Merkmalsbasierte Ansätze zur neuronalen Klassifikation subantarktischer Algen

Diplomarbeit

am Lehrstuhl für Kognitive Systeme
des Instituts für Informatik und Praktische Mathematik
der Christian–Albrechts–Universität zu Kiel
Prof. Dr. Gerald Sommer

vorgelegt von

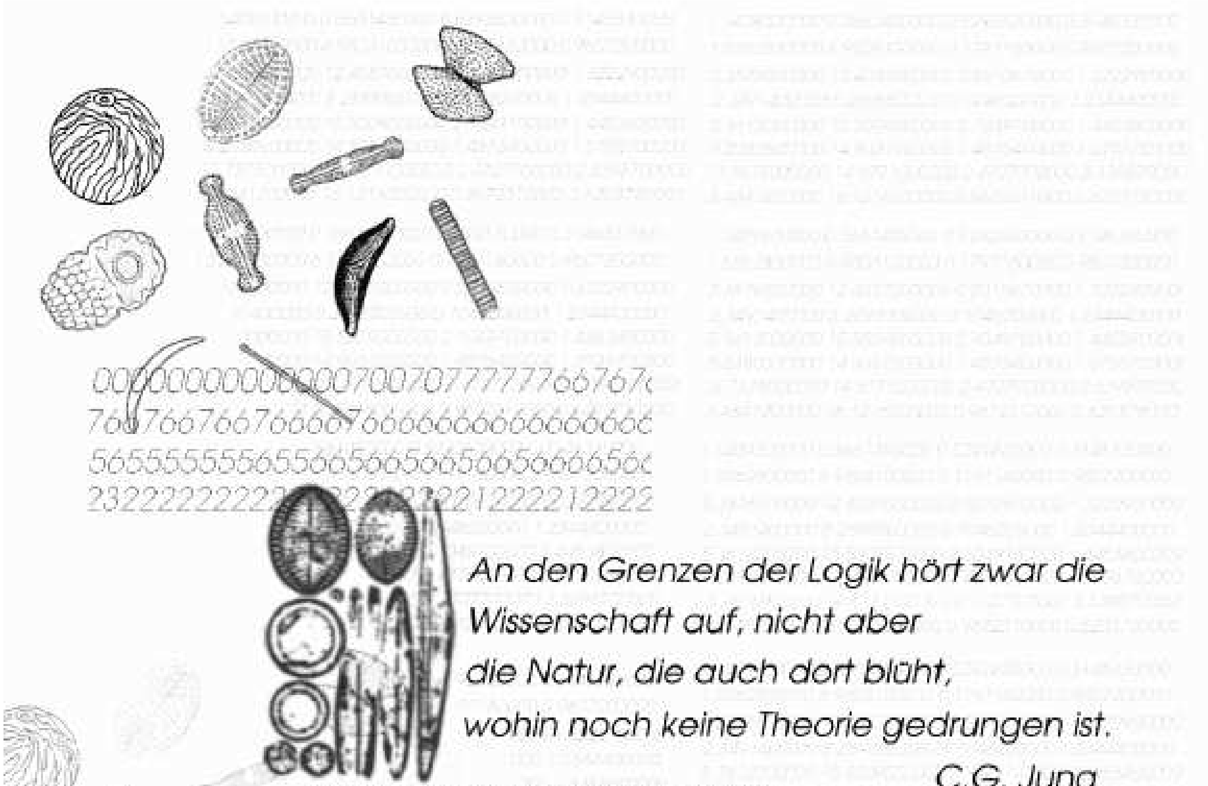
RAMONA KLOOS

September 1999

Betreuer: Prof. Dr. Gerald Sommer

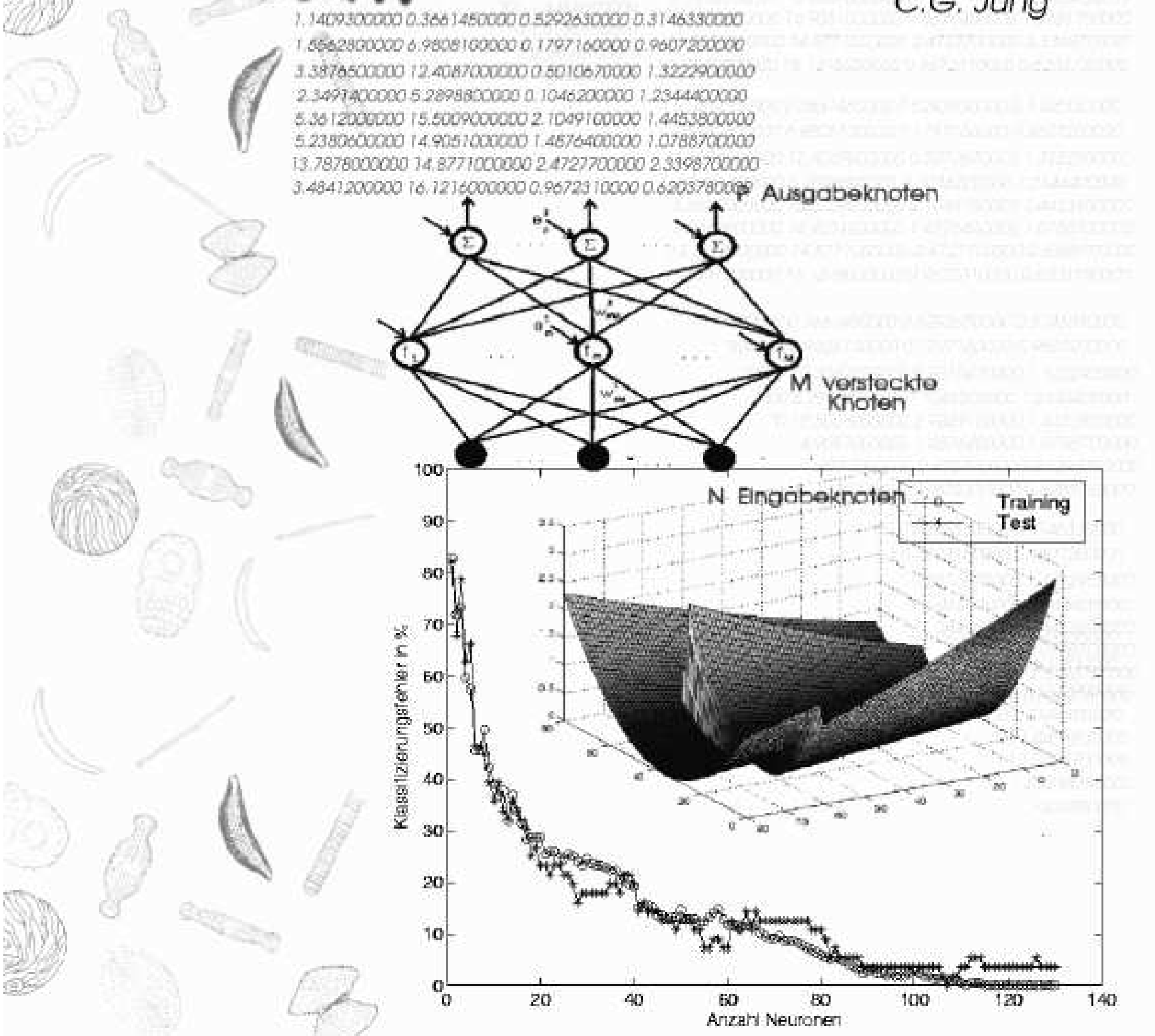
Zusammenfassung

In dieser Diplomarbeit wird ein merkmalsbasiertes Klassifikationsverfahren für Algen mittels Neuronaler Netze vorgestellt und untersucht. Im Mittelpunkt der Arbeit steht dabei die Konstruktion eines rotations-, translations- und skalierungsinvarianten Merkmalsvektors und dessen Integration in die Topologie zweier verschiedener Neuronaler Netze (das Geometrische Netz und das DCS-Netz). Es erfolgen vergleichende experimentelle Studien zur Beurteilung der Leistungsfähigkeit der angewendeten Netze zur Eignung der verschiedenen Merkmalsvektoren.



An den Grenzen der Logik hört zwar die Wissenschaft auf, nicht aber die Natur, die auch dort blüht, wohin noch keine Theorie gedrungen ist.

C.G. Jung



Inhaltsverzeichnis

Kapitel 1

Einführung

Das **Ziel** dieser Diplomarbeit ist es, ein merkmalsbasiertes Klassifikationsverfahren für antarktische Algen mittels Neuronaler Netze zu entwickeln, zu testen und hinsichtlich seiner Anwendungseignung zu bewerten. Von zentraler Bedeutung ist dabei die Konstruktion eines rotations-, translations-, und skalierungsinvarianten Merkmalsvektors ([MZ92]). Um solche Merkmale extrahieren zu können, sind Vorverarbeitungsschritte notwendig, die für die Qualität und Beschaffenheit der Merkmalsvektoren, die als Eingabe für das Neuronale Netz dienen werden, von entscheidender Bedeutung sind. Der Methodik der Vorverarbeitung ist aufgrund ihrer substantiellen Bedeutung für die Güte des Klassifikators ein eigenes Kapitel gewidmet. Die Merkmale, die diese Invarianzen vorweisen und deren Eigenschaften wir uns zunutze gemacht haben, sind zum einen die Fourierdeskriptoren ([PF77], [Bur79], [Arb90]) und zum anderen die Momente (nach [Hu62]). Auch weitere Merkmale, wie der Umfang, die Rundheit, die Exzentrizität ([Jäh97]) werden berücksichtigt und mit den anderen Merkmalen zu verschiedenen Merkmalsvektoren geeignet zusammengetragen. Dabei wird die Integration dieser Merkmalsvektoren in die Struktur zweier unterschiedlicher Neuronaler Netze (Grundprinzipien siehe [Ham93]) untersucht. Die Netze werden mit einem Teil der Daten trainiert und mit einem hiervon disjunkten Teil getestet. Bei den Neuronalen Netzen handelt es sich erstens um das von Jörg Bruske im Rahmen seiner Dissertation am Lehrstuhl für Kognitive Systeme entwickelte *DCS-Netz* und zweitens um das auf der Geometrischen Algebra basierende *Geometrische Netz* ([BCB97], [Buc97]). Diese zwei Netze werden erstmalig auf ihre Eignung und hinsichtlich ihrer Leistungsfähigkeit für eine Klassifikationsaufgabe hin untersucht und miteinander verglichen.

Die **Idee** für diese Diplomarbeit entstand in Zusammenarbeit mit dem *Institut für Polarökologie* der *Christian–Albrechts–Universität zu Kiel*. Herr Professor Bölter ist dort seit zwanzig Jahren auf dem Gebiet der polarökologischen Forschung tätig. Eines seiner Hauptforschungfelder ist die Bestimmung des *Biovolumens* der in den Antarktischen Gewässern und Böden existierenden Mikroorganismen ([BMD93], [Böl94], [Böl95]). In regelmäßigen Abständen sind Expeditionen in die Antarktis organisiert und durchgefürt worden (z. B. 1996 nach Casey, in die australische Antarktis, [BB93]), wobei Proben aus verschiedenen geologischen Schichten entnommen worden sind, um sie hinsichtlich ihrer Taxonomie und

Verteilung in Bodenhorizonten zu untersuchen und auszuwerten.

Die qualitative und quantitative Auswertung von mikroskopischen Analyseverfahren kann, sogar für das geschulte Mikrobiologen–Auge, mühselig und zeitintensiv werden. Es liegt nahe, die Erkennung/Wiedererkennung von Formen und die Klassifikation dominanter Algen und Cyanobakterien (verschiedene Klassen von Formen und Größen), insbesondere der Diatomeen zu automatisieren. (Eine ausführliche Beschreibung der Algentypen und der Terminologie findet sich in der polarökologischen Einführung oder auch in [Gam69]). Die automatische Klassifikation und Zählung der häufigsten Algen wird aufgrund von binären morphologischen Merkmalen vorgenommen und soll für acht Klassen von Algen gelöst werden (für eine genaue Unterteilung der Klassen nach morphologischer Taxonomie siehe [DF97]). Die morphologische Ähnlichkeit vieler Algen führt unter Anwendung mikroskopischer Bilder eventuell zu linear nicht mehr trennbaren Merkmalsgruppen (Cluster). Deshalb soll ein nichtlineares neuronales Verfahren eingesetzt werden, welches die Clusteranalyse und damit den Entwurf des Klassifikators durch Lernen aus Beispielbildern mit Hilfe von Selbstorganisation unterstützen soll. Auf diesem Wege ist die Verbindung zur digitalen Bildverarbeitung und damit zum *Lehrstuhl für Kognitive Systeme* an der *Christian–Albrechts–Universität zu Kiel*, geleitet von Prof. Sommer, hergestellt.

Das erforderliche Datenmaterial für die Ausführung des neuronalen Lernverfahrens wurde aus der *Datei morphologischer Kriterien für phototrophe Organismen antarktischer Böden* entnommen ([Böl96]). Sie basiert auf derzeit 950 Abbildungen in der Antarktis gefundener Algen und Cyanobakterien und ist Ausgangspunkt für oben genannte Mustererkennung. Diese Datei soll in enger Kooperation mit den Kollegen des Auslandes auch hinsichtlich taxonomischer Aspekte vervollständigt werden. Die Datei wurde durch weitere Quellen ([SPPW79], [EG95], [PHR91]) ergänzt und damit der Aufbau einer allgemein zugänglichen Internationalen Dantenbank [BSK98] realisiert, die via Internet allen interessierten Arbeitsgruppen zugänglich gemacht wurde (siehe hierfür 2.1.1).

Der **Weg**, der zur Verwirklichung des oben beschriebenen Arbeitszieles führt, soll hier kurz geschildert werden. Die vorliegende Arbeit baut sich wie folgt auf:
Im Anschluß an diese Einführung erfolgt in Kapitel 2 eine ausführliche Darstellung der Methodik der Vorverarbeitung. Dabei wird zunächst kurz auf den Aufbau der Algen-Datenbank und der unterschiedlichen Qualität des Bildmaterials eingegangen. Anschließend werden die Arbeitsschritte der digitalen Vorverarbeitung vorgestellt und die informationstheoretischen Grundlagen der Merkmalsextraktion detailliert beschrieben.
In Kapitel 3 wird die Methodik der durchgeführten Vergleichsstudien offengelegt. Besonderes Gewicht liegt hierbei auf einer eingehenden, deduktiven Darlegung der modelltheoretischen Ansätze der im Rahmen dieser Arbeit zur Anwendung gelangten Neuronalen Netze.
In Kapitel 4 werden die Ergebnisse der vergleichenden Studien vorgestellt, bevor sie in Kapitel 5 abschließend methodenkritisch diskutiert und hinsichtlich ihrer Anwendungseignung bewertet werden. Die Arbeit schließt mit einem kurzen Ausblick auf den zukünftigen Forschungsbedarf.

