5.1).
Beispiel: `SCALEIMAGE(Norm,Kopf,minx,maxx,miny,maxy,minz,maxz,256,256,128)`

`CLIPIMAGE`
Parameter: $img_1, img_2, pl, const$
Verwendet das in pl beschriebene Polygon als Maske auf img_2 und schreibt die abhängig von $const$ innerhalb oder außerhalb des Polygon liegenden Punkte von img_2 in img_1.
Beispiel: `CLIPIMAGE(ROI,Bild,ROIKontur)`

`SCALEGRAYSCALES`
Parameter: $img_1, img_2, int_1, int_2$
Der verwendete Grauwertbereich des Bildes img_2 wird auf den Bereich von int_1 bis int_2 skaliert und das Ergebnis in img_1 gespeichert.
Beispiel: `SCALGRAYSCALES(NormGV,Kopf,0,255)`

`IFSFILTER`
Parameter: img_1, img_2, ifs

Wendet die ikonische Fuzzy-Menge ifs als Ortsfilter auf img_2 an und schreibt das Ergebnis in img_1.
Beispiel: `IFSFILTER(ROIKleinhirn,Kopf,IFSKleinhirn)`

`FSFILTER`
Parameter: img_1, img_2, fs
Wendet die Fuzzy-Menge fs als Grauwertfilter auf img_2 an und schreibt das Ergebnis in img_1.
Beispiel: `FSFILTER(KleinHirnGrauwerte,Kopf,FSKleinhirn)`

`SEGMENTATE`
Parameter: sl, img, con, fb, int
Das Bild img wird in Bezug auf die in con angegenen Eigenschaften zerlegt und die Menge der gefundenen Segmente in die Segmentliste sl geschrieben. Als Auflösung für die Unschärfegrade wird der Wert int verwendet, als Interpretation der Terme fb.
Beispiel: `SEGMENTATE(SListe,Kopf,KleinhirnKonzept,MRTFMengen,64)`

C.5 Befehle für Segmente und Punkte

`GETSIZEX`
Parameter: int, seg
Speichert die transversale Schichtanzahl vom Segment seg in der Variablen int.
Beispiel: `GETSIZEX(Tiefe,SegKopf)`

`GETSIZEY`
Parameter: int, seg
Identisch mit `GETSIZEX` für die coronare Schichtanzahl.

`GETSIZEZ`
Parameter: int, seg
Identisch mit `GETSIZEX` für die sagittale Schichtanzahl.

`SETSIZEX`
Parameter: seg, int
Setzt die transversale Schichtanzahl vom Segment seg.
Beispiel: `SETSIZEX(SegKopf,AnzSchichtenBild)`

`SETSIZEY`
Parameter: seg, int
Identisch mit `SETSIZEX` für die coronare Schichtanzahl.

`SETSIZEZ`
Parameter: *seg*, *int*
Identisch mit `SETSIZEX` für die sagittale Schichtanzahl.

`GETFEATUREVALUE`
Parameter: *real*, *seg*, *const*
Schreibt den Wert der Eigenschaft *const* des Segmentes *seg* in die Variable *real*. Als Eigenschaften sind alle Funktionen für Segmente möglich, wie sie auch in Abschnitt 6.1.2.2 verwendet werden.
Beispiel: `GETFEATUREVALUE(Groesse,Segment,SIZE)`

`ADDLIST`
Parameter: *sl*, *seg*
Hängt das Element *seg* an die Segmentliste *sl* an.
Beispiel: `ADDLIST(SegmentListe,NeuesSegment)`

`GETBESTSEGMENT`
Parameter: *seg*, *sl*
Sucht das beste Segment (abhängig vom `COMPARE`-Wert, (vgl. Abschnitt 6.1.2.7)) aus der Liste *sl* und schreibt das Ergebnis in *seg*.
Beispiel: `GETBESTSEGMENT(BestSegment,SegmentListe)`

`GETSEGMENT`
Parameter: *seg*, *sl*, *int*
Sucht das Segment Nummer *int* aus der Liste *sl* und schreibt das Ergebnis in *seg*.
Beispiel: `GETSEGMENT(Segment,SegmentListe,i)`

`GETCOMPAREVALUE`
Parameter: *real*, *seg*
Schreibt das Ergebnis der `COMPARE`-Berechnung des Segmentes *seg* in *real*.
Beispiel: `GETCOMPAREVALUE(result,Segment)`

`UPDATELIST`
Parameter: *sl*, *seg*, *int*
Ersetzt das Element an der Position *int* in der Segmentliste *sl* durch das Element *seg*.
Beispiel: `UPDATELIST(SegmentListe,BestSegment,pos)`

`GETPOINT`
Parameter: *pt*, *pl*, *int*
Liest den Punkt *int* aus der Liste *pl* und schreibt das Ergebnis in *var*.

Beispiel: `GETPOINT(Punkt,PunktListe,i)`

`GETPOINTX`
Parameter: var, pl, int
Liest die X-Koordinate des Punktes int aus der Liste pl und schreibt das Ergebnis in var.
Beispiel: `GETPOINTX(xPos,PunktListe,i)`

`GETPOINTY`
Parameter: var, pl, int
Identisch mit GETPOINTX für die Y-Koordinate des Punktes.

`GETPOINTZ`
Parameter: var, pl, int
Identisch mit GETPOINTX für die Z-Koordinate des Punktes.

`SETPOINTLIST`
Parameter: pl, pt, int
Schreibt den Punkt pt an die Position int in die Liste pl.
Beispiel: `SETPOINTLIST(PunktListe,Punkt,i)`

`SETPOINTLISTX`
Parameter: pl, var, int
Schreibt den Wert var in die X-Koordinate des Punktes an Position int aus der Liste pl.
Beispiel: `SETPOINTLISTX(Punktliste,,i)`

`SETPOINTLISTY`
Parameter: pl, var, int
Identisch mit SETPOINTLISTX für die Y-Koordinate des Punktes.

`SETPOINTLISTZ`
Parameter: pl, var, int
Identisch mit SETPOINTLISTX für die Z-Koordinate des Punktes.

`SPLIT`
Parameter: sl, seg, int
Das Segment seg wird an allen Stegen, die kleiner als int Punkte breit sind, aufgetrennt. Danach werden alle neu entstehenden Segmente in sl gespeichert.
Beispiel: `SPLIT(NeueListe,Segment,3)`

`MERGE`
Parameter: sl_1, sl_2

Es werden Kombinationen (maximal 2^n bei n Segmenten in sl_2) von Segmenten aus sl_2 gebildet, die mindestens einen Punkt gemeinsam haben und als neues Segment in sl_1 gespeichert.
Beispiel: `MERGE(Kombinationen,SegListe)`

`DEFINEROUNDNESSSAMPLES`
Parameter: $int, const$
Legt für die Eigenschaft $roundness_R$ die Anzahl der Strahlen fest, als Startwinkel wird dabei die Hauptachse gewählt, falls $const = TRUE$ gilt, sonst der Null-Grad Winkel bzgl. des gewählten Koordinantensystems.
Beispiel: `DEFINEROUNDNESSSAMPLES(30,TRUE)`

`USEMETRICMEASURES`
Parameter: $const$
Legt fest, ob die metrischen Maße verwendet werden oder mit einheitlichen Kantenlägen der Bildpunkte gerechnet wird.
Beispiel: `USEMETRICMEASURES(yes)`

`ORIENTATE`
Parameter: seg, int
Orientiert die Hauptachse des Segments seg auf int Grad aus.
Beispiel: `ORIENTATE(Hirnstamm,0)`