1.1 Polarökologische Einführung

Was sind Algen?

Algen sind primär autotrophe ein- bis mehrzellige Thallophyten (Lagerpflanzen), die häufig Kolonien oder filamentöse Strukturen bilden ([Din90]). Als Primärproduzenten organischer Substanz sowie als Nahrungsquelle für heterotrophe Organismen sind Algen wesentliche Glieder aquatischer und terrestrischer Ökosysteme. Während marine und limnische Ökosysteme - je nach Lichtdurchlässigkeit - bis zu einer Tiefe von 150m Algen Lebensraum bieten, beschränkt sich ihr Vorkommen in terrestrischen Ökosystemen vor allem auf lichtgeflutete Habitate im Bereich der Vegetationsdecke und deren Bestandsabfall, exponierte Steine, sowie die obersten Millimeter der Bodenkrume ([Gam69], [HJM93], [EG95]).

Unter den ariden bis extrem ariden Bedingungen der Antarktis sind photoautotrophe Mikroorganismen als Primärproduzenten der organischen Bodensubstanz von besonderer Bedeutung. So dienen sie nicht nur als Energielieferanten heterotropher Mikroorganismen, sondern haben Einfluß auf die bodenbildenden Prozesse: Der Boden kann als regulatorisches Hauptkompartiment in den Stoff-, Energie- und Informationsflüssen terrestrischer Ökosysteme aufgefaßt werden. Unter den extremen Umweltbedingungen der Antarktis sind *höher entwickelte Pflanzen* kaum überlebensfähig, daher können Kryptogamen (Moose, Flechten, *Algen*) ökologische Nischen besiedeln.

Die räumliche Verteilung der Algen im Boden konzentriert sich vor allem auf die Oberfläche und die obersten 5 bis 10 cm der Rhizosphäre, diese Schichten stellen die idealen Lebensräume für viele Bodenalgen dar.

Funktion und Vielfalt der Algen

Algen sind jedoch nicht nur von großer ökologischer, sondern auch von phyllogenetischer Bedeutung. Da man vermutet, daß alle Gruppen der Tiere und Pflanzen im Meer entstanden sind - wo man noch heute Vertreter uralter evolutionärer Linien finden kann - bieten phykologische Arbeiten einen wichtigen Beitrag zum Verständnis der *Diversität* und *Phyllogenie* der Pflanzenwelt. Elektronenmikroskopische und molekulargenetische Untersuchungen haben dabei in jüngerer Zeit zu einer durchgreifenden Überarbeitung des Algensystems geführt. So bieten diese modernen Methoden die Möglichkeit beim Formulieren stammesgeschichtlicher Hypothesen das Gebiet reiner Spekulation zu verlassen: Stammbäume basieren jetzt vielfach auf den Sequenzen der ribosomalen RNA und des ribosomalen RNA-Gens, sowie der Struktur und Sequenzen des Chloroplastengenoms ([HJM93]). Im Boden lebende Formen sind meist kleiner als die verwandten aquatischen Arten oder Stämme; die meisten sind Kosmopoliten, das heißt euryöke Sippen mit sehr weiter ökologischer Amplitude und zum Teil weltweiter Verbreitung ([EG95]). Die floristische Zusammensetzung der Bodenalgenflora und damit die Anzahl der als Bodenalgen im weitesten Sinne bezeichneten Sippen wird allgemein unterschiedlich betrachtet ([EG95]). Im Kontext dieser Arbeit werden als Algen die Gruppe der photoautotrophen Mikroorganismen verstanden ([Böl96], die neben *Chlorophyceaen* (Grünalgen), *Xanthophyceaen* (Gelbgrünalgen) und *Chrysophyceaen* (Diatomeen bzw. Kieselalgen) auch die Gruppe der *Cyanophyceaen* (Blaualgen) umfaßt.

Forschungsstand

Forschungen im Bereich der Bodenalgenkunde beschränken sich zumeist auf punktuelle, quantitative Erfassungen der Algen in verschiedenen Böden oder in qualitativer Weise auf die Floristik und Taxonomie diverser Bodenalgengruppen. Neben speziellen mikrobiologischen Methoden, wie der *Epifluoreszenzmikroskopie* nach [BMD93] (eine Methode zur direkten Bestimmung des *Biovolumens* photoautotropher Mikroorganismen, bzw. deren Mikrohabitate), erfordert das Studium aeroterrestrischer Algen daher traditionell eine umfassende Kenntnis der Taxonomie, Morphologie und Zytologie zahlreicher Formenkreise. Die dazu notwendige Literatur ist weltweit verstreut und dem praxisorientierten Bodenmikrobiologen oder Phykologen oft nicht zugänglich. Im Rahmen dieser Arbiet wurden die bestehenden Quellen ergänzt und in einer Datenbank ([BSK98]) zusammengetragen. Dabei wurden aus polarökologischer Sicht folgende morphologische Gruppen abgedeckt:

- elliptoid
- zylindrisch
- kokkoid
- filamentös

Untersuchungen über Struktur und Zusammensetzung spezifischer Bodenalgengesellschaften, ausgehend von genauen qualitativen Analysen, liegen vor allem aus ariden Regionen vor. Für detallierte Angaben über den Informationsgehalt der Datenbank, das Datenmaterial und die (historischen) Quellen wird auf den entsprechenden Abschnitt des zweiten Kapitels verwiesen (2.1.1).

Kapitel 2

Methodik der Vorverarbeitung

In diesem Kapitel werden u.a. die Schritte erläutert, die notwendig sind, um aus einer *rohen* Datenmenge (eine Datenbank bestehend aus Bilderbeispielen) Merkmalsvektoren zu extrahieren. Diese Vektoren stellen eine komprimierte Information des urspünglichen Datums dar und eignen sich, aufgrund ihres Informationsgehaltes und (z.T.) invarianten Eigenschaften, als Eingabe für das Training eines Neuronalen Netzes, das die Separierbarkeit von acht Algen–Klassen lernen soll. Dieser Prozess wird in der Bildverarbeitung und Mustererkennung unter dem Begriff *Vorverarbeitung* geführt ([Jäh97], [Hab95]). Neben den *morphologischen Operatoren* ([Ser82]) werden die mathematischen Grundlagen der *Fourierdeskriptoren* ([Arb90], [Bur79]) und der *Momente* (nach [Hu62]) erörtert, wobei die Betonung auf deren invarianten Eigenschaften und den damit verbundenen Möglichkeiten liegt.

2.1 Datenaufbereitung (von der Alge zum Bild)

2.1.1 Datenmaterial

Um Bilder mit Rechensystemen bearbeiten zu können, müssen sie in Datenformate umgesetzt werden, die rechnerkompatibel sind. Diese Umsetzung heißt Digitalisierung. Für die photographischen Vorlagen der zu klassifizierenden Algen und für die Strichzeichnungen wurden hierzu Zeilenabtaster (Scanner) mit unterschiedlicher technischer Realisierung der Abtastung eingesetzt. In den folgenden Abschnitten werden die Quellen und die Datenbank näher behandelt.

Die Datenbank

Auf Basis einer vom Institut für Polarökologie der Universität Kiel zur Verfügung gestellten Bildauswahl wurde die der Arbeit zugrunde liegende Datenbank erstellt ([BSK98]). Sie repräsentiert das derzeitig verfügbare phänomenologische Spektrum subantarktischer Algen. Die Bildvorlagen sind als Grauwertbilder in einer mSQL-Datenbank abgelegt worden. Mit ca. 2000 Datensätzen bildet diese Datenbank ein Referenzsystem, das sehr unterschiedliche Quellen beinhaltet. Der Informationsgehalt dieser Datenbank umfaßt:

- Non-marine Algen (Böden und limnische Lebensräume [EG95])
- Informationen über
  - Spezies
  - Fundort
  - Habitate
  - Autorenreferenzen
  - Bildnachweis (Fotografie oder Zeichnung)

Leider sind nicht alle diese Bilder für die Klassifikationsaufgabe geeignet. Lediglich die Bilder, die nur eine einzige Zelle enthalten, kommen für das Training der Netze im Frage. Bildvorlagen, die mehrere Algen in einem Bild enthalten oder verschiedene Stadien einer Meyose (bzw. eines ähnlichen Teilungsprozesses) dokumentieren, sind unbrauchbar, da der Lernalgorithmus sich hauptsächlich auf die Form, auf die Morphologie stützt. Nach einer Selektion des Datenmaterials, basierend auf dem einmaligen Vorkommen einer Algenform in einem Bildbeispiel und durch den Verzicht auf Bilder, die Algenpopulationen beinhalten, ist das für das Training der Netze erforderliche Bildmaterial auf ca. 300 Muster, die die oben genannten Kriterien erfüllen, reduziert worden. Die selektierten Klassenrepräsentanten sind hinsichtlich ihrer Morphologie und polarökologischen Taxonomie in 8 Klassen eingeteilt worden. Einige Beispiele zu den Algen–Spezien (mit *sp.* abgekürzt) für die einzelnen Klassen sind in der folgenden Zusammenfassung aufgeführt:

- **Klasse 1** –die Kugelförmigen– beinhaltet u.a. Algen der Gattung *Cyclotella sp., Chlorella sp., Melosira sp.* oder *Trochisciopsis*
- **Klasse 2** –die Ovalen, Kokkoiden– beinhaltet u.a. Algen der Gattung *Coconeis sp.* oder *Surirella sp.*
- **Klasse 3** –die Zylindrischen– wird vertreten z.B. durch *Achnanthes sp.*
- **Klasse 4** –die Elliptoiden– z.B. *Cosmarium sp.*
- **Klasse 5** –die Asymmetrischen– z.B. *Eunotia sp.*

- **Klasse 6** –die Sichelförmigen– u.a. durch *Closteriopsis sp.* und *Chlorolobion sp.* repräsentiert
- **Klasse 7** –die Filamentösen– u.a. *Nitzschia sp.*
- **Klasse 8** –die länglichen Blaualgen– repräsentiert u.a durch *Cyanobakterien* in verschiedenen Formen

Um die unterschiedlichen Morphologien und Algengattungen besser zu verdeutlichen, werden zu den einzelnen Klassen einige Repräsentanten bildlich dargestellt:

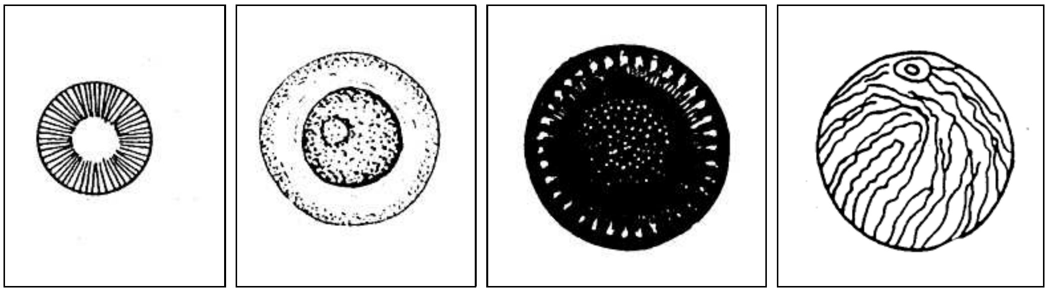


Abbildung 2.1: *Repräsentanten der* **Klasse 1**, *z.B. Chlorella sp. (2. von links) oder Trochisciopsis sp. (ganz rechts)*

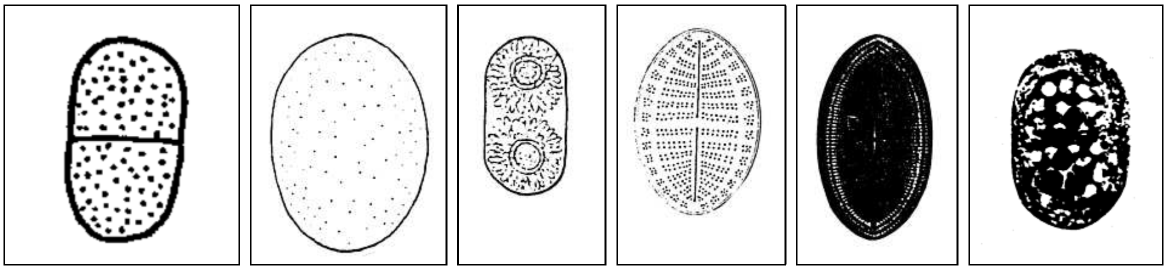


Abbildung 2.2: *Repräsentanten der* **Klasse 2**, *z.B. Cocconeis sp. (1. von links) oder Surirella sp. (3. von links)*

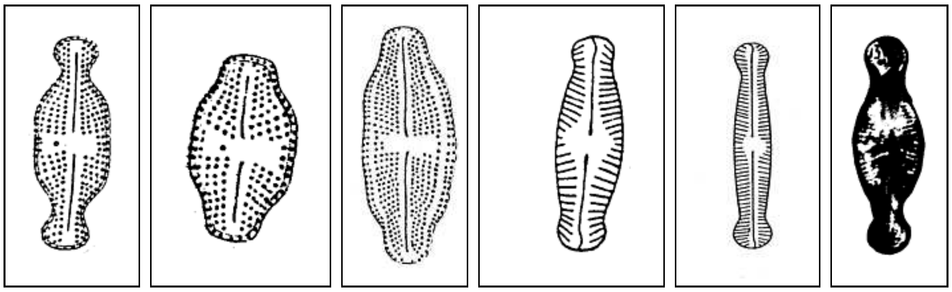


Abbildung 2.3: *Repräsentanten der* **Klasse 3**, *z.B. Achnanthes sp. (1. von links)*

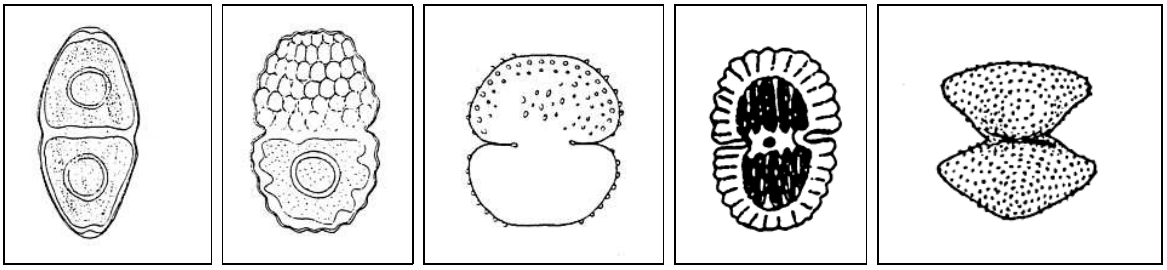


Abbildung 2.4: *Repräsentanten der* **Klasse 4**, *z.B. Cosmarium sp. (4. von links)*

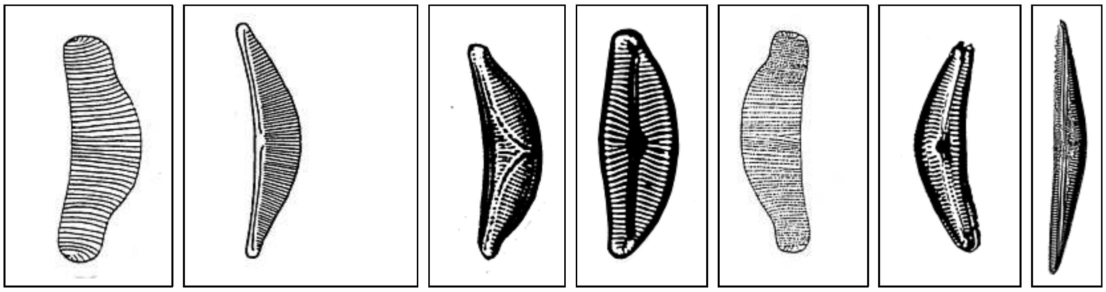


Abbildung 2.5: *Repräsentanten der* **Klasse 5**, *z.B. Eunotia sp. (1. von links)*

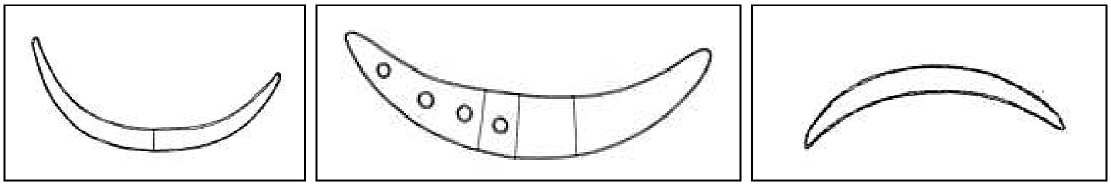


Abbildung 2.6: *Repräsentanten der* **Klasse 6**, *z.B. Closteriopsis sp. oder Chlorolobion sp.*

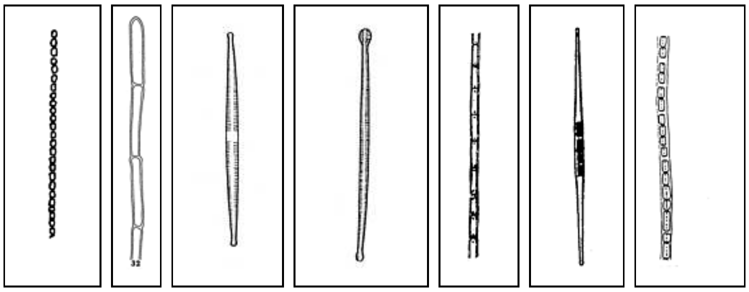


Abbildung 2.7: *Repräsentanten der* **Klasse 7**, *z.B. Nitzschia sp.*

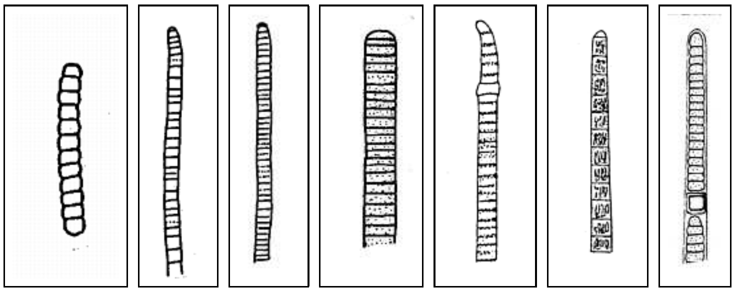


Abbildung 2.8: *Repräsentanten der* **Klasse 8**, *Cyanobakterien in verschiedenen Formen*

Die Quellen

Die Spannweite der in der Datenbank abgelegten Bilder umfasst sowohl handgezeichnete Bilderbeispiele aus dem 19. Jahrhundert, als auch modernste elektonenmikroskopische Aufnahmen. Als erste Beschreibung einer Bodenalge gilt die bei Dillenius (1741) abgebildete *Nostosc commune*, eine weitverbreitete und seit dem Mittelalter wohlbekannte aerophytische Cyanobakterie. Seit Beginn des 19. Jahrhundert wurden bei algentaxonomischen Untersuchungen auch Bodenalgen berücksichtigt. Einige dieser Quellen sind von unschätzbarem Wert, wenn man bedenkt, daß ein Polarforscher zu seiner Zeit bei weitem nicht die Möglichkeiten der Datenaufnahme gehabt hat, die dem heutigen Forscher zur Verfügung stehen. Einige der Zeichnungen sind in detailtreuer und mühseliger Handarbeit angefertigt worden und tragen heute noch dazu bei, die Kategorisierung der Algen, die mit dem Elektronenmikroskop aufgenommen worden sind, nach ihrer Struktur und Beschaffenheit einzuordnen und zu vergleichen. Zahlreiche nationale Südpolarexpeditionen europäischer Hegemonialstaaten Ende des 19. Jahrhunderts/Anfang des 20. Jahrhunderts, die Deutsche Südpolarexpedition 1901 – 1903 auf dem Forschungsschiff *Gauss* ([Wil24]) und die Norwegische Polarexpedition 1893 – 1896 geleitet von Fridtjof Nansen ([Nan95]) haben zu der Vervollständigung dieser Quellen beigetragen. So wurden im Rahmen dieser Arbeit vor allem die Abbildungen aus [Car13], [Wil24], [PHR91], sowie [SPPW79] digital verfügbar gemacht. Darüber hinaus floß auch das von [BB93] im Rahmen mehrerer Antarktisexpeditionen gesammelte Bildmaterial in die Datenbank ein.

Internet

Um den Zugriff auf dieses Datenmaterial auch anderen Arbeitsgruppen (Prof. Dr. Komarek in Brno und Dr. Cavacini/Prof. Fumanti an der Universita *La Sapienza*, Dipart. di Biologia Vegetale in Rom) und Interessierten aus aller Welt zu ermöglichen, haben wir sie in Form von html-Seiten via Internet zugänglich gemacht. Sie sind wie folgt gegliedert und unter den angegebenen Webseiten abrufbar:

- Die Gesamtdatenbank (ca. 2000 Eintragungen auf 215 Übersichtsseiten mit jeweils neun Algen pro Seite) ist unter der folgenden URL–Adresse zu finden:

  http://www.ks.informatik.uni-kiel.de/˜ rkl/algen_global/index.html

- Die für die Klassifikationsaufgabe selektierte Datenbank (ca. 300 Eintragungen auf 71 Übersichtsseiten mit jeweils neun Algen pro Seite) ist unter der URL–Adresse:

  http://www.ks.informatik.uni-kiel.de/˜ rkl/algen_select/test3.html

- Die endgültige Gliederung in acht Klassen mit den entsprechenden Verweisen auf die einzelnen Klassen findet sich unter:

http://www.ks.informatik.uni-kiel.de/˜ rkl/Algen_klassen.html

- Die Übersicht ist unter:

  http://www.ks.informatik.uni-kiel.de/˜ rkl/index.html abrufbar.

2.1.2 Vorverarbeitung

Bei der Vorverarbeitung von digitalisierten Bildern werden Techniken wie Bildverbesserung, geometrische Entzerrung oder digitale Filterung eingesetzt. Auch die Verwendung von speziellen Speicherungstechniken oder die Berechnung von geeigneten Merkmalen zur Bildsegmentierung kann zum Bereich der Bildvorverarbeitung gerechnet werden.

Im Rahmen dieser Diplomarbeit wurde die gesamte Vorverarbeitung mit Hilfe der visuellen Programmieroberfläche **Cantata2.0** unter **Khoros** realisiert. Einzelne Komponenten (Glyphs) werden in Form von Pfeildiagrammen zu einer *Workspace* miteinander verbunden, die dann gemeinsam einen Rechenvorgang, einen Berechnungsablauf darstellen.

Diese Plattform bietet auch die Möglichkeit nebst vorhandenen Tools und Routinen, eigene C-Programme zu integrieren. Damit ist eine homogene Programmiermethodik entstanden, die sich sowohl vorhandene Implementierungen zunutze macht (z.B. das Konvertieren aus einem Bildformat ins andere), als auch das Integrieren selbstentwickelter Programme ermöglicht (z.B. die äquidistante Abtastung der Kontur als Vorstufe der Fouriertransformation, um die Rotationsinvarianz zu gewährleisten).

Die Originalbilder sind als *.gif*–Dateien abgelegt, die je nach Auflösung des Scannvorganges und Beschaffenheit der Vorlage unterschiedlich groß ausfallen. Aus diesem Grunde werden sie als erstes in ein einheitliches Format gebracht. Die Größe der Bilder beträgt dann 512×512 Pixel. Dies wird realisiert mit Hilfe einer *Workspace* in **Cantata2.0**, wobei das Originalbild in einen neutralen Hintergrund (Grauwert 0) eingefügt wird. Die Struktur und die Proportionen der Alge werden beibehalten, es ändert sich lediglich der *Rahmen*. Die Wirkungsweise dieser *Workspace* wird in der Abbildung 2.9 veranschaulicht.

Dies ist deswegen notwendig, weil anschließend die morphologischen Operatoren **Opening** und **Closing** (siehe 2.1.2) angewendet werden, um die Struktur zu füllen. Diese Operationen werden realisiert unter Verwendung eines Strukturierungselementes (siehe 2.1.2), was bei einem größeren Diameter zu einer Überschneidung der bearbeiteten Algenform mit dem zu nahe liegenden Rahmen des gesamten Bildes führen kann. Dies kann man vermeiden, indem man einen bestimmten Mindestabstand zwischen abgebildeter Alge und Bildrahmen einhält.

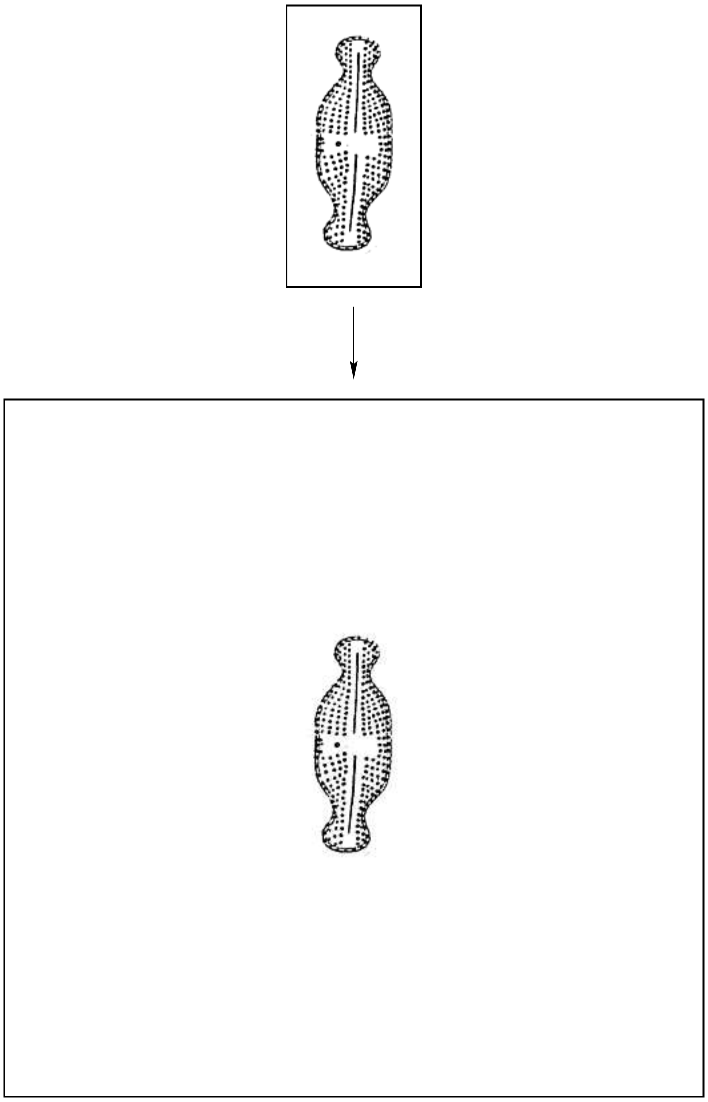


Abbildung 2.9: *Originalbild eingefügt in einer* 512 × 512 *Umgebung. Die Objektgröße bleibt konstant, es ändert sich lediglich die Bildgröße.*

Binarisierung

Bei der Digitalisierung von Bildern fallen in der Regel große Datenmengen an. So benötigt z.B. ein Bild mit $512 \times 512 = 262144$ Bildpunkten und 8 Bit Farbtiefe pro Pixel bereits 256 KByte. In manchen Anwendungsbereichen, z.B. bei der digitalen Aufbereitung von Strichzeichnungen, ist die Bildgröße von 512×512 Pixel nicht ausreichend. Es muß aufgrund der detailreichen Vorlagen mit einem sehr feinen Raster gearbeitet weden, was nicht selten Bilder mit einigen tausend Bildzeilen und einigen tausend Pixeln pro Zeile zur Folge hat. Es stellt sich daher die Frage, wie digitalisierte Bilder in modernen Rechensystemen gespeichert werden und welche Möglichkeiten es gibt, diese großen Datenmengen mit Methoden der Datenreduktion und Datenkompression zu verringern.

Die Binärbilderzeugung ist vor allem in der Mustererkennung eine wichtige Methode zur

Bildsegmentierung. Bei der Binarisierung wird aus einem Originalbild S mit einer Grauwertmenge von 256 $(128, 64, 32...)$ verschiedenen Grauwerten ein Bild S' mit nur zwei Grauwerten erzeugt. Auf diese Weise kann ein Bild, das ein helles Objekt auf einem dunklen Hintergrund enthält, in die Bildbereiche **Hintergrund** und **Objekt** eingeteilt werden. In der Regel werden diese beiden Grauwerte mit einem Bit, also mit den Zahlen 0 und 1, dargestellt.

Es sei S das originale Grauwertbild. Dann wird das Binärbild S' gemäß

$$S \mapsto S' : \; s'(x,y) = \begin{cases} 1 & , \quad \text{falls } s(x,y) > c, \\ 0 & , \quad \text{sonst} \end{cases} \tag{2.1}$$

berechnet.
Damit erhält man ein binäres Bild, indem etwa der Hintergrund durch den Grauwert 0 und der Bereich des Objektes durch den Grauwert 1 kodiert ist.