C.6 Befehle für Konzepte und Regeln

`COMPARE`
Parameter: con, seg, fb, sl
Berechnet einen Entsprechungswert des Segmentes seg zum Konzept con, wobei als Interpretation die in fb angegebenen Fuzzy-Mengen verwendet werden, die Liste bereits gefundener Segmente wird durch sl angegeben.
Beispiel: `COMPARE(AktKonzept,NeuesSegment,MRTKopfFMengen,Gefunden)`

`GETCONCEPT`
Parameter: con, kb, int
Schreibt das Konzept mit Index int aus der Wissensbasis kb in die Variable con.
Beispiel: `GETCONCEPT(Konzept,WBasis,pos)`

`GETRULE`
Parameter: $rule, rb, int$
Schreibt die Regel mit Index int aus der Regelbasis rb in die Variable $rule$.
Beispiel: `GETRULE(Regel,RBasis,pos)`

`LOADKNOWLEDGEBASE`
Parameter: kb, str
Läd die in *str* angebene Datei als Wissensbasis in die Variable *kb*.
Beispiel: `LOADKNOWLEDGEBASE(KBasis,MRT_KBase.kbase)`

`LOADRULEBASE`
Parameter: rb, str
Läd die in *str* angebene Datei als Regelmengen-Basis in die Variable *rb*.
Beispiel: `LOADRULEBASE(FRegeln,MRT_FRegeln.rbase)`

`APPLYRULEBASE`
Parameter: con, rb, seg
Betrachtet, ob das Segment *seg* die in der Regelbasis *rb* hinterlegten Regeln erfüllt und führt entsprechende Anpassungen im Konzept *con* aus.
Beispiel: `APPLYRULEBASE(Kleinhirn,ErweitereSuchbereichsRB,BestSegment)`

C.7 Befehle für Fuzzy-Mengen

`SET`
Parameter: $fs, paramlist$
Setzt die Eigenschaften einer Fuzzy-Menge fs auf die entsprechende Anzahl Parameter (Dargestellt durch *paramlist*), wobei die in Festlegung 6.1 getroffenen Regeln gelten.
Beispiel: `SET(Groesse,linear,0,0.5,0.5,1,no,yes)`

`SET`
Parameter: $ifs, int_1, int_2, int_3, fs_1, fs_2, fs_3, const$
Setzt die Parameter einer ikonischen Fuzzy-Menge auf die Größe ($int_1 \times int_2 \times int_3$) und verknüft die zylindrischen Erweiterungen der Fuzzy-Mengen fs_1, fs_2 und fs_3 mit Hilfe der in *const* angegebenen Norm.
Beispiel: `SET(Ortsfilter,256,256,1,HorizontalFS,Vertikal,NULL,MIN)`

`SETCORE`
Parameter: $fs, real_1, real_2$
Setzt den Kern einer Fuzzy-Menge fs. Ist die Fuzzy-Menge fs nicht normal, so wird sie normalisiert.
Beispiel: `SETCORE(Groesse,0.4,0.6)`

`SETMAX`
Parameter: $fs, real_1, real_2$

Setzt den Maximalbereich (also den Kern bei einer Höhe 1) einer Fuzzy-Menge fs.
Beispiel: `SETMAX(Groesse,0.4,0.6)`

`SETSUPPORT`
Parameter: $fs, real_1, real_2$
Setzt den Träger einer Fuzzy-Menge fs.
Beispiel: `SETSUPPORT(Groesse,0.2,0.8)`

`SETHEIGHT`
Parameter: $fs, real$
Setzt die Höhe einer Fuzzy-Menge fs.
Beispiel: `SETHEIGHT(Groesse,0.9)`

`SETHEIGHT`
Parameter: $fs, real_1, real_2, real_3$
Setzt die Höhen einer Fuzzy-Menge fs.
Beispiel: `SETHEIGHT(Groesse,0,1,0)`

`SETTYPE`
Parameter: $fs, const$
Setzt den Typ einer Fuzzy-Menge fs auf $const$.
Beispiel: `SETTYPE(Groesse,LINEAR)`

`GETCORE`
Parameter: $real_1, real_2, fs$
Schreibt den Kernbereich einer Fuzzy-Menge fs in die Variablen $real_1$ und $real_2$. Falls die Höhe nicht gleich 1 ist, entspricht dieser Befehl `GETMAX`.
Beispiel: `GETCORE(links,rechts,Groesse)`

`GETMAX`
Parameter: $real_1, real_2, fs$
Schreibt den Maximalbereich einer Fuzzy-Menge fs in die Variablen $real_1$ und $real_2$.Falls die Höhe gleich 1 ist, entspricht dieser Befehl `GETCORE`.
Beispiel: `GETMAX(links,rechts,Groesse)`

`GETSUPPORT`
Parameter: $real_1, real_2, fs$
Schreibt den Trägerbereich einer Fuzzy-Menge fs in die Variablen $real_1$ und $real_2$.
Beispiel: `GETSUPPORT(left,right,Groesse)`

`GETTYPE`

Parameter: int, fs
Schreibt den Typ einer Fuzzy-Menge fs in die Variable int.
Beispiel: `GETTYPE(Typ,Groesse)`

`GETDEGREE`
Parameter: $real_1, fs, real_2$
Berechnet den Zugehörigkeitswert der Variablen fs an der Stelle $real_2$ und schreibt das Ergebnis in die Variable $real_1$.
Beispiel: `GETDEGREE(Ergebnis,Groesse,0.5)`

`SETIFSDEGREE`
Parameter: $ifs, int_1, int_2, int_3, real$
Setzt den Zugehörigkeitswert an der Stelle (int_1, int_2, int_3) in der ikonischen Fuzzy-Menge ifs auf den Wert $real$.
Beispiel: `SETIFSDEGREE(Ortsfilter,128,128,1,1.0))`

`GETIFSDEGREE`
Parameter: $real, ifs, int_1, int_2, int_3$
Liest den Zugehörigkeitswert an der Stelle (int_1, int_2, int_3) in der ikonischen Fuzzy-Menge ifs und schreibt das Ergebnis in $real$.
Beispiel: `GETIFSDEGREE(MuIFS,Ortsfilter,128,128,1))`

`SELECTNORM`
Parameter: $const_1, const_2$
Verwendet für die Berechnung von Zugehörigkeitswerten die in $const_1$ bzw. $const_2$ angegebene Norm.
Beispiel: `SELECTNORM(MIN,MAX)`

`TNORM`
Parameter: fs_1, fs_2, fs_3
Verknüpft die Fuzzy-Mengen fs_2 und fs_3 entsprechend der aktuell gewählten T-Norm τ und schreibt das Ergebnis in fs_1. Die Fuzzy-Mengen dürfen auch ikonische Fuzzy-Mengen sein.
Beispiel: `TNORM(Lage,Vertikal,Horizontal)`

`SNORM`
Parameter: fs_1, fs_2, fs_3
Analog zu `TNORM` mit einer S-Norm σ als Verknüpfung.