Die Konstante c ist dabei ein **Schwellwert**, der bildabhängig ermittelt wird. Er kann z.B. aus dem Histogramm (Grauwertverteilung eines Bildes) von S abgeleitet werden. Da der Schwellwert für alle Bildpunkte derselbe ist, wird die obige Formel als *Binarisierung mit statischem (fixem) Schwellwert* bezeichnet. Bei vielen Bildvorlagen ist die Wahl eines festen Schwellwertes nicht sinnvoll, z.B. bei photographischen Vorlagen, bei denen durch die Einflüsse des verwendeten Objektivs Randabschattungen (Vignettierungen) auftreten. Hier muß der Schwellwert ortsabhängig von der Position (x, y) des jeweiligen Bildpunktes ermittelt werden. Da solche Bildvorlagen in der Datenbank nur in sehr geringer Anzahl vorkommen (etwa 10%), wurde auf das Verfahren der *dynamischen (adaptiven) Binärbilderzeugung* verzichtet.

Bei einer originalen Grauwertmenge $G = \{0, 1, ..., 255\}$ ergibt die Binarisierung einen Datenreduktionsfaktor von $1 : 8$ oder 0.125. Für ein Bild mit 512×512 Bildpunkten werden also statt 256 KByte nur 32 KByte zur Speicherung benötigt.

In der Regel ist es zu empfehlen, die Grauwerte vor der Segmentierung mit einem Tiefpaßoperator zu glätten, um zu erzwingen, daß sich die beiden Regionen nicht allzu verästelt ausprägen. Dabei ist es allerdings zu bedenken, daß durch einen Glättungsoperator auch wesentliche Bildinformation verloren gehen kann.

Insbesondere bei der Verarbeitung von Bildern mit medizinischen Präparaten wie Blutzellen oder Chromosomen aber auch bei der Untersuchung von Werkstücken im Rahmen der Qualitätskontrolle wird derzeit intern meistens mit Binärbildern gearbeitet, es ist aber ein Trend zur Verwendung multisensorieller Grauwertbilder zu beobachten.

Einfache morphologische Operatoren

Der Name *morphologischer Operator* stammt aus der Morphologie, welche in der Biologie und den Geowissenschaften die Form von Objekten beschreibt. Ein morphologischer Operator ist also ein Operator, der die Form von Objekten beeinflußt. In welcher Weise dies geschieht und welche verschiedene Arten es gibt, soll im folgenden näher erläutert werden.

Die morphologischen Operatoren können dazu verwendet werden, kleine Löcher und Risse zu füllen oder kleine Objekte zu eliminieren. Die Größe des Strukturelementes bestimmt das Ergebnis.

Die **Erosion** und die **Dilatation** sind elementare morphologische Operatoren aus denen sich komplexere Operatoren, wie z.B **Opening** und **Closing** erzeugen lassen.

Einer der Ansätze solche Operatoren zu definieren besteht darin, sie als Mengenoperationen auf Bildpunkte anzusehen. Dazu betrachten wir A als Menge aller Pixel der Bildmatrix, die ungleich Null sind. B ist die Menge der Maskenpixel ungleich Null. Die Größe der Maske bestimmt die Wirkung des Operators. Daher wird die Maske oft als Strukturelement bezeichnet. Mit $B_p := B + \{p\}$ bezeichnet man die mit ihrem Referenzpunkt (meistens–aber nicht notwendigerweise–dem Zentrum) zum Bildpunkt p verschobene Maske ([Som97]).

Die folgenden Definitionen basieren auf den Mengenverknüpfungen, die unter dem Namen **Minkowski-Addition**, bzw. **Minkowski-Subtraktion** bekannt sind (siehe [Ser82]). Die Definitionen hierfür lauten:
Seien A und B zweidimensionale Binärobjekte, dann heißen

- $A + B := \{a + b \mid a \in A, b \in B\}$
  Minkowski-Addition und
- $A - B := \{a - b \mid a \in A, b \in B\}$
  Minkowski-Subtraktion

Die **Erosion** $\mathcal{E}$ wird dann als

$$\mathcal{E} = A \ominus B := \{p \mid \forall b \in B \; \exists a \in A : p = a + b \in A\} = \{p \mid B_p \subset A\} \tag{2.2}$$

definiert.

Eine Erosionsoperation wirkt z.B. wie ein Netz mit Löchern in der Form des Strukturelementes B. Alle Objekte, die durch die Löcher passen, rutschen hindurch und werden aus dem Bild eliminiert. Ein Objekt verbleibt nur dann im Bild, wenn das Strukturelement zumindest an einem Punkt völlig von Objekt-Bildpunkten bedeckt ist, sonst verschwindet es. Das strukturbildende Element B muß nicht notwendigerweise zusammenhängend sein und der Aufpunkt (also der Koordinatenursprung) von B muß nicht Element von B sein (Beispiele siehe in Abbildung 2.10).

○ ○ ● ○ ○	Elementarraute
○ ○ ● ○	Elementarrechteck
○ ○ ○ ○ ● ○ ○ ○ ○	8-Nachbarn
○ ○ ○ ○ ● ○ ○ ○ ○	schräges Element

Abbildung 2.10: *Beispiele für strukturierende Elemente. Mit* $\bullet$ *ist der Bezugspunkt und mit* $\circ$ *sind die Nachbarn markiert.*

Die *Workspace*, die die Arbeitsweise des Erosionsoperators verdeutlicht, ist in der Abbildung 2.11 zu sehen.

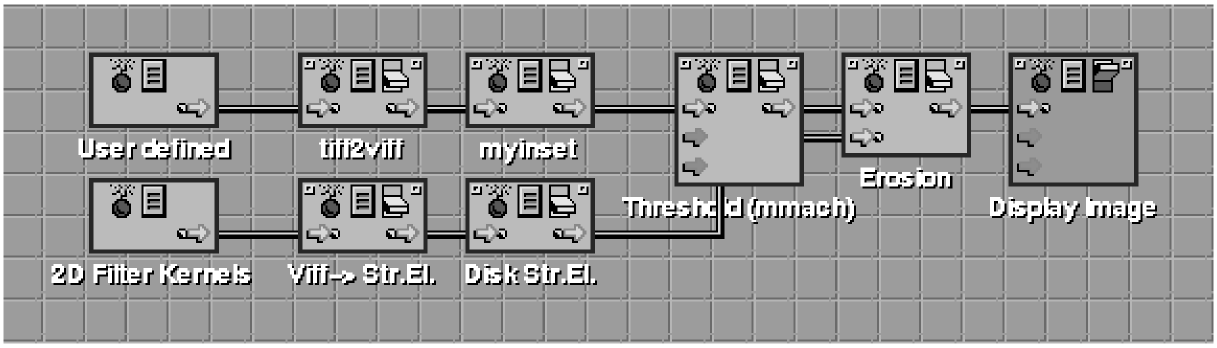


Abbildung 2.11: *Der* **Erosionsoperator** *realisiert mit Hilfe einer Workspace unter der visuellen Programmieroberfläche in* **Cantata**.

An den Beispielen in der Abbildung 2.12 kann man erkennen, daß die *Unreinheiten* im Hintergrund des Bildes verschwunden sind, die Objekte werden stetig ausgefüllt und werden dadurch außen größer und dicker, was eine Aufhebung der klaren Linien zwischen den Objekten nach sich zieht. Die Objekte würden nach einer großen Anzahl an Iterationen zu einem Objekt konglomerieren (verklumpen). Eine Wirkung des Erosionsoperators nach 50 Iterationen ist in der Abbildung 2.13 zu sehen.

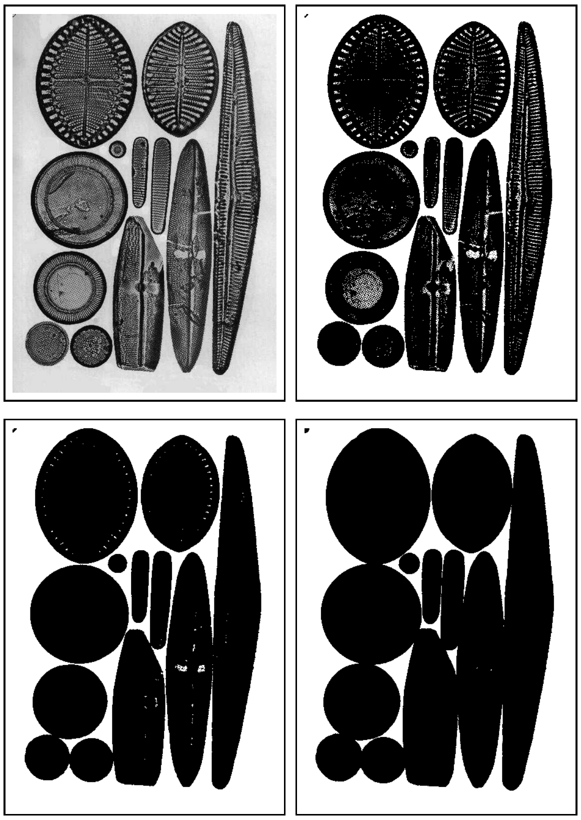


Abbildung 2.12: *Erosion mit einen Disk Kernel* 3×3. *Von links oben nach rechts unten: das Originalbild, Erosion nach* 2, 6 *und* 14 *Iterationen.*

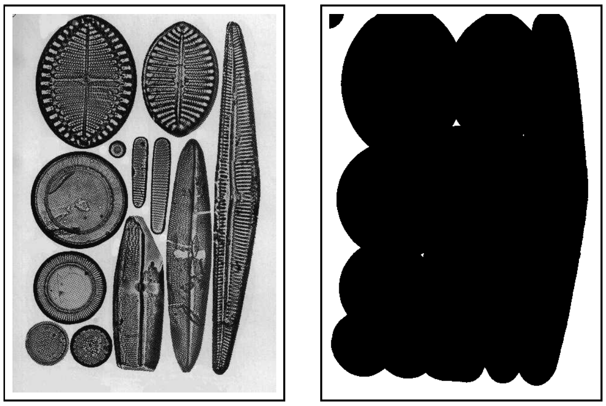


Abbildung 2.13: *Konglomerat der Bildobjekte nach* 50 *Iterationen. Die Konturen der einzelnen Objekte sind verschwommen. Der Erosionsoperator wurde zu häufig angewendet.*

Die **Dilatation** $\mathcal{D}$ ist definiert als

$$\mathcal{D} = A \oplus B := \{p \mid \exists a \in A, b \in B : p = a + b\} = \{p \mid B_p \cap A \neq \emptyset\}. \tag{2.3}$$

Man kann also sagen, daß die Minkowski-Addition und der Dilatationsoperator gleich sind.

Eigenschaften der Operationen **Dilatation** und **Erosion**:

- **Kommutativität** und **Assoziativität**
  Morphologische Operatoren sind im allgemeinen nicht kommutativ:

  $$B_1 \oplus B_2 = B_2 \oplus B_1 \tag{2.4}$$

  aber

  $$B_1 \ominus B_2 \neq B_2 \ominus B_1. \tag{2.5}$$

  Wir sehen, daß die Erosion nicht kommutativ ist, wenn wir den Spezialfall betrachten, daß $B_1 \supset B_2$. Dann ergibt die Erosion von B_2 durch B_1 die leere Menge. Werden jedoch Erosions- und Dilatationsmasken nacheinander auf das gleiche Bild A angewandt, sind sie kommutativ:

  $$(A \oplus B_1) \oplus B_2 = A \oplus (B_1 \oplus B_2)(A \ominus B_1) \ominus B_2 = A \ominus (B_1 \ominus B_2)$$

  Generell ist die kaskadierte Operation mit k Strukturelementen $B_1, B_2, ..., B_k$ der Operation mit dem Strukturelement $B = B_1 \oplus B_2 \oplus ... \oplus B_k$ äquivalent. Wir können also große Strukturelemente ebenso zerlegen, wie wir es bei linearen verschiebungsinvarianten Operatoren auch tun. Ein Beispiel hierfür ist die Konstruktion separierbarer Strukturelemente aus horizontalen und vertikalen Elementen $B = B_x \oplus B_y$. Schwieriger zu erhalten sind jedoch isotrope, d.h. kreisförmige Strukturelemente. Das Problem liegt darin, daß die Dilatation horizontaler und vertikaler Strukturelemente stets zu einem rechteckigen Strukturelement, nicht aber zu einer kreisförmigen, runden Maske führt. Wir haben uns bei der Verwendung eines Strukturelementes im Rahmen der Erosion- und der Opening-Operation für ein kreisförmiges Strukturelement entschieden, das wir durch eindimensionale Strukturelemente angenähert haben, die in mehreren Richtungen und nicht entlang der Achsen laufen.

- **Monotonie**
  Erosion und Dilatation sind monotone Operationen: Die Monotonieeigenschaft bedeutet, daß die Teilmengenrelationen in Bezug auf Erosion und Dilatation invariant sind.

  $$\begin{aligned} A_1 \subseteq A_2 &\rightsquigarrow A_1 \oplus B \subseteq A_2 \oplus B \\ A_1 \subseteq A_2 &\rightsquigarrow A_1 \ominus B \subseteq A_2 \ominus B. \end{aligned} \tag{2.6}$$

- **Distributivität**
  Lineare verschiebungsinvariante morphologische Operatoren sind distributiv bezüglich der Addition. Für Erosion und Dilatation muß man zwischen der Vereinigungs- und der Schnittmenge zweier Bilder A_1 und A_2 unterscheiden:

$$\begin{aligned}(A_1 \cap A_2) \oplus B &\subseteq ((A_1 \oplus B) \cap (A_2 \oplus B)) \\ (A_1 \cap A_2) \ominus B &= ((A_1 \ominus B) \cap (A_2 \ominus B))\end{aligned} \tag{2.7}$$

  und

$$\begin{aligned}(A_1 \cup A_2) \oplus B &= ((A_1 \oplus B) \cup (A_2 \oplus B)) \\ (A_1 \cup A_2) \ominus B &\supseteq ((A_1 \ominus B) \cap (A_2 \ominus B)).\end{aligned} \tag{2.8}$$

  Die Erosion ist also distributiv bezüglich der Schnittmengenoperation und die Dilatation bezüglich der Vereinigungsoperation.

- **Dualität**
  Erosion und Dilatation sind *duale* Operatoren. Die Erosion mit der Negation eines binären Bildes ist äquivalent mit der Negation des Originalbildes. Analoges gilt bei der Vertauschung der beiden Operationen:

$$\begin{aligned}\overline{A} \ominus B &= \overline{A \oplus B} \\ \overline{A} \oplus B &= \overline{A \ominus B}.\end{aligned} \tag{2.9}$$

  Die vollständigen Beweise hierfür finden sich in [Ser82].

Komplexe morphologische Operatoren

Opening $\mathcal{O}$

Bei diesem morphologischem Operator handelt es sich um eine Hintereinanderausführung der vorangegangenen elementaren Operationen Erosion und Dilatation. Aus diesem Grunde war es notwendig auf die Eigenschaften, wie Kommutativität, Assoziativität und Distributivität näher einzugehen, da ansonsten die Hintereinanderausführung der elementaren morphologischen Operatoren nicht als selbstverständlich hätte angenommen werden können.

Die Erosion wird zur Entfernung kleiner Objekte verwendet. Sie hat jedoch den Nachteil, daß alle im Bild verbleibenden Objekte kleiner werden. Durch anschließende Dilatation des Bildes mit dem gleichen Strukturelement läßt sich dieser Effekt vermeiden, da die Dilatation die Objekte größer, ausgedehnter werden läßt. Die Kombination aus Erosion und anschließender Dilatation wird **Opening** $\mathcal{O}$ genannt und folgendermaßen definiert:

$$\mathcal{O} = A \circ B := (A \ominus B) \oplus B. \tag{2.10}$$

Das Opening siebt alle Objekte aus, die das Strukturelement in keinem Punkt vollständig enthalten, vermeidet jedoch die Größenreduktion (wie es bei der Erosion der Fall ist) aller Objekte durch die anschließende Dilatation. Sie ist daher ideal zur Beseitigung von Linien, deren Breite geringer ist als der Durchmesser des Strukturelementes. Die Objekte erscheinen dadurch kompakter, da die Objektgrenzen glatter sind. Die *Workspace*, die eine Opening-Operation realisiert, ist in der Abbildung 2.14 zu sehen, ihre Wirkungsweise als Füllalgorithmus in den darauf folgenden Beispielen aufgezeigt.

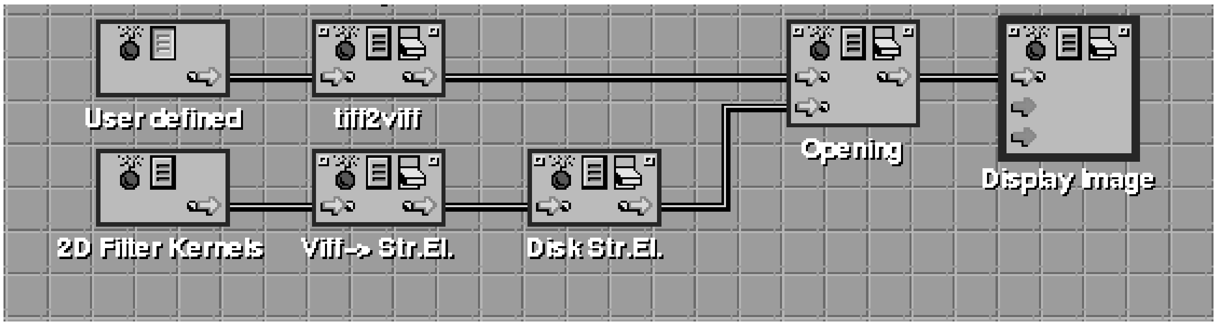


Abbildung 2.14: *Der* **Opening–Operator** *realisiert mit Hilfe einer Workspace unter der visuellen Programmieroberfläche in* **Cantata**.

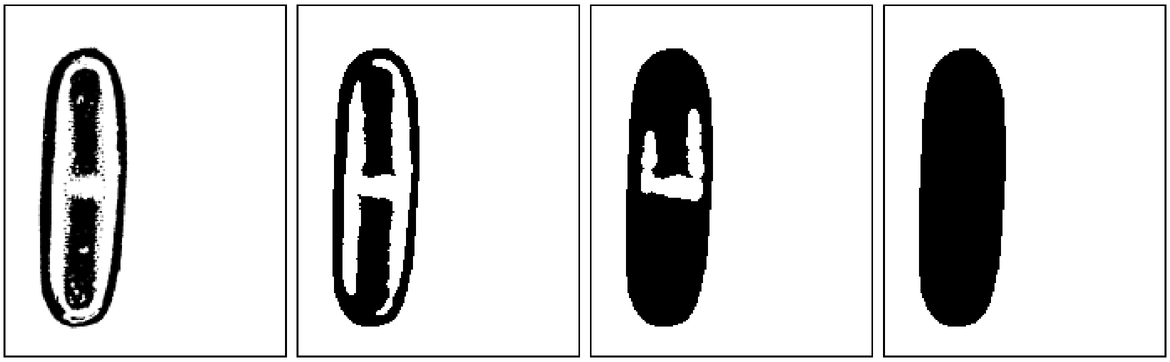


Abbildung 2.15: *Opening als Füllalgorithmus. Von links nach rechts: das Originalbild, Opening nach 14, 16 und 20 Iterationen.*

Das in der Abbildung 2.16 angegebene Exempel ist demonstrativ für die Operation Opening, da später die *Dualität* mit der Operation Closing anhand des dargestellten bzw. des negierten Bildmusters demonstriert wird.

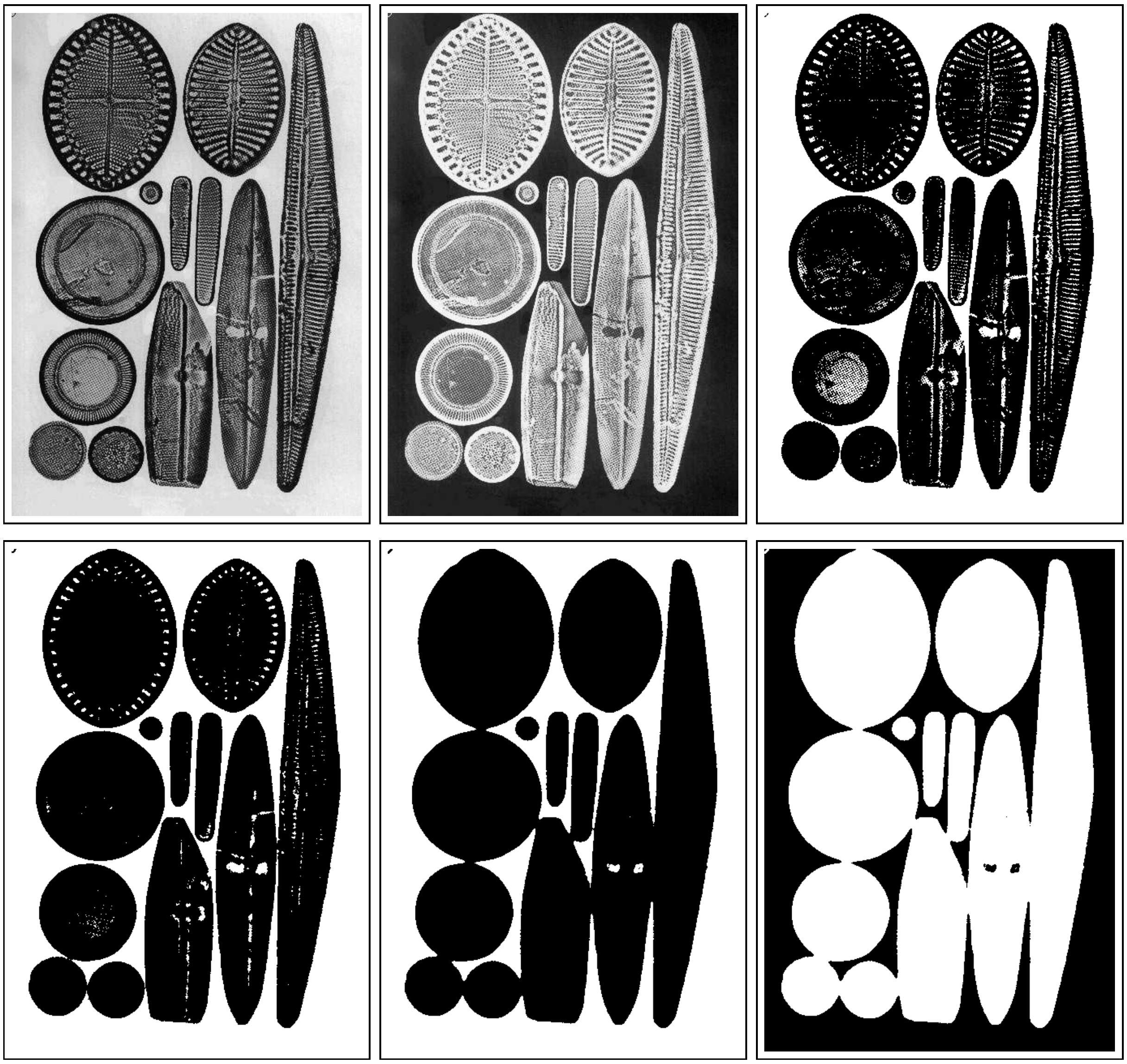


Abbildung 2.16: *Opening als Füllalgorithmus. Von links nach rechts: das Originalbild, das negierte Bild, Opening nach 2, 4 und 10 Iterationen und das negierte Ergebnis*

Closing $\mathcal{C}$

Die morphologische Operation Dilatation vergrößert Objekte und füllt kleine Löcher und Risse aus. Die generelle Vergrößerung der Objekte durch die Größe der Strukturelemente kann durch eine nachfolgende Erosion ausgeglichen werden. Die Kombination von Dilatation und Erosion wird *Closing* genannt und ist wie folgt definiert:

$$\mathcal{C} = A \bullet B := (A \oplus B) \ominus B. \tag{2.11}$$

Die Arbeitsweise dieses Operators sei durch zwei Beispiele verdeutlicht. Die von zwei auf sechs Schritte inkrementierte Anwendung des *Closing*-Operators führt zu einer stufenweisen Auflösung der Form der Zelle. Dies ist in der Abbildung 2.17 deutlich zu sehen.

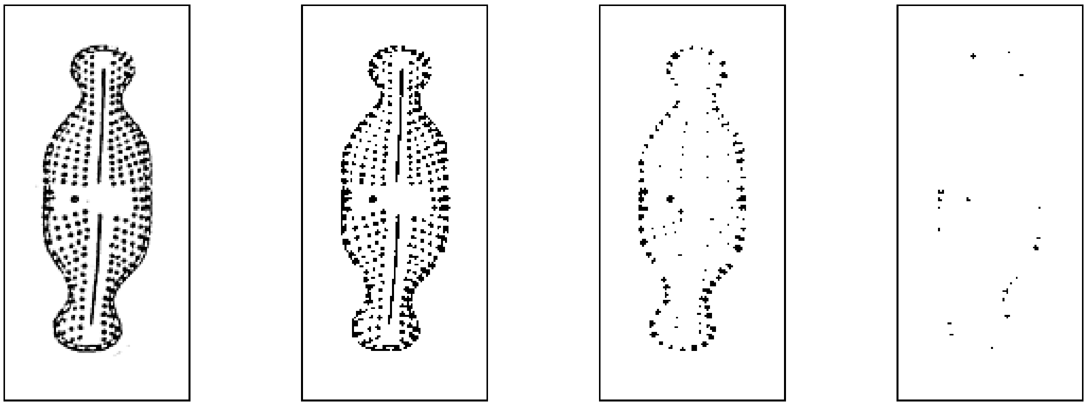


Abbildung 2.17: *Closing-Operator. Von links nach rechts: das Originalbild, Closing nach 2, 4 und 6 Iterationen.*

In der Abbildung 2.18 wird der *Closing*-Operator auf das schon bekannte Bild mit mehreren Objekten angewendet. Die Bilder stellen die Veränderungen nach zwei, vier und zehn Iterationen im Inneren der Zelle dar. Diese Abbildung spiegelt einerseits die morphologische, formverändernde Wirkung dieses Operators wieder, auf der anderen Seite ist sie ein *bildhafter* Beweis für die Dualität, die zwischen den Operatoren *Opening* und *Closing* besteht.

Die Änderungen der Fläche von Objekten duch die unterschiedlichen Operationen läßt sich durch die folgende Beziehung zusammenfassen:

$$A \ominus B \subseteq A \circ B \subseteq A \subseteq A \bullet B \subseteq A \oplus B. \tag{2.12}$$

Eigenschaften der Operationen **Opening** und **Closing**:

- *Opening* und *Closing* sind *idempotente* Operationen:

$$A \bullet B = (A \bullet B) \bullet B \tag{2.13}$$

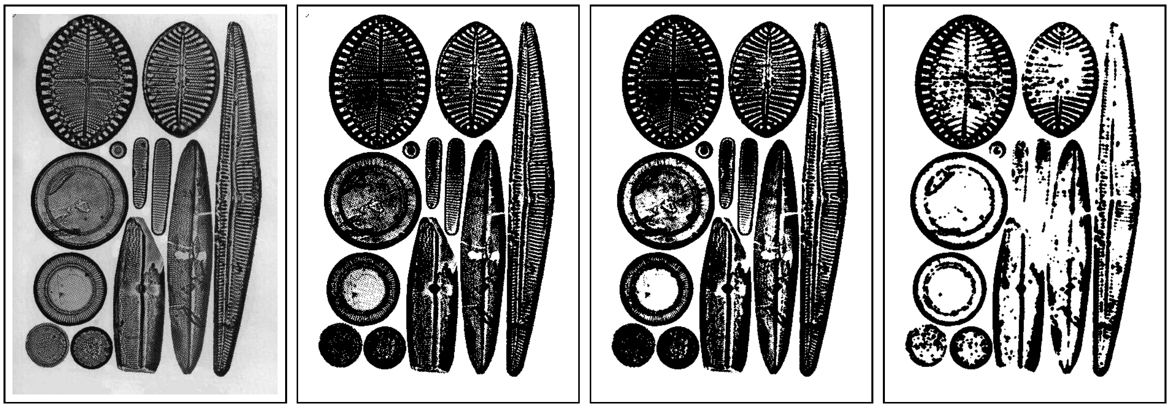


Abbildung 2.18: *Closing-Operator. Von links nach rechts: das Originalbild, Closing nach 2, 4 und 10 Iterationen.*

$$A \circ B = (A \circ B) \circ B.$$

Das bedeutet, daß eine wiederholte Anwendung einer *Closing*– und einer *Opening*–Operation mit dem gleichen Strukturelement keine weiteren Veränderungen bewirkt.

Die Anwendung dieser Operatoren ist auf dem negierten Bild vorgenommen worden, um die Funktionsweise und die Dualität besser zu verdeutlichen. In der Abbildung 2.19 ist als erstes das negierte Originalbild zu sehen, gefolgt von den Ergebnissen der Anwendung nach zwei, vier und zehn Iterationsschritten. Als letztes Beispiel wurde das Bild nach dem zehnten Iterationsschritt noch einmal negiert und als Ergebnis haben wir ein uns schon bekanntes Bild erhalten, dasselbe, was mit der Operation *Opening* auch erreicht wurde. Ein Füllen des Forminneren kann also sowohl durch die Operation *Opening* auf das Originalbild, als auch durch die Operation *Closing* auf das negierte Bild mit anschließender Negierung des Ergebnisses erreicht werden. Als Vergleich hierzu betrachte man die unter der Operation *Opening* angegebene Abbildung 2.16.