`SELECTDEFUZZYMODE`
Parameter: $const$
Legt die Methode der Defuzzifinzierung fest, die angewandt wird.
Beispiel: `SELECTDEFUZZYMODE(COG)`

`LOADFUZZYSETS`
Parameter: fb, str
Läd die in str angebene Datei als Fuzzy-Mengen-Basis in die Variable fb.
Beispiel: `LOADFUZZYSETS(FMengen,MRT_FMengen.fset)`

`LOADFOMBASE`
Parameter: $fomb, str$
Läd die in str angebene Datei als FOM-Basis in die Variable $fomb$.
Beispiel: `LOADFOMBASE(FomMengen,MRT_FMengen.foms)`

C.8 Befehle für neuronale Netze

`SETNET`
Parameter: $ann, int_1, int_2, real$
Der Eingabewert des Neurons int_2 in der Schicht int_1 des Netzes ann wird auf den Wert $real$ gesetzt.
Beispiel: `SETNEURON(Netz,1,1,0.5)`

`SETNET`
Parameter: $ann, str, real$
Der Eingabewert des Neurons, welches mit str im Netz ann benannt ist, wird auf den Wert $real$ gesetzt.
Beispiel: `SETNEURON(Netz,Groesse,0.5)`

`GETNET`
Parameter: $real, ann, int_1, int_2$
Der Ausgabewert des Neurons int_2 in Schicht int_1 des Netzes ann wird in $real$ geschrieben.
Beispiel: `GETNET(Ausgabe,Netz,1,1)`

`GETNET`
Parameter: $real, ann, str$
Der Ausgabewert des Neurons, welches mit str im Netz ann benannt ist, wird in $real$ geschrieben.
Beispiel: `GETNET(Ausgabe,Netz,Struktur)`

`SETWEIGHT`
Parameter: $ann, int_1, int_2, int_3, int_4, real$
Das Gewicht der Verbindung von int_2 der Schicht int_1 zum Neuron int_4 der Schicht int_3 des Netzes ann wird auf den Wert $real$ gesetzt.

Beispiel: `SETWEIGHT(Netz,1,1,2,1,0.5)`

`SETWEIGHT`
Parameter: $ann, str_1, str_2, real$
Das Gewicht der Verbindung von Neuron mstr_1 zum Neuron str_2 des Netzes ann wird auf den Wert $real$ gesetzt.
Beispiel: `SETWEIGHT(Netz,Groesse,Hidden_1,0.5)`

`GETWEIGHT`
Parameter: $real, ann, int_1, int_2, int_3, int_4$
Das Gewicht der Verbindung von Neuron int_2 der Schicht int_1 zum Neuron int_4 der Schicht int_3 des Netzes ann wird in $real$ geschrieben.
Beispiel: `GETWEIGHT(gew,Netz,1,1,2,1)`

`GETWEIGHT`
Parameter: $ann, str_1, str_2, real$
Das Gewicht der Verbindung von Neuron str_1 zum Neuron str_2 des Netzes ann wird auf in ann geschrieben.
Beispiel: `GETWEIGHT(gew,Netz,Groesse,Hidden_1)`

`SETNEURONTHETA`
Parameter: $ann, int_1, int_2, real$
Setzt den Schwellenwert des Neurons int_2 in Schicht int_1 des Netzes ann auf $real$.
Beispiel: `SETNEURONTHETA(Netz,1,1,0.5)`

`GETNEURONTHETA`
Parameter: $real, ann, int_1, int_2$
Schreibt den Schwellenwert des Neurons int_2 in Schicht int_1 des Netzes ann in $real$.
Beispiel: `GETNEURONTHETA(theta,Netz,1,1)`

`SETNEURONAGGFUNC`
Parameter: $ann, int_1, int_2, const$
Setzt die Aggregationsfunktion des Neurons int_2 in Schicht int_1 des Netzes ann auf $const$.
Beispiel: `SETNEURONAGGFUNC(Netz,1,1,ADD)`

`SETNEURONAGGFUNC`
Parameter: $ann, str, const$
Setzt die Aggregationsfunktion des Neurons str des Netzes ann auf $const$.
Beispiel: `SETNEURONAGGFUNC(Netz,Hidden_1,ADD)`

`GETNEURONAGGFUNC`

Parameter: int_1, ann, int_2, int_3
Schreibt die Aggregationsfunktion des Neurons int_3 in Schicht int_2 des Netzes ann in int_1.
Beispiel: `GETNEURONAGGFUNC(WelcheFunction,Netz,1,1)`

`GETNEURONAGGFUNC`
Parameter: int, ann, str
Schreibt die Aggregationsfunktion des Neurons str des Netzes ann in int.
Beispiel: `GETNEURONAGGFUNC(WelcheFunction,Hidden_1,1,1)`

`SETNEURONACTFUNC`
Parameter: $ann, int_1, int_2, const$
Setzt die Aktivierungsfunktion des Neurons int_2 in Schicht int_1 des Netzes ann auf $const$.
Beispiel: `SETNEURONACTFUNC(Netz,1,1,SIGMOIDAL)`

`SETNEURONACTFUNC`
Parameter: $ann, str, const$
Setzt die Aktivierungsfunktion des Neurons str des Netzes ann auf $const$.
Beispiel: `SETNEURONACTFUNC(Netz,Hidden_1,SIGMOIDAL)`

`GETNEURONACTFUNC`
Parameter: int, ann, int_1, int_2
Schreibt die Aktivierungsfunktion des Neurons int_2 in Schicht int_1 des Netzes ann in int.
Beispiel: `GETNEURONACTFUNC(WelcheFunction,Netz,1,1)`

`SETNEURONOUTFUNC`
Parameter: $ann, int_1, int_2, const$
Setzt die Ausgabefunktion des Neurons int_2 in Schicht int_1 des Netzes ann auf $const$.
Beispiel: `SETNEURONOUTFUNC(Netz,1,1,IDENTITY)`

`SETNEURONOUTFUNC`
Parameter: $ann, str, const$
Setzt die Ausgabefunktion des Neurons str des Netzes ann auf $const$.
Beispiel: `SETNEURONOUTFUNC(Netz,Hidden_1,IDENTITY)`

`GETNEURONOUTFUNC`
Parameter: int_1, ann, int_2, int_3
Schreibt die Ausgabefunktion des Neurons int_3 in Schicht int_2 des Netzes ann in $const$.
Beispiel: `GETNEURONOUTFUNC(WelcheFunction,Netz,1,1)`

`GETNEURONAGGFUNC`
Parameter: int, ann, str
Schreibt die Ausgabefunktion des Neurons str des Netzes ann in int.
Beispiel: `GETNEURONOUTFUNC(WelcheFunction,Hidden_1,1,1)`

`SETUNITDOMAIN`
Parameter: $ann, real_1, real_2$
Setzt den Eingabebereich der Neuronen des Netzes ann auf $[real_1, real_2]$.
Beispiel: `SETUNITDOMAIN(Netz,0,1)`

`GETUNITDOMAIN`
Parameter: $real_1, real_2, ann$
Schreibt den Eingabebereich des Netzes ann in $real_1$ und $real_2$.
Beispiel: `GETUNITDOMAIN(min,max,Netz)`

`SETUNITRANGE`
Parameter: $ann, real_1, real_2$
Setzt den Wertebereich der Neuronen des Netzes ann auf $[real_1, real_2]$.
Beispiel: `SETUNITRANGE(Netz,0,1)`

`GETUNITRANGE`
Parameter: $real_1, real_2, ann$
Schreibt den Wertebereich des Netzes ann in $real_1$ und $real_2$.
Beispiel: `GETUNITRANGE(min,max,Netz)`

`SETWEIGHTRANGE`
Parameter: $ann, real_1, real_2$
Setzt den Wertebereich der Gewiche des Netzes ann auf $[real_1, real_2]$.
Beispiel: `SETUNITRANGE(Netz,0,1)`

`GETWEIGHTRANGE`
Parameter: $real_1, real_2, ann$
Schreibt den Wertebereich der Gewichte ann in $real_1$ und $real_2$.
Beispiel: `GETUNITRANGE(min,max,Netz)`