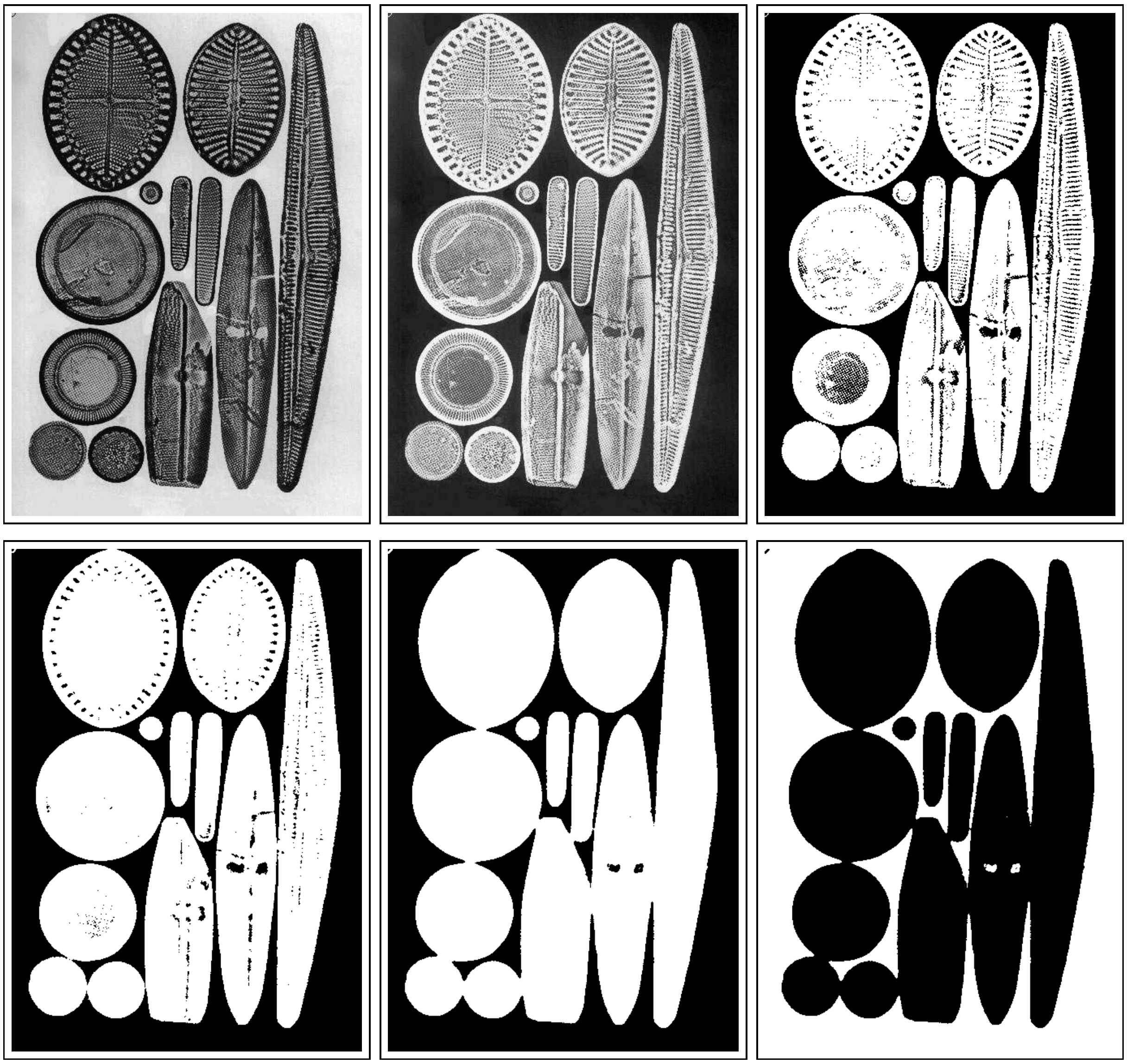


Abbildung 2.19: *Closing-Operator. Von links oben nach rechts unten: das Originalbild, das negierte Bild, Closing nach 2, 4 und 10 Iterationen und das negierte Ergebnis.*

Konturextraktion

Morphologische Operatoren arbeiten auf Binärbildern. Diese kann man in Form einer Matrix darstellen. Die Speicherung der Bildpunkte betreffend, heißt das, daß wir jedes Pixel des Objekts und jeden Hindergrundbildpunkt speichern müssen. Binärbilder können aber viel kompakter die gleiche Information speichern, z.B. in dem man nur die Randpunkte, d.h nur die Kontur abspeichert. Es ist völlig ausreichend, wenn man keine Informationen aus der Textur gewinnen möchte und sich *nur* auf Informationen betreffend der Kontur und der Form konzentriert, also lediglich die Randpunkte speichert. Es lohnt sich ebenfalls zu untersuchen, ob die Extraktion von Formparametern oder ob Operationen, welche die Form eines Objektes verändern, ebenfalls mit dieser kompakten Datenstruktur durchgeführt werden können. Neben der *Lauflängenkodierung* (siehe [Hab91]) und der *Quadtrees* (basierend auf der rekursiven Zerlegung des Raumes in Quadranten) ist der **Kettencode** eines der gängigsten Verfahren zur komprimierten Darstellung der binären Datenstrukturen.

In diesem Abschnitt wird der **Kettencode** als Konturextraktionsverfahren vorgestellt, das aus dem gefülltem (erodiertem) Bild eine 1–Pixel breite Kontur zurückliefert.

Als Eingabe dient das gefüllte Bild. Als Ausgabe erhält man das Konturbild und eine Textdatei, die die Koordinaten der Kontur beinhaltet. Diese werden für die weitere Bearbeitung benötigt.

Die *Workspace* die unter **Khoros** die Konturextraktion realisiert wird in der Abbildung 2.20 dargestellt.

An einem konkreten Beispiel wird die Konturextraktion verdeutlicht, siehe hierzu die Abbildung 2.21.

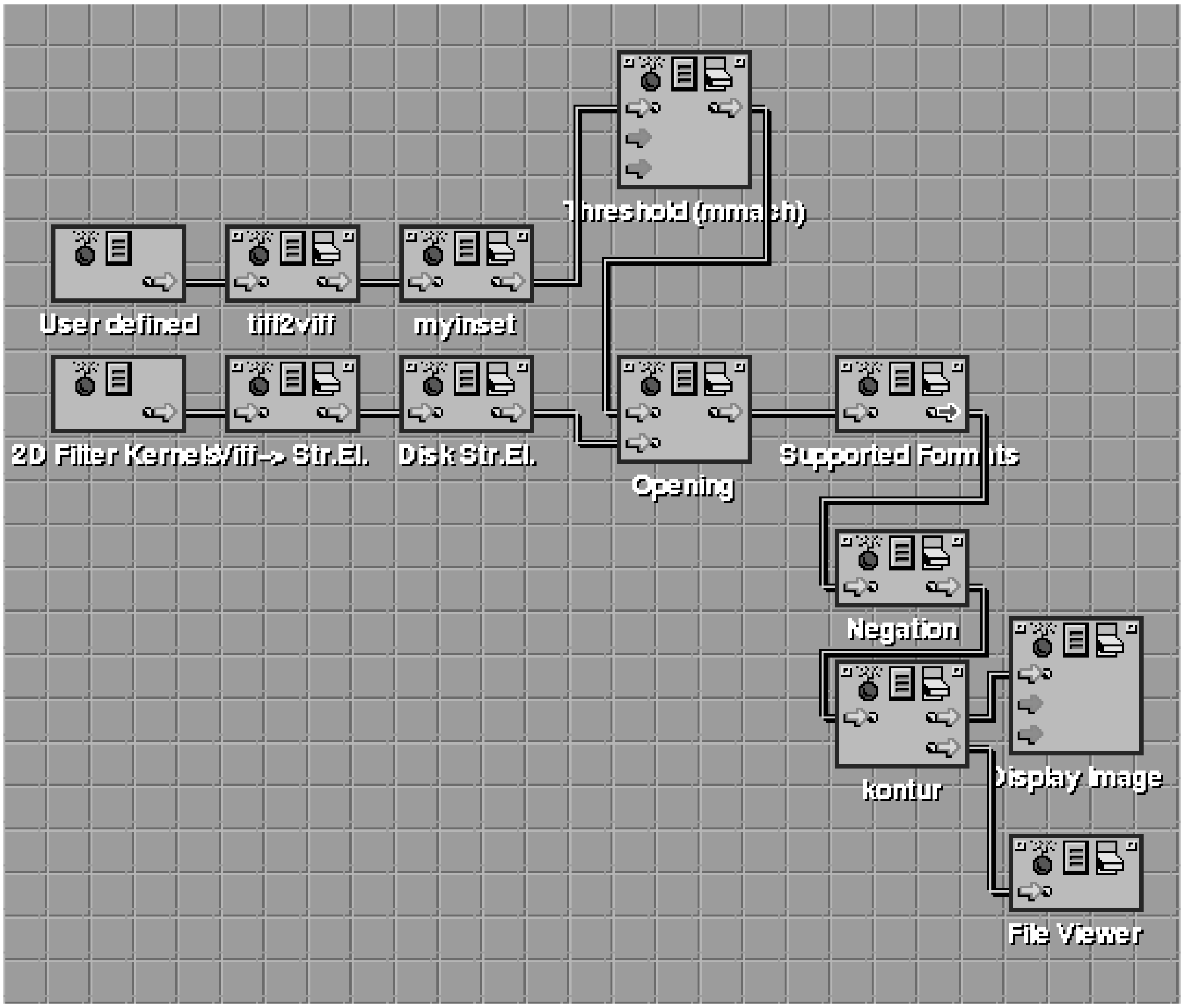


Abbildung 2.20: *Diese Workspace zeigt wie die Kontur extrahiert wird.*

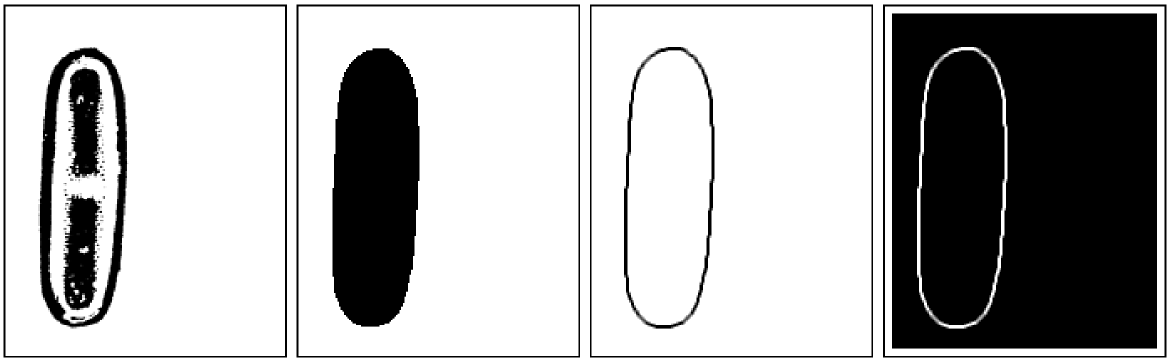


Abbildung 2.21: *Konturextraktion. Von links nach rechts: das Originalbild, Füllen durch Opening, die extrahierte Kontur, invertiertes Bild zur Datenminimierung.*

Zum **Kettencode**:

Der Kettencode ist eine objektbezogene Datenstruktur zur effektiven (und Speicherplatz sparenden) Darstellung des Randes eines Binärobjektes auf einem diskreten Gitter. Statt die Positionen aller Randpixel zu speichern wird folgendermaßen verfahren:

- Wähle einen Startbildpunkt und speichere dessen Koordinaten.
- Wähle einen Algorithmus, der das Bild Zeile für Zeile abtastet (auch meandrische Abtastung möglich). In diesem Fall ist der Startpunkt der oberste linke Bildpunkt des Objektes.
- Folge dem Rand entgegen dem Uhrzeigersinn. Bei einer *8er-Nachbarschaft* gibt es acht, bei einer *4er-Nachbarschaft* gibt es vier mögliche Richtungen, in denen man die Kontur weiterverfolgen kann. Die Nachbarschaftsrelationen sind in der Abbildung 2.22 zu sehen.

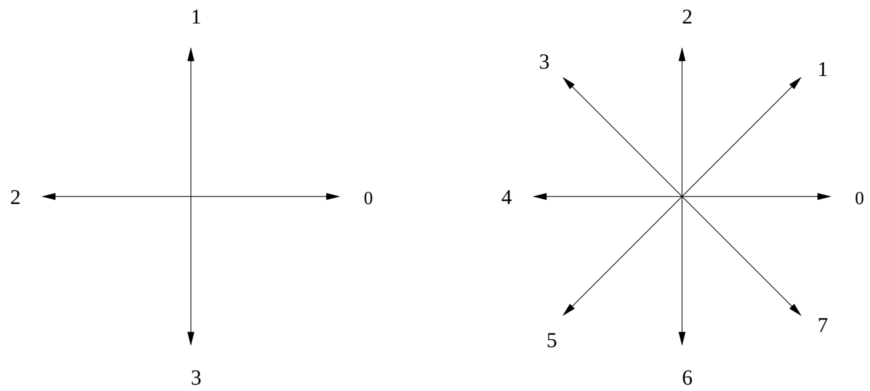


Abbildung 2.22: *Richtungskodierung in einer 4er-Nachbarschaft (links) und einer 8er-Nachbarschaft (rechts)*

- Die Kettencode-Darstellung für die Kontur des Bildes in der Abbildung 2.23 hat folgende Form:
  00000000000000700707777776676766676666667667667667676776776776767676676
  7667667667666676666666666666666666666656666656666566656665655565565655
  565555555565566656665666566665666656566556455454454444453444343434233332
  232222222222222212222122221222212222122212221222212222221222222222221222
  123222222222222222222222222222322222222223222222222322222222322222222232222223
  22222222222222111011001.
- Die Länge der Kontur beträgt in diesem Fall 377.
- Erster Objektpunkt bei den Koordinaten: 245, 172.
- Anzahl der gesetzten Pixel: 88309.