`GETNUMINPUTNEURONS`
Parameter: int, ann
Schreibt die Anzahl der Eingabeneuronen von ann in int.
Beispiel: `GETNUMINPUTNEURONS(anz,Netz)`

`GETNUMOUTPUTNEURONS`
Parameter: int, ann

Schreibt die Anzahl der Ausgabeneuronen von *ann* in *int*.
Beispiel: `GETNUMOUTPUTNEURONS(anz,Netz)`

`GETNUMLAYERS`
Parameter: *int*, *ann*
Schreibt die Anzahl der Schichten von *ann* in *int*.
Beispiel: `GETNUMLAYERS(anz,Netz)`

`APPLYNET`
Parameter: *ann*
Berechnet die Ausgabewerte aller Neuronen des Netzes *ann* mit den aktuellen Eingabewerten.
Beispiel: `APPLYNET(Netz)`

`LOADANN`
Parameter: *ann*, *str*
Läd die in *str* angebene Datei als neuronales Netz in die Variable *ann*.
Beispiel: `LOADANN(Netz,MRT_Klassifizierer.ann)`

`LOADCNN`
Parameter: *cnn*, *str*
Läd die in *name* angebene Datei als zellulares neuronales Netz in die Variable *var*.
Beispiel: `LOADCNN(Netz,MRT_Denoise.cnn)`

`APPLYNET`
Parameter: img_1, cnn, img_2
Wendet das zellulare neuronale Netz *cnn* auf img_2 an und schreibt das Ergebnis in img_1.
Beispiel: `APPLYNET(KopfOhneRauschen,DeNoiseNet,Kopf)`

C.9 Befehle für evolutionäre Algorithmen

`EAPARAM`
Parameter: fb
Wählt die Variable vom Typ `FUZZYBASE`, die bei einem Optimierungslauf berücksichtigt werden soll.
Beispiel: `EAPARAM(MRTBasis)`

`EASOURCEIMAGE`
Parameter: *img*

Definiert das Bild, welches als Eingabebild behandelt werden soll und im Laufe der Optimierung durch die definierten Bilddaten ersetzt wird.
Beispiel: `EASOURCEIMAGE(Eingabebild)`

`EAFINALIMAGE`
Parameter: *img*
Definiert das Bild, welches nach der Verarbeitung als Ergebnisbild betrachtet werden soll umd mit dem Zielbild `EADESTIMAGE` vergleichen wird.
Beispiel: `EAFINALIMAGE(Ergebnis)`

`EADESTIMAGE`
Parameter: *img*
Definiert das Bild, welches das Zielbild darstellt, mit dem das `EAFINALIMAGE` verglichen wird.
Beispiel: `EADESTIMAGE(Sollbild)`

`EAREADY`
Parameter: keine
Gibt an, daß die Optimierung an dieser Stelle abgebrochen werden kann, da alle benötigten Daten vorliegen. Das System berechnet mit den bekannten Bilddaten die Fitneß der getesten Individuen.
Beispiel: `EAREADY()`

`IFEAMODE`
Parameter: keine
Prüft, ob die Programmausführung im Optimierungsmodus ist und führt im Wahrheitsfall den folgenden Block aus, ansonsten den ggf. existierenden `ELSE`-Teil.
Beispiel: `IFEAMODE(){EASOURCEIMAGE(Bild)}ELSE{LoadImage(Bild,Name)}`

C.10 Vordefinierte Konstanten

Tabelle C.1 zeigt die in CIM^{2}BA/P vordefinierten Konstanten, die als Parameter verwendet werden können.

Name	Anwendung	Bedeutung
TRUE		Wahrheitswert *wahr*
FALSE		Wahrheitswert *falsch*
NULL		Entspricht einem leeren Parameter
RETURNVAL		Variable, die den Rückgabewert einer
		Funktion enthält
DONTCARE		Schlüsselwort zum Ignorieren einer Eigenschaft
SAGITTAL	BV	Sagittale Ansicht eines 3D-Bildes
TRANSVERSAL	BV	Transversale Ansicht eines 3D-Bildes
CORONAR	BV	Coronare Ansicht eine 3D-Bildes
INSIDE	BV	Innerhalb eines Polygons liegende Punkte
OUTSIDE	BV	Außerhalb eines Polygons liegende Punkte
LINEAR	FL	Fuzzy-Mengen-Typ 1
SIGMOIDAL	FL	Fuzzy-Mengen-Typ 2
INVSIGMOIDAL	FL	Fuzzy-Mengen-Typ 3
RADIAL	FL	Fuzzy-Mengen-Typ 4
INVRADIAL	FL	Fuzzy-Mengen-Typ 5
MIN	FL	Minimumfunktion (T-Norm) $\min(a, b)$
MAX	FL	Maximumfunktion (S-Norm) $\max(a, b)$
ALGPROD	FL	Algebraisches Produkt (T-Norm) $a \cdot b$
ALGSUM	FL	Algebraische Summe (S-Norm) $a + b - a \cdot b$
BOUNDEDDIFF	FL	Begrenzte Differenz (T-Norm) $\max(0, a + b - 1)$
BOUNDEDSUM	FL	Begrenzte Summe (S-Norm) $\min(1, a + b)$
DRASTICPROD	FL	Drastisches Produkt (T-Norm)
		Wenn $(max(a, b) = 1) \min(a, b)$ sonst 0
DRASTICSUM	FL	Drastische Summe (S-Norm)
		Wenn $(min(a, b) = 0) \max(a, b)$ sonst 1
COG	FL	Schwerpunktmethode (Defuzifizierung)
LOM	FL	Linkes Maximum (Defuzifizierung)
MOM	FL	Mittleres Maximum (Defuzifizierung)
ROM	FL	Rechtes Mamimum (Defuzifizierung)
ADD	KNN	Additionsfunktion (Aggregation)
PROD	KNN	Produktfunktion (Aggregation)
BINARY	KNN	Binäre Schwellenwertfunktion (Aktivierung)
LINEAR	KNN	Lineare Funktion (Aktivierung)
LOGISTIC	KNN	Logistische Funktion (Aktivierung) $(1 + e^{-x})^{-1}$
TANH	KNN	Tangens Hyperbolicus-Funktion (Aktivierung)
IDENTITY	KNN	Indentitätsfunktion (Ausgabe)

Tabelle C.1: Tabelle der vordefinierten Konstanten.

Abbildungsverzeichnis

Tabellenverzeichnis

Literaturverzeichnis

[AA94] N. N. Aizenberg und I. N. Aizenberg: *CNN-like networks based on multivalued and universal binary neurons: learning and application to image processing.* In: *Third IEEE International Workshop on Cellular Neural Networks and their Applications*, Seiten 153–158, Rom, 1994.

[AAH+01] I. N. Aizenberg, N. N. Aizenberg, J. Hiltner, C. Moraga und E. Meyer zu Bexten: *Cellular neural networks and computational intelligence in medical image processing.* Image and Vision Computing, 19(4):177–183, März 2001.

[AAMM98] N. N. Aizenberg, I. N. Aizenberg, C. Moraga und E. Meyer zu Bexten: *CNN-Algorithms for Medical Image Processing.* In: *Eufit '98- 6th European Congress on Intelligent Techniques & Soft Computing*, Seiten 1311–1315, Aachen, September 1998. Verlag und Druck Mainz GmbH.

[Abm94] W. Abmayr: *Einführung in die digitale Bildverarbeitung.* B.G. Teubner, Stuttgart, 1994.

[Aiz92] I. N. Aizenberg: *The universal logical element over the complex numbers field.* In: *Second IEEE International Workshop on Cellular Neural Networks and their Applications*, Seiten 36–41, München, 1992.

[Aiz99] I. N. Aizenberg: *Neural Networks Based on Multi-valued and Universal Binary Neurons: Theory, Application to Image Processing and Recognition.* In: B. Reusch (Herausgeber): *Computational Intelligence – Theory and Applications*, Band 1625 der Reihe *Lecture Notes in Computer Science*, Seiten 306–316, Berlin – Heidelberg, 1999. Springer-Verlag.

[AMST97] B. Arnolds, H. Müller, D. Saupe und T. Tolxdorff (Herausgeber): *Digitale Bildverarbeitung in der Medizin*, Freiburg, März 1997. Klinikum der Albert-Ludwigs-Universität.

[Ara96] K. Arakawa: *Median filter based on fuzzy rules and its application to image restoration.* Fuzzy Sets and Systems, 77:3–13, 1996.

[BA97] C. K. Bounsaythip und J. T. Alander: *Genetic Algorithms Applied to Image Processing - A Review.* In: *Proceedings of the 3rd Nordic Workshop on Genetic Algorithms (3NWGA)*, Seiten 172–192, Helsinki, Finland, August 1997.

[Bal81] D. Ballard: *Generalizing the Hough Transform to Detect Arbitrary Shapes.* Pattern Recognition, 13(2):111–122, 1981.

[Bez94] J. C. Bezdek: *What is Computational Intelligence?* In: J. M. Zurada, R. J. Marks und C. J. Robinson (Herausgeber): *Computational Intelligence – Imitating Life*, Seiten 1–12. IEEE Press, 1994.

[BG93] H. Bandemer und S. Gottwald: *Einführung in Fuzzy-Methoden.* Akademie Verlag, Berlin, 4. Auflage, 1993.

[BGKM98] H. Brinck, R. Grebe, J. Krone und V. Metzler: *Unüberwachte Bildsegmentierung durch die Adaption geometrischer Objekte mit einem Evolutionsalgorithmus.* In: T. Lehmann, V. Metzler, K. Spitzer und T. Tolxdorff (Herausgeber): *Bildverarbeitung für die Medizin 1998*, Informatik aktuell, Seiten 174–178, Berlin – Heidelberg, März 1998. Springer Verlag.

[BI98] A. Blake und M. Isard: *Active Contours.* Springer-Verlag, London, 1998.

[Bie97] B. Biewer: *Fuzzy-Methoden.* Springer-Verlag, Berlin – Heidelberg, 1997.

[BKKS96] H. Brinck, R. Kottenhoff, J. Krone und C. Strätgen: *Evolutionsalgorithmen in der Medizinischen Bildverarbeitung.* In: T. Lehmann, I. Scholl und K. Spitzer (Herausgeber): *Bildverarbeitung für die Medizin 1996*, Seiten 345–348, Aachen, 1996. Verlag der Augustinus Buchhandlung.

[BP92a] G. N. Bebis und G. M. Papadourakis: *Object Recognition using Invariant Object Boundary Representations and Neural Network Models.* Pattern Recognition, 25(1):25–44, 1992.

[BP92b] J. C. Bezdek und S. K. Pal (Herausgeber): *Fuzzy Models for Pattern Recognition.* IEEE Press, New York, 1992.

[BRJ99] G. Booch, J. Rumbaugh und I. Jacobson: *Das UML-Handbuch.* Addison-Wesley (Deutschland) GmbH, Bonn, 1999.

[CG99] E. Cantú-Paz und D. E. Goldberg: *Parallel Genetic Algorithms with Distributed Panmictic Populations.* Technischer Bericht 99006, University of Illinois at Urban-Champaign – Illinois Genetic Algorithms Laboratory, Illinois, USA, Januar 1999.

[CR93] L. O. Chua und T. Roska: *The CNN Paradigm.* IEEE Transactions on Circuits and Systems – Fundamental Theory and Applications, 40(3):147–156, März 1993.

[CY88a] L. O. Chua und L. Yang: *Cellular Neural Networks: Applications.* Transactions on Circuits and Systems, 35(10):1273–1290, 1988.

[CY88b] L. O. Chua und L. Yang: *Cellular Neural Networks: Theory.* Transactions on Circuits and Systems, 35(10):1257–1272, 1988.

[DG97] J. Dorn und G. Gottlob: *Künstliche Intelligenz.* In: P. Rechenberg und G. Pomberger (Herausgeber): *Informatik-Handbuch*, Seiten 819–838. Carl Hanser Verlag, München – Wien, 1997.

[DH73] R. O. Duda und P. E. Hart: *Pattern Classification and Scene Analysis.* John Whiley & Sons, Inc., New York, 1973.

[DLT72] A. De Luca und S. Termini: *A Definition of a Nonprobabilistic Entropy in the Setting of Fuzzy Sets Theory.* Information and Control, 20(4):301–312, Mai 1972.

[EGLM99] H. Evers, G. Glombitza, T. Lehmann und H.-P. Meinzer (Herausgeber): *Bildverarbeitung für die Medizin 1999*, Informatik aktuell, Berlin – Heidelberg, März 1999. Springer Verlag.

[ESE+99] U. Engelmann, A. Schröter, H. Evers, S. Gundel, M. Schwab und H.-P. Meinzer: *CHILI: Eine Integrationsplattform für medizinische Bildverarbeitungsmethoden.* In: H. Evers, G. Glombitza, T. Lehmann und H.-P. Meinzer (Herausgeber): *Bildverarbeitung für die Medizin 1999*, Informatik aktuell, Seiten 248–252, Berlin – Heidelberg, März 1999. Springer Verlag.

[Fat91] M. Fathi: *Methoden zur Wissensakquisition.* Forschungsbericht 392, Universität Dortmund – FB Informatik, Dortmund, 1991.

[Fat92] M. Fathi: *Wissensakquisition und Wissensmodellierung bei Expertensystemanwendungen.* Skriptum zur Vorlesung, Universität Dortmund – FB Informatik, 1992.

[FH01] M. Fathi und L. Hildebrand: *Intelligent Control Systems Using Soft Computing Methodologies*, Kapitel Evolutionary Concepts for Image

Processing Applications, Seiten 383–412. CRC Press, Boca Raton (Fl), 2001.

[FHJT97] M. Fathi, J. Hiltner, M. Jäger und K.-H. Temme (Herausgeber): *PG Bambus – Baukasten für die Analyse und Modellierung medizinischer Bilddaten mit Hilfe unscharfen Wissens.* Projektgruppenendbericht. Universität Dortmund – FB Informatik, Dortmund, 1997.

[FHTT96] M. Fathi, J. Hiltner, K.-H. Temme und C. Tresp (Herausgeber): *PG MOSES – Anfragen mit unscharfem Gehalt in medizinischen Tutoringsystemen.* Projektgruppenendbericht. Universität Dortmund – FB Informatik, Dortmund, 1996.

[FT94] M. Fathi und K.-H. Temme (Herausgeber): *Ein wissensbasiertes Fuzzy-System zur Bildverarbeitung in der Hirnforschung.* Projektgruppenendbericht. Universität Dortmund – FB Informatik, Dortmund, 1994.

[FTHB94] M. Fathi, C. Tresp, J. Hiltner und K. Becker: *Fuzzy Set Optimization in Use of Medical MR-Image Analysis Based on Evolution Stategies.* In: *IEEE World Wisemen/Women Workshop (WWW)*, Seiten 135–142, Nagoya, Japan, August 1994.