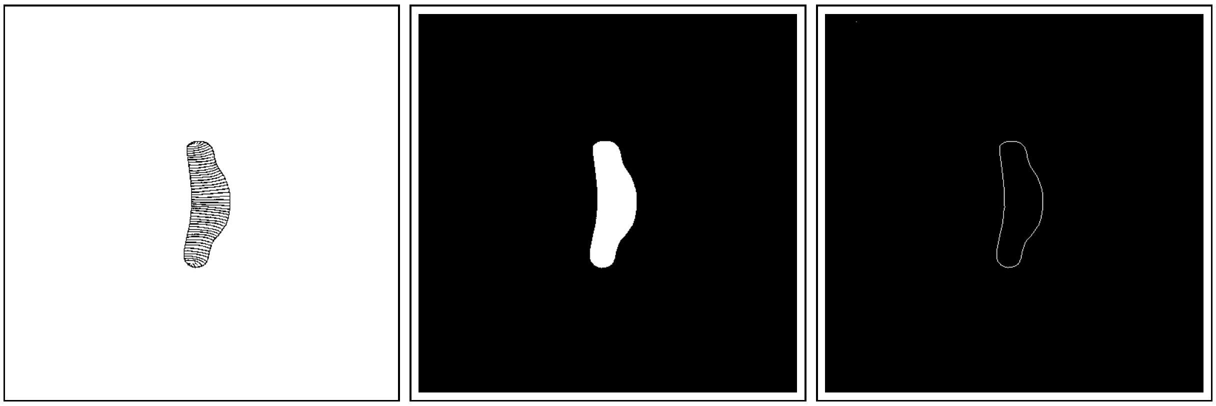


Abbildung 2.23: *Konturextraktion. Von links nach rechts: das Originalbild, Füllen durch Opening und Negation, die extrahierte Kontur.*

Die extrahierte Kontur als Liste, das Konturbild und das gefüllte Bild sind unsere aktuellen auch im Sinne der Datenminimierung noch erhaltenen Informationen über die Ursprungsdaten. In dem folgenden Kapitel werden wir Methoden vorstellen, die u.a. invariante Eigenschaften besitzen und die lediglich bezogen auf die drei genannten Informationsquellen beziehend, eine rotations–, skalierungs– uns translationsinvariante Merkmalsberechnung ermöglichen. Die daraus resultierenden, einzelnen aber auch kombinierten Merkmalsvektoren sollen später (nach geeigneter Vorverarbeitung, siehe 3.5) als Lern– und Testvorlage für die Neuronalen Netze dienen.

2.2 Merkmalsextraktion (vom Bild zum Merkmal)

Von jeder Alge steht uns eine einzige Ansicht zur Verfügung. Somit lösen wir in unseren Berechnungen die Frage der invarianten Eigenschaften für eine 2–dimensionale Problemstellung.

2.2.1 Invarianz

Die Berechnung von Invarianten hat für die Mustererkennung eine zentrale Bedeutung, da bei vielen praktischen Aufgabenstellungen Bilder, Objekte und Figuren oft durch Transformationen *verfälscht* werden. Bei allen bisher besprochenen Verfahren wurde die Anordnung der Bildpunkte in der Bildmatrix S nicht geändert, d.h. die Ortskoordinaten (x, y) der Bildpunkte waren von den Operationen nicht betroffen. Bei der Problemstellung der Klassifikation von Bildern anhand von Merkmalen kann man den Fall haben, daß zwei Bilder desselben Objekts, die jedoch unter verschiedenen Aufnahmebedingungen gemacht wurden, zu einander in Beziehung stehen und zu einer Klasse gehören. Man kann auch den Fall haben, daß zwei Algen derselben Klasse unterschiedlich groß abgebildet worden sind. Die Diskrepanzen innerhalb einer Klasse können dabei folgende sein:

- Objekte sind unterschiedlich groß
- Objekte sind gedreht
- Objekte sind nicht zentriert

Daher besteht die Notwendigkeit, solche Bilder vor weiteren Verarbeitungsschritten einer Transformation der Ortskoordinaten zu unterziehen. Zur Gewährleistung der Deckungsgleichheit ist dazu in der Regel eine Anpassung der Ortskoordinaten der Bilder notwendig. Die grundsätzliche Problemstellung kann wie folgt beschrieben werden: Es sei $S = (s(x,y))$ die Bildmatrix des Originalbildes mit den Ortskoordinaten x, y. Eine Transformation der Ortskoordinaten $S \mapsto S'$ wird beschrieben durch

$$s'(x', y') = s(x, y)\,, \tag{2.14}$$

wobei $x' = f_1(x, y)$ und $y' = f_2(x, y)$ die Abbildungen der Transformation sind. Eine Transformation dieser Art bewirkt also, daß der Grauwert $g = s(x, y)$ des Originalbildes S (des *verzerrten* Bildes) im transformierten (*entzerrten*) Bild S' in der Position mit den Ortskoordinaten x' und y' erscheint. Je nach Art der Transformationsfunktionen ergeben sich verschiedene Arten der Modifikation der Ortskoordinaten. Man unterscheidet zwischen **projektiven** und **affinen Transformationen**, wobei die affinen Transformationen als Untergruppe (Sonderfall) der projektiven Transformationen aufgefaßt werden können. Bei den affinen Abbildungen bleiben Parallelität, Teilverhältnis und Flächenverhältnis erhalten. Bei der projektiven Abbildung gehen schließlich auch diese Invarianten verloren, insbesondere die Parallelität: Die Bilder paralleler Geraden schneiden sich in einem Fluchtpunkt.

Geht man davon aus, daß die Ausdehnung des Objektes *klein* gegenüber der Aufnahmeentfernung ist, läßt sich die projektive Abbildung zwischen Objekt- und Bildebene durch eine näherungsweise affine Abbildung beschreiben. Im Fall, daß Objekt- und Bildebene parallel zueinander sind, geht die projektive Abbildung vollständig in die affine über.

Bei der **allgemeinen affinen Transformation** der Ortskoordinaten sind die Funktionen lineare Polynome in x und y:

$$\begin{aligned} x' &= f_1(x, y) = a_0 + a_1 x + a_2 y \\ y' &= f_2(x, y) = b_0 + b_1 x + b_2 y. \end{aligned} \tag{2.15}$$

In Matrizenschreibweise lautet (2.15):

$$\vec{x}' = A\vec{x} + \vec{v} \tag{2.16}$$

oder

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = \begin{pmatrix} a_1 & a_2 \\ b_1 & b_2 \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} a_0 \\ b_0 \end{pmatrix} . \tag{2.17}$$

Durch entsprechende Wahl der Parameter a_i und b_i erhält man verschiedene Transformationen der Ortskoordinaten.

Bei der **Vergrößerung (Skalierung)** wird das Originalbild S mit L Bildzeilen und R Bildspalten in ein Bild S' überführt, das $L' > L$ und $R' > R$ Bildzeilen, bzw. Bildspalten besitzt. Ist der Vergrößerungsmaßstab in beiden Richtungen gleich, so haben die Funktionen der Transformation die einfach Form

$$f_1(x, y) = ax \text{ und } f_2(x, y) = ay\,. \tag{2.18}$$

Dabei ist $a > 1$ der Vergrößerungsfaktor. Durch Einsetzen der Beziehung (2.18) in (2.14) erhält man den Zusammenhang:

$$s'(ax, ay) = s(x, y)\,. \tag{2.19}$$

Diese Vorgehensweise bezeichnet man als **direkte Methode**. Ausgehend von den Koordinaten des verzerrten Bildes wird die Position des Grauwertes im entzerrten Bild berechnet.

Bei der **indirekten Methode** wird der umgekehrte Weg eingeschlagen. Zu jeder Position des entzerrten Bildes wird im verzerrten Bild der zugehörige Grauwert bestimmt. Hierzu werden allerdings die Umkehrfunktionen $f_1^{-1} = f_2^{-1} = 1/a$ benötigt, die auch überall definiert sind, da $a > 1$ vorausgesetzt wurde. Der Algorithmus lautet dann ([Hab91]):

$$\begin{aligned} &\text{FOR } x' := 0 \text{ TO } a * (L-1) \\ &\text{FOR } y' := 0 \text{ TO } a * (R-1) \\ &\text{BEGIN} \\ &x := \ \text{ROUND}(x'/a); \\ &y := \ \text{ROUND}(y'/a); \\ &s'(x', y') := s(x, y) \\ &\text{END.} \end{aligned} \tag{2.20}$$

Bei der Koordinatenberechnung mit (2.20) tritt der Fall ein, daß die berechneten Werte x'/a und y'/a nicht ganzzahlig sind und somit keine Bildpunktposition im verzerrten oder entzerrten Bild bezeichnen. Es ist dann die Frage, welche Position gewählt wird und wie der zugehörige Grauwert berechnet werden soll. Diese Fragestellungen werden mit dem Begriff **resampling** bezeichnet.

Die **Verkleinerung** kann mit der Formel (2.18) ausgedrückt werden, wenn man $0 < a < 1$ annimmt. Unterschiedliche Maßstabsfaktoren in Zeilen- und Spaltenrichtung können durch folgende Wahl der Funktionen erreicht werden:

$$
\begin{aligned}
f_1(x,y) &= ax \\
f_2(x,y) &= bx\,,
\end{aligned}
\tag{2.21}
$$

mit $a, b > 0$.

Bei einer **Verschiebung (Translation)** ist die Matrix A (vergleiche hierzu die Matrixschreibweise bei den affinen Abbildungen) die Einheitsmatrix. Der Verschiebungsvektor v gibt die Verschiebungsanteile in x- und y-Richtung an:

$$
\begin{aligned}
x' &= x + a_0\,, \\
y' &= y + b_0\,.
\end{aligned}
\tag{2.22}
$$

Unter einer Translation bleiben alle geometrischen Größen eines Objektes und die Orientierung erhalten.

Drehungen (Rotationen) können ebenfall durch (2.15) dargestellt werden. Die Matrix A hat bei einer Drehung um den Winkel α die Gestalt:

$$
A = \begin{pmatrix} \cos\alpha & \sin\alpha \\ -sin\alpha & \cos\alpha \end{pmatrix}. \tag{2.23}
$$

In diesem Abschnitt haben wir die affinen Transformationen **Skalierung (Vergrößerung, Verkleinerung), Translation** und **Rotation** vorgestellt. Es gilt Größen zu berechnen, die invariant gegenüber solchen in der Realität ausreichenden Transformationen sind. Einige Invarianten sind schon sehr lange bekannt und häufig in der Mathematik und in der Mustererkennung verwendet worden, wie z.B das Doppelverhältnis bei projektiven Abbildungen oder das Teilverhältnis bei affinen Abbildungen. Bei unserer Aufgabenstellung ist es aber wichtig, vollständige Sätze von Invarianten zu berechnen, damit die einzelnen Algen eindeutig durch ihre invarianten Merkmale beschrieben werden können. Seit Anfang der 60er Jahre beschäftigen sich Wissenschaftler mit diesem Thema in der Bildverarbeitung. Zu den wichtigsten Arbeiten zählen die von Burghardt, Arbter, Hu und Wang ([Bur79], [Arb90], [Hu62], [Wan77]). Die Tabelle 2.24 verdeutlicht die Zusammenhänge zwischen diesen Arbeiten, die unabhängig voneinander entstanden sind und trotzdem auf einem gemeinsamen Grundprinzip beruhen.

Transformation 2D–Bild	Translation	Skalierung	Rotation	affine Transfor–mation
geordnete Punktmenge			eindim. Fourier–deskriptoren [Bur79]	eindim. Fourier–deskriptoren [Arb90] projektive Invarianten
Grauwertbilder (diskrete 2D–Funktionen)	zweidim. Fourier–deskriptoren			
Grauwertbilder (stetige 2D–Funktion)		Legendre Invarianten	Hu–Moment–Invarianten [Hu62]	Wang–Invarianten [Wan77] Tensorinvarianten [Rei93]

Abbildung 2.24: *1– und 2–dimensionale Fourierdeskriptoren, Momente und Legendre Invarianten eingeteilt nach ihrer Invarianz bezüglich Rotation, Translation und Skalierung.*

2.2.2 Fourierdeskriptoren

Die Methode zur Beschreibung ebener Kurven durch Fourierdeskriptoren beruht auf der Theorie der Fourierreihen für periodische Funktionen. Die Fourierdeskriptoren verwenden wie der Kettencode nur die Objektränder, beschreiben eine Kurve jedoch nicht auf einem diskreten Gitter. Fourierdeskriptoren können für kontinuierliche oder abgetastete Kurven formuliert werden. Betrachten wir die in der Abbildung 2.25 gezeichnete geschlossene Kurve.

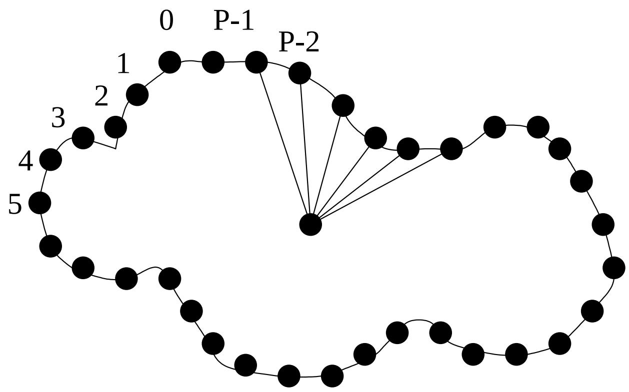


Abbildung 2.25: *Äquidistante Abtastung der Kurve mit P Punkten und Veranschaulichung der Parameterdarstellung einer geschlossenen Kurve. Der Parameter p ist die Pfadlänge vom Startpunkt (x_0, y_0) aus. Abtastrichtung ist entgegen dem Uhrzeigersinn.*

Wir können sie in Parameterdarstellung wiedergeben, indem wir die Pfadlänge p von einem Startpunkt (x_0, y_0) aus als Parameter verwenden.

Es ist nicht einfach, eine Kurve mit gleichmäßig voneinander entfernten Abtastpunkten zu erzeugen. Diskrete Randkurven wie der Kettencode haben in dieser Hinsicht Nachteile. Bei einer 8er–Nachbarschaft sind die Abtastpunkte nicht äquidistant, wohl aber bei einer 4er–Nachbarschaft (hierfür siehe Abbildung 2.22). Hier ist der Rand ausgefranst, weil die Teilstücke nur in horizontaler und vertikaler Richtung verlaufen können. Daher ist der Umfang eher zu groß. Es ist also nicht empfehlenswert, eine kontinuierliche Randkurve aus Punkten auf einem regelmäßigen Gitter zu bilden.

Eine kontinuierliche Randkurve hat die Komponenten $x(p)$ und $y(p)$. Die Konturlinie eines Objektes läßt sich also in der komplexen Ebene mit den komplexen Zahlen $k(p) = x(p) + iy(p)$ zu einer zyklischen Kurve kombinieren. Ist P der Umfang der Kurve, dann gilt:

$$k(p + nP) = k(p) \qquad (n \in \mathbb{N}). \tag{2.24}$$

Betrachtet man die Kontur eines Objektes als Funktion des Parameters in der komplexen Ebene gemäß folgender Funktion:

$$k(p) = x(p) + i\,y(p) \qquad \text{mit } 1 \leq p \leq P, \tag{2.25}$$

so kann diese, sofern es sich um eine geschlossene Kontur handelt (dies ist in unserem Fall durch die Vorverarbeitung, insbesondere durch die Konturextraktion gewährleistet), ausgehend von einem Startpunkt $p = 0$, ohne Absetzen unendlich oft durchlaufen werden. Bei einer offenen Kontur kann der nicht geschlossene Bereich durch eine fiktive Gerade geschlossen werden. Entlang dieser Gerade ändert sich der Parameter p nicht und somit weist die Kurve $k(p)$ keine Unstetigkeitsstelle auf. In beiden Fällen wird $k(p)$ zu einer periodischen Funktion mit der Periode P, die durch die Fourierreihe

$$\hat{k}(p) = \sum_{k=-\infty}^{\infty} C_k \, \exp\left(\frac{-2\pi ikp}{P}\right) \tag{2.26}$$

mit den Koeffizienten

$$C_k = \frac{1}{P} \int_0^P k(p) \, \exp\left(\frac{-2\pi ikp}{P}\right) dp \qquad (k = 0, \pm 1, \ldots, \pm\infty) \tag{2.27}$$

repräsentiert wird. Der Abbruch der Reihe in 2.26 nach endlich vielen Gliedern ergibt die Fourierapproximation der Kurve.