[FW99] F. Fuchs und T. Wleklik: *Entwicklung eines Simulators für zellulare neuronale Netze unter besonderer Berücksichtigung der Methoden von I.N. Aizenberg bezüglich CNN-UBN und CNN-MVN.* Diplomarbeit, Universität Dortmund – FB Informatik, Dortmund, 1999.

[GB97] D. Graham und A. Barrett: *Knowledge-Based Image Processing Systems*, Band 13 der Reihe *Applied Computing.* Springer-Verlag, London, 1997.

[Gör93] G. Görz (Herausgeber): *Einführung in die künstliche Intelligenz.* Addison-Wesley (Deutschland) GmbH, Bonn, 1993.

[GT00] V. Gruhn und A. Thiel: *Komponentenmodelle.* Addison-Wesley (Deutschland) GmbH, München, 2000.

[GW93] R. C. Gonzalez und R. E. Woods: *Digital Image Processing.* Addison-Wesley Publishing Company, Inc., Reading (MA), 1993.

[Hab91] P. Haberäcker: *Digitale Bildverarbeitung – Grundlagen und Anwendungen.* Carl Hanser Verlag, München – Wien, 4. Auflage, 1991.

[Hab95] P. Haberäcker: *Praxis der Digitalen Bildverarbeitung und Mustererkennung.* Carl Hanser Verlag, München – Wien, 1995.

[HAMM98] J. Hiltner, I. N. Aizenberg, E. Meyer zu Bexten und C. Moraga: *Neural Networks and Fuzzy Logic in Medical Image Processing.* In: T. Yamakawa und G. Matsumuto (Herausgeber): *Methodologies for the Conception, Design and Application of Soft Computing*, Seiten 325–328, Iizuka, Japan, Oktober 1998. World Scientific Publishing Co. Pte. Ltd.

[Han98] K. M. Hanson (Herausgeber): *Image Processing (Medical Imaging 1998)*, Band 3338, Newport Beach, California, USA, 1998. Society of Photo-optical Instrumentation Engineers.

[Han99] K. M. Hanson (Herausgeber): *Image Processing (Medical Imaging 1999)*, Band 3661, Newport Beach, California, USA, 1999. Society of Photo-optical Instrumentation Engineers.

[Han00] H. Handels: *Medizinische Bildverarbeitung.* B.G. Teubner, Stuttgart, 2000.

[Hei97] R. Heider: *Evolutionäre Synthese neuronaler Netze unter Verwendung von Graph-Grammatiken.* Dissertation, Universität Dortmund – FB Informatik, Dortmund, 1997.

[HF00] L. Hildebrand und M. Fathi: *Vision systems for the inspection of resistance welding joints.* In: K. W. Tobin und N. S. Chang (Herausgeber): *Procedings of SPIE – Vol. 3966*, San Jose, USA, 2000.

[HFR98] J. Hiltner, M. Fathi und B. Reusch: *Fuzzy Object Descriptions for Medical Image Processing.* In: T. Yamakawa und G. Matsumuto (Herausgeber): *Methodologies for the Conception, Design and Application of Soft Computing*, Seiten 333–336, Iizuka, Japan, Oktober 1998. World Scientific Publishing Co. Pte. Ltd. Best PhD-Student Paper Award.

[HFR99] J. Hiltner, M. Fathi und B. Reusch (Herausgeber): *PG DAWN – Entwicklung eines wissensgesteuerten Bildverarbeitungssystems zur Erkennung von Verkalkungen in Koronararterien.* Projektgruppenendbericht. Universität Dortmund – FB Informatik, Dortmund, 1999.

[HFR00] J. Hiltner, M. Fathi und B. Reusch: *New Model for Medical Image Analysis based on Computational Intelligence.* In: K. M. Hanson (Herausgeber): *Medical Imaging 2000*, Seiten 1295–1302, San Diego, USA, Februar 2000. SPIE - International Socienty for Optical Engineering.

[HFR01] J. Hiltner, M. Fathi und B. Reusch: *An approach to use linguistic and model-based fuzzy expert knowledge for the analysis of MRT images.* Image and Vision Computing, 19(4):195–206, März 2001.

[HHLM01] H. Handels, A. Horsch, T. Lehmann und H.-P. Meinzer (Herausgeber): *Bildverarbeitung für die Medizin 2001*, Informatik aktuell, Berlin – Heidelberg, März 2001. Springer Verlag.

[Hil96] L. Hildebrand: *Evolutionsstrategien zur automatischen Generierung und Optimierung von Fuzzy-Inferenzsystemen.* Diplomarbeit, Universität Dortmund – FB Informatik, Dortmund, 1996.

[Hil98] J. Hiltner: *Operatoren zur deskriptiven und modellbasierten unscharfen Wissensbeschreibung in der medizinischen Bildverarbeitung.* In: T. Lehmann, V. Metzler, K. Spitzer und T. Tolxdorff (Herausgeber): *Bildverarbeitung für die Medizin 1998*, Seiten 114–118, Berlin – Heidelberg, März 1998. Springer-Verlag.

[Hil00] J. Hiltner: *Wissensbasierte Segmentierung mit Hilfe aktiver Konturen.* In: B. Reusch (Herausgeber): *Interdisziplinäre Methoden der Informatik*, Nummer 729 in *Forschungsbericht*, Seiten 93–100, Dortmund, März 2000. Universität Dortmund – FB Informatik.

[Hil01] J. Hiltner: *Ein Baukasten zur Analyse medizinischer Bilddaten mit Hilfe neuronaler Netze und Fuzzy-Logik.* In: H. Handels, A. Horsch, T. Lehmann und H.-P. Meinzer (Herausgeber): *Bildverarbeitung für die Medizin 2001*, Seiten 315–319, Berlin – Heidelberg, März 2001. Springer Verlag.

[HJMF97] J. Hiltner, M. Jäger, E. Meyer zu Bexten und M. Fathi: *Analyse medizinischer Bilddaten mit Hilfe unscharfen Wissens.* In: B.Arnolds et al. [AMST97], Seiten 127–131.

[HJMT99] J. Hiltner, M. Jäger, M. Moser und C. Tresp: *Fuzzy Image Analysis for Medical Applications.* In: L. Jain, R. Johnson, Y. Takefuji und L. Zadeh (Herausgeber): *Knowledge-Based Intelligent Techniques in Industry*, Kapitel 3, Seiten 85–116. CRC-Press, Boca Raton (FL), USA, 1999.

[HKH98] Y. Hata, S. Kobashi und S. Hirano: *Medical Image Segmentation by Fuzzy Logic Techniques.* In: *IEEE International Conference on Systems, Man and Cybernetics (SMC '98)*, Seiten 4098–4103, San Diego, USA, Oktober 1998.

[HKP94] J. H. Han, L. T. Koczy und T. Poston: *Fuzzy Hough transform.* Pattern Recognition Letters, 15:649–658, Juli 1994.

[HL00] A. Horsch und T. Lehmann (Herausgeber): *Bildverarbeitung für die Medizin 2000*, Informatik aktuell, Berlin – Heidelberg, März 2000. Springer Verlag.

[Hol75] J. H. Holland: *Adaptation in Natural and Artificial Systems: An Introductory Analysis with Applications to Biology, Control, and Artificial Intelligence.* University of Michigan Press, Ann Arbor, USA, 1975.

[Hou62] P. Hough: *Methods and Means for Recognizing Complex Patterns*, 1962. U.S. Patent No. 3069654.

[HR00] L. Hildebrand und B. Reusch: *Fuzzy Color Processing.* In: E. E. Kerre und M. Nachtegael (Herausgeber): *Fuzzy Techniques in Image Processing*, Studies in Fuzziness and Soft Computing, Kapitel 10, Seiten 267–286. Physica-Verlag, Heidelberg – New York, 2000.

[HRF99] L. Hildebrand, B. Reusch und M. Fathi: *Directed Mutation – A New Selfadaptation for Evolution Strategies.* In: P. J. Angeline (Herausgeber): *Congress on Evolutionary Computation*, Band 2, Seiten 1550–1557, Piscataway, New Jersey, 1999. IEEE, IEEE Service Center.

[HRK+99] H. Handels, T. Roß, J. Kreusch, H. Wolff und S. J. Pöppl: *Feature Selection for optimized skin tumor recognition using genetic algorithms.* Artificial Intelligence in Medicine, 16:283–297, 1999.

[HT94] J. Hiltner und C. Tresp: *Optimierung eines wissensbasierten Systems zur Analyse von MR-Tomogrammen – Evolutionäre Strategien und Bildnachbearbeitung.* Diplomarbeit, Universität Dortmund – FB Informatik, Dortmund, September 1994.

[HW95] L.-K. Huang und M.-J. J. Wang: *Image Thresholding by Minimizing the Measure of Fuzziness.* Pattern Recognition, 28(1):41–51, 1995.

[Jäh97] B. Jähne: *Digitale Bildverarbeitung.* Springer-Verlag, Berlin – Heidelberg, 4. Auflage, 1997.

[JM96] M. Jäger und M. Moser: *Ein Modellentwurf zur wissensbasierten Analyse von 3D-Bildern unter Berücksichtigung von Unsicherheiten.* Diplomarbeit, Universität Dortmund – FB Informatik, Dortmund, 1996.

[Ker00] M. Kersten: *Optimale Bilder.* c't Heft 20, September 2000.

[Kho99] Khoral Research, Inc.: *Khoros 2000*, 1999. Albuquerque, New Mexico, USA.

[Köh96] C. Köhn: *Bildanalyse und Bilddatenkompression.* Carl Hanser Verlag, München – Wien, 1996.

[Kop97] H. Kopp: *Bildverarbeitung interaktiv.* B.G. Teubner, Stuttgart, 1997.

[KTL+98] G. Krell, H. R. Tizhoosh, T. Lilienblum, C. Moore und B. Michaelis: *Enhancement and assciative restauration of electronic portal images in radiotherapy.* International Journal of Medical Informatics, (49):157–171, 1998.

[KWT87] M. Kass, A. Witkin und D. Terzopoulos: *Snakes: Active Contour Models.* In: *International Journal of Computer Vision*, Band 3, Seiten 259–268, 1987.

[KZ95] R. Klette und P. Zamperoni: *Handbuch der Operatoren für die Bildbearbeitung.* Friedrich Vieweg und Sohn Verlagsgesellschaft mbH, Braunschweig – Wiesbaden, 2. Auflage, 1995.

[LE89] C. E. Liedtke und M. Ender: *Wissensbasierte Bildverarbeitung*, Band 19 der Reihe *Nachrichtentechnik.* Springer-Verlag, Berlin – Heidelberg, 1989.

[LH95] Y.-G. Lee und Y.-C. Hsueh: *Fuzzy logic approach for removing noise.* Neuro Computing, 9:349–355, 1995.

[LHH02] T. Lehmann, J. Hiltner und H. Handels: *Handbuch der Medizininformatik*, Kapitel 8. Carl Hanser Verlag, München – Wien, 2002. Erscheint März 2002.

[LMST98] T. Lehmann, V. Metzler, K. Spitzer und T. Tolxdorff (Herausgeber): *Bildverarbeitung für die Medizin 1998*, Informatik aktuell, Berlin – Heidelberg, März 1998. Springer Verlag.

[LOPR97] T. Lehmann, W. Oberschelp, E. Pelikan und R. Repges: *Bildverarbeitung für die Medizin.* Springer-Verlag, Berlin – Heidelberg, 1997.

[LSS96] T. Lehmann, I. Scholl und K. Spitzer (Herausgeber): *Bildverarbeitung für die Medizin 1996*, Aachen, März 1996. Verlag der Augustinus Buchhandlung.

[LVIF98] H. U. Lemke, M. W. Vannier, K. Inamura und A. G. Farman (Herausgeber): *Computer Assisted Radiology and Surgery*, Amsterdam, 1998. Elsevier Science B.V.

[LVIF99] H. U. Lemke, M. W. Vannier, K. Inamura und A. G. Farman (Herausgeber): *Computer Assisted Radiology and Surgery*, Amsterdam, 1999. Elsevier Science B.V.

[May99] J. Mayer: *Die Architektur des Gehirns.* `http://www.ims.uni-stuttgart.de/phonetik/joerg/sgtutorial/architektur.h%tml`, 1999. Universität Stuttgart, Institut für natürliche Sprachverarbeitung.

[Mei00] H.-P. Meinzer: *20 Jahre medizinische Bildverarbeitung – Ränder, Regionen, Intelligenz und Wahrnehmung.* In: A.Horsch und T. Lehmann [HL00], Seiten 1–9.

[Men90] W. Menhardt: *Unscharfe Mengen (Fuzzy Sets) zur Behandlung von Unsicherheit in der Bildanalyse.* Dissertation, Universität Hamburg, Hamburg, 1990.

[MHJ98] E. Meyer zu Bexten, J. Hiltner und M. Jäger: *Entwicklung grafischer Benutzeroberflächen.* In: C. Horn und I. O. Kerner (Herausgeber): *Lehr- und Übungsbuch Informatik – Technische Informatik und Systemgestaltung*, Band 4, Kapitel 5, Seiten 239–274. Fachbuchverlag Leipzig – im Carl Hanser Verlag, München – Wien, 1998.

[MHM98] E. Meyer zu Bexten, J. Hiltner und C. Moraga: *State of the Art and Trends on Medical Image Processing in Germany.* In: T. Yamakawa und G. Matsumuto (Herausgeber): *Methodologies for the Conception, Design and Application of Soft Computing*, Seiten 317–320, Iizuka, Japan, Oktober 1998. World Scientific Publishing Co. Pte. Ltd.

[Min74] M. Minsky: *A framework for representing knowledge.* Technischer Bericht, Massachusetts Institute of Technology A.I. Laboratory, 1974.

[Mül92] H. Müller: *Digitale Bildverarbeitung.* Skriptum zur Vorlesung, Universität Dortmund – FB Informatik, 1992.

[MVKG97] V. Metzler, R. Vandenhouten, J. Krone und R. Grebe: *Wissensbasierte Bildsegmentierung mittels stochastischer Optimierung.* In: B. Arnolds, H. Müller, D. Saupe und T. Tolxdorff (Herausgeber): *Digitale Bildverarbeitung in der Medizin*, Seiten 75–80, Freiburg, März 1997. Klinikum der Albert-Ludwigs-Universität.

[NB87] H. Niemann und H. Bunke: *Künstliche Intelligenz in Bild- und Sprachanalyse.* B.G. Teubner, Stuttgart, 1987.

[Neu95] H. A. Neumann: *Objektorientierte Entwicklung von Software-Systemen.* Addison-Wesley (Deutschland) GmbH, Bonn, 1995.

[Nie83] H. Niemann: *Klassifikation von Mustern.* Springer-Verlag, Berlin – Heidelberg, 1983.

[NKK94] D. Nauck, F. Klawonn und R. Kruse: *Neuronale Netze und Fuzzy-Systeme.* Künstliche Intelligenz. Friedrich Vieweg und Sohn Verlagsgesellschaft mbH, Braunschweig – Wiesbaden, 1994.

[Pat97] D. Patterson: *Künstliche neuronale Netze – Das Lehrbuch.* Prentice Hall Verlag GmbH, München, 1997.

[PDMP+94] K. Philip, E. Dove, D. Mc Pherson, N. Gotteiner, W. Stanford und K. Chandran: *The Fuzzy Hough Transform-Feature Extraction in Medical Images.* IEEE Transactions on Medical Imaging, 13(2):235–240, 1994.

[Ped89] W. Pedrycz: *Fuzzy Control and Fuzzy Systems.* John Wiley & Son Inc., Yew York, 1989.

[PJRP83] C. L. Partain, A. E. James, F. D. Rollo und R. R. Price (Herausgeber): *Nuclear Magnetic Resonance (NMR) Imaging.* W. B. Saunders Company, Philadelphia, PA, USA, 1983.

[Poh00] H. Pohlheim: *Evolutionäre Algorithmen.* Springer Verlag, Berlin – Heidelberg, 2000.

[Pol97] S. Pollandt: *Fuzzy-Begriffe.* Springer-Verlag, Berlin – Heidelberg, 1997.

[Pre70] J. Prewitt: *Object Enhancement and Extraction.* In: B. Lipkin und A. Rosenfeld (Herausgeber): *Picture Processing and Psychopictorics*, Seiten 75–149, New York, 1970. Acedemic Press.

[Qbe01] Qbeo, Inc.: *PhotoGenetics*, 2001. `http://www.qbeo.com`.

[Rec73] I. Rechenberg: *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution.* Frommann-Holzboog, Stuttgart, 1973.

[Rei91] U. Reimer: *Einführung in die Wissensrepräsentation.* B.G. Teubner, Stuttgart, 1991.

[Res98] Research Systems, Inc.: *Interactive Data Language*, 1998. `http://www.rsinc.com`.

[RFH98] B. Reusch, M. Fathi und J. Hiltner (Herausgeber): *PG Anomalia – Entwicklung eines wissensgesteuerten Bildverarbeitungssystems zur*

Erkennung von pathologischen Strukturen in radiologischen Bilddaten. Projektgruppenendbericht. Universität Dortmund – FB Informatik, Dortmund, 1998.

[RHM99] J. Rittscher, J. Hiltner und C. Moraga: *Künstliche Neuronale Netzwerke zur Vorhersage der Hirnkontur.* In: H.Evers et al. [EGLM99], Seiten 302–306.

[Rit98] J. Rittscher: *Medizinische Bildverarbeitung mit neuronalen Netzen.* Diplomarbeit, Universität Dortmund – FB Informatik, Dortmund, 1998.

[RMS91] H. Ritter, T. Martinetz und K. Schulten: *Neuronale Netze.* Addison-Wesley (Deutschland) GmbH, Bonn, 1991.

[Roj94] R. Rojas: *Mathematische Aspekte der angewandten Informatik*, Kapitel Was können neuronale Netze?, Seiten 55–88. Spektrum Akademischer Verlag, Mannheim, 1994.

[Ros58] F. Rosenblatt: *The Perceptron: A probabilistic model for information storage and organization in the brain.* Psychological Review, 65:386–408, 1958.

[Ros79] A. Rosenfeld: *Fuzzy Digital Topology.* Information and Control, 40(1):76–87, 1979.

[SBS+99] A. Schenk, J. Breitenborn, D. Selle, T. Schindewolf, Böm, W. Spindler, H. Jürgens und H.-O. Peitgen: *HLabMed-Workstation – Eine Entwicklungsumgebung für radiologische Anwendungen.* In: H. Evers, G. Glombitza, T. Lehmann und H.-P. Meinzer (Herausgeber): *Bildverarbeitung für die Medizin 1999*, Informatik aktuell, Seiten 238–242, Berlin – Heidelberg, März 1999. Springer Verlag.

[Sch77] H.-P. Schwefel: *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, Band 26 der Reihe *Interdisciplinary Systems Research.* Birkhäuser Verlag, Basel, 1977.

[Sch95] H.-P. Schwefel: *Evolution and Optimum Seeking.* Sixth-Generation Computer Technology. John-Wiley & Sons, Inc., New York, 1995.

[SHF94] E. Schöneburg, F. Heinzmann und S. Feddersen: *Genetische Algorithmen und Evolutionsstrategien.* Addison-Wesley (Deutschland) GmbH, Bonn, 1994.

[SHP01] C. Sieg, H. Handels und S. J. Pöppl: *Automatische Segmentierung von kontrastmittelaufnehmenden Hirntumoren in multispektralen*

MR-Bilddaten mittels Backpropagation-Netzwerken. In: H.Handels et al. [HHLM01], Seiten 347–351.

[Sie92] Siemens AG: *Magnete, Spins und Resonanzen – Eine Einführung in die Grundlagen der Kernspintomographie*, 1992.

[Sof98] Soft Imaging Systems: *analySIS*, 1998. `http://www.soft-imaging.de`.

[Spi93] M. Spies: *Unsicheres Wissen.* Spektrum Akademischer Verlag GmbH, Heidelberg – Berlin – Oxford, 1993.

[Thi97] H. Thiele: *Einführung in die Fuzzy-Logik.* Skriptum zur Vorlesung, Universität Dortmund – FB Informatik, 1997.

[Tiz98] H. R. Tizhoosh: *Fuzzy-Bildverarbeitung – Einführung in Theorie und Praxis.* Springer-Verlag, Berlin – Heidelberg, 1998.

[VPCC98] G. Valli, R. Poli, S. Cagnoni und G. Coppini: *Neural Networks and Prior Knowledge Help the Segmentation of Medical Images.* Journal of Computing and Technology, 6(2):117–133, Juni 1998.

[VV00] C. Vertan und B. Vasile: *Fuzzy Nonlinear Filtering of Color Images: A Survey.* In: E. E. Kerre und M. Nachtegael (Herausgeber): *Fuzzy Techniques in Image Processing*, Studies in Fuzziness and Soft Computing, Kapitel 9, Seiten 248–266. Physica-Verlag, Heidelberg – New York, 2000.

[Wah89] F. M. Wahl: *Digitale Bildsignalverarbeitung*, Band 13 der Reihe *Nachrichtentechnik.* Springer-Verlag, Berlin – Heidelberg, 1989.

[Yin99] P.-Y. Yin: *A fast scheme for optimal thresholding using genetic algorithms.* Signal Processing, (72):85–95, 1999.

[Zad65] L. A. Zadeh: *Fuzzy Sets.* Information and control, 8(3):338–353, Juni 1965.

[Zad75a] L. A. Zadeh: *The Concept of a Linguistic Variable and its Application to Approximate Reasoning – I.* Information Sciences, (8):199–249, 1975.

[Zad75b] L. A. Zadeh: *The Concept of a Linguistic Variable and its Application to Approximate Reasoning – II.* Information Sciences, (8):301–357, 1975.

[Zad75c] L. A. Zadeh: *The Concept of a Linguistic Variable and its Application to Approximate Reasoning – III.* Information Sciences, (9):43–80, 1975.

[Zei84] E. Zeitler: *Kernspintomographie – Einführung für Ärzte und Medizinstudenten.* Deutscher Ärzte Verlag GmbH, Köln–Lövenich, 1984.

[Zel94] A. Zell: *Simulation neuronaler Netze.* Addison-Wesley (Deutschland) GmbH, Bonn, 1994.

Index