Aus den Fourierkoeffizienten kann wieder die periodische Kurve rekonstruiert werden:

$$k(p) = \sum_{k=-\infty}^{\infty} C_k \, \exp\left(\frac{2\pi ikp}{P}\right). \tag{2.28}$$

Die Koeffizienten C_k werden als **Fourierdeskriptoren** der Randkurve bezeichnet. Die geometrische Bedeutung der Deskriptoren läßt sich leicht erkennen. Der erste Koeffizient

$$C_0 = \frac{1}{p}\int_0^P k(p)\, dp = \frac{1}{p}\int_0^P x(p)\, dp + \frac{i}{p}\int_0^P y(p)\, dp \tag{2.29}$$

liefert den Mittelpunkt oder das *Zentroid* der Randkurve. Der zweite Koeffizient beschreibt einen Kreis:

$$\begin{aligned} k_1(p) &= C_1 \exp\left(\frac{2\pi ip}{P}\right) \\ &= r_1 \exp(\varphi_1 + 2\pi ip/P). \end{aligned} \tag{2.30}$$

Der Radius r_1 und der Startpunkt bei einem Winkel von φ_1 werden durch $C_1 = r_1 exp(i\varphi_1)$ bestimmt. Der Koeffizient C_{-1} ergibt ebenfalls einen Kreis, der jedoch in entgegengesetzter Richtung (mit dem Uhrzeiger) durchlaufen wird:

$$k_{-1}(p) = r_{-1} \exp(2\pi i(\varphi_1 - p/P)). \tag{2.31}$$

Mit beiden komplexen Koeffizienten zusammen (insgesamt also vier Parametern) kann eine Ellipse mit beliebigen Halbachsen a und b, beliebiger Orientierung ϑ und beliebigen Startwinkel φ_0 auf der Kontur gebildet werden. Als Beispiel betrachten wir $\varphi_1 = \varphi_{-1} = 0$. Dann ist

$$\begin{aligned} C_1(p) = C_{-1}(p) &= (r_1 + r_2) \cdot \cos\left(\frac{2\pi p}{P}\right) \\ &+ i(r_1 - r_2) \cdot \sin\left(\frac{2\pi p}{P}\right). \end{aligned} \tag{2.32}$$

Dies ist die Parameterdarstellung einer Ellipse, deren Halbachsen parallel zu den Achsen des Koordinatensystems ausgerichtet sind und deren Startpunkt auf der x–Achse liegt.

Aus der geometrischen Veranschaulichung der Deskriptoren sehen wir, daß Fourierdeskriptoren immer paarweise auftreten. Die Paarung von Koeffizienten höherer Ordnung ergibt ebenfalls eine Ellipse. Diese Ellipsen werden jedoch p–mal durchlaufen. Die Überlagerung mit der Basisellipse des ersten Paares bedeutet, daß die Fourierdeskriptoren höherer Ordnung immer mehr Details zur Randkurve beisteuern. Die Experimente im Kapitel 4 zeigen, daß nur wenige Koeffizienten zur Beschreibung recht komplizierter Formen notwendig sind.

Fourierdeskriptoren können auch aus den abgetasteten Konturen k_n berechnet werden. Ist der Umfang der geschlossenen Kurve P, so müssen N Abtastpunkte in gleichmäßigen Abständen P/N abgegriffen werden (2.25). Dann ist

$$C_k = \frac{1}{N} \sum_{n=0}^{N-1} k_n \exp\left(\frac{-2\pi i k n}{N}\right). \tag{2.33}$$

Einige Beispiele für die Abtastung der Kontur in äquidistanten Schritten, sind in Abbildung 2.26 zu sehen.

Eigenschaften der Fourierdeskriptoren

Symmetrien können in Fourierdeskriptoren leicht detektiert werden. Hat die Kontur eine *m–zahlige Rotationssymmetrie*, können nur die Deskriptoren mit den Indizes $k_{1\pm km}$ ungleich null sein.
Fourierdeskriptoren können auch für *nichtgeschlossene Kurven* verwendet werden. Um solche Kurven zu schließen, verfolgt man sie einfach vorwärts und rückwärts. Offene Kurven können leicht erkannt werden, da die Fläche null ist.

Die *Fläche* eines Objektes kann man direkt aus den Fourierdeskriptoren ermitteln:

$$A = \pi \sum_{k=-N/2}^{N/2-1} k\,|C_k|^2. \tag{2.34}$$

Aus 2.34 kann man schließen, daß $|C_{-k}| = |C_k|$. Beginnt der Durchlauf an einem der Endpunkte, so ist sogar $C_{-k} = C_k$.

Dies ist eine schnelle Methode der Flächenberechnung, die höchstens so viele Operationen benötigt, wie die Kontur Punkte enthält. Man kann also auf die Methode des Aufsummierens der Objektelemente verzichten. Je mehr Fourierdeskriptoren wir verwenden, umso detaillierter wird die Randkurve.

Bei der Berechnung des *Umfangs* mittels Fourierdeskriptoren gibt es leider keine einfache Formel, da die Berechnung von Ellipsen zu elliptischen Integralen führt. Allerdings ergibt sich eine gute Näherung des Umfangs aus der Konstruktion der Randkurve mit Abtastpunkten in gleichmäßigen Abständen durch die Anzahl der Abtastpunkte multipliziert mit dem mittleren Abstand zwischen den Punkten.

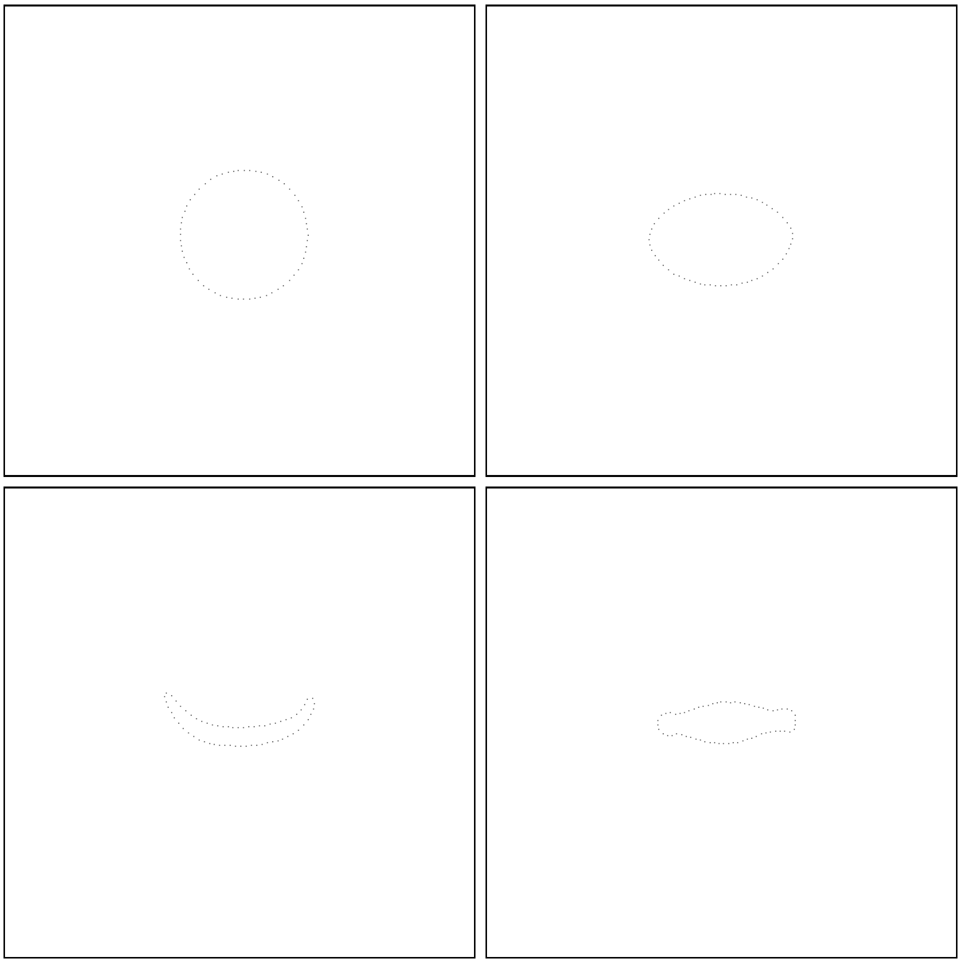


Abbildung 2.26: *Äquidistante Abtastung der Kontur verschiedener Algen aus unterschiedlichen Klassen als Vorstufe für die anschließende eindimensionale Fouriertransformation über die Distanzen zwischen dem Zentroid der Zelle und dem ausgewählter Konturpunkt. Es sind immer* 64 *Abtastpunkte gewählt, die durch die Fouriertransformation und unter Berücksichtigung der Symmetrie einen* 32*–dim.* **Fourierdeskriptoren–Vektor** *ergeben.*

Invariante Eigenschaften der Fourierdeskriptoren

- **Translationsinvarianz**
  Die Position des Objekts beeinflußt nur einen einzigen Koeffizienten, nämlich C_0. Alle anderen sind translationsinvariant. Diese Eigenschaft haben wir uns zunutze gemacht, den null–ten Fourierdeskriptor weggelassen und nur die anderen als Elemente eines Merkmalsvektors berücksichtigt.

- **Rotationssinvarianz**
  Wird eine Kontur um den Winkel φ_0 entgegen dem Uhrzeigersinn gedreht, wird der Fourierdeskriptor C_k entsprechend dem Verschiebungstheorem der Fouriertransformation mit dem Phasenfaktor $\exp(ik\varphi_0)$ multipliziert. Durch diese Verschiebungseigenschaft wird die Konstruktion rotationsinvarianter Fourierdeskriptoren einfach. Man kann beispielsweise die Phase aller Fourierdeskriptoren zur Phase φ_1 von C_1 in

Beziehung setzten und die Phasenverschiebung $k\varphi_1$ von den Phasen der anderen Deskriptoren extrahieren. Dadurch erhalten die Deskriptoren die Phasendifferenz zur Phase des ersten Deskriptors und sind damit, bis auf den ersten, rotationsinvariant.

- **Skalierungsinvarianz**
  Wird die Kontur mit einem Koeffizienten α skaliert, werden auch alle Fourierdeskriptoren mit α skaliert. Bei einem Objekt mit der Fläche ungleich null, dessen Kontur entgegen dem Uhrzeigersinn verfolgt wird, ist der erste Koeffizient immer ungleich null. Man kann also alle Fourierdeskriptoren auf $|C_1|$ beziehen, um skalierungsinvariante Deskriptoren für das Objekt zu erhalten. Man beachte, daß die skalierten Deskriptoren immer noch die vollständige Information enthalten.

Da wir uns im Kapitel 4 um einen Vergleich von Merkmalsvektoren, bezüglich ihrer Eignung für neuronale Klassifikationsverfahren bemühen werden, ist es sinnvoll weitere Merkmale einzuführen, die ebenso die oben benannten Invarianzen vorweisen. Die Momente nach [Hu62] erfüllen diese Kriterien und werden im kommenden Abschnitt vorgestellt.

2.2.3 Momente

Sei $f : N \times N \longrightarrow \{0,1\}$ eine zweidimensionale, binäre Funktion, die die binäre Darstellung eines Objektes in der Bildebene repräsentiert. Sei mit $S = \{(x,y)|f(x,y) = 1\}$ die Menge der gesetzten Punkte des Objekts angegeben. In unserem Fall enthält S die Pixel der extrahierten Kontur. Für jedes Paar natürlicher Zahlen (j,k) ist das digitale (j,k)-te Moment von S definiert durch

$$M_{jk}(S) = \sum_{(x,y)\in S} x^j y^k. \tag{2.35}$$

$M_{00}(S)$ gibt dann die Anzahl der Bildpunkte von S an. Die invarianten Momente sind Funktionen der digitalen (j,k)-ten Momente, die unter bestimmten Transformationen unverändert bleiben. Um eine zwei-dimensionale Konturerkennung durchführen zu können, benötigt man Momente, die unter Translation, Rotation, Skalierung oder ggf. Scherung invariant bleiben. Der Schwerpunkt $(\overline{x}, \overline{y})$ eines Objektes läßt sich ebenfalls mit Hilfe der Momente definieren:

$$\begin{aligned} \bar{x} &= \frac{M_{10}(S)}{M_{00}(S)} \\ \bar{y} &= \frac{M_{01}(S)}{M_{00}(S)}. \end{aligned} \tag{2.36}$$

Mit dem Schwerpunkt läßt sich das zentrale (j,k)-te Moment von S definieren durch

$$\mu_{jk}(S) = \sum_{(x,y)\in S} (x-\bar{x})^j\,(y-\bar{y})^k. \qquad (2.37)$$

μ_{00} ist die Summe der Grauwertverteilung $f(x,y)$ über dem Eingabebereich der xy–Ebene. μ_{10} bzw. μ_{01} sind die statischen Momente und μ_{20} und μ_{02} sind die Trägheitsmomente, jeweils bezüglich der Parallelen zur y– bzw. x–Achse durch den Schwerpunkt $(\bar{x},\bar{y})$.

Behauptung: Die zentralen Momente sind invariant unter Translation.
Beweis: siehe Anhang.

Die Standardabweichungen in x– bzw. y–Richtung können auch als Funktionen von Momenten aufgeschreiben werden:

$$\begin{aligned} \sigma_x &= \sqrt{\frac{\mu_{20}}{M_{00}}} \\ \sigma_y &= \sqrt{\frac{\mu_{02}}{M_{00}}}. \end{aligned} \qquad (2.38)$$

Damit können die Koordinaten bezüglich ihrer Standardabweichungen normalisiert werden:

$$\begin{aligned} x' &= \frac{(x-\bar{x})}{\sigma_x} \\ y' &= \frac{(y-\bar{y})}{\sigma_y}. \end{aligned} \qquad (2.39)$$

Dies bedeutet, daß die Mittelwerte von x' und y' gleich null und die Varianzen gleich eins sind. Die Normalisierung bezüglich der Fläche liefert normalisierte zentrale Momente:

$$m_{jk} = \frac{\sum (x')^j\,(y')^k}{M_{00}}. \qquad (2.40)$$

Diese Momente sind invariant unter Translation und Skalierung, allgemein unter affinen Transformationen der Form $x^* = ax + b$, $y^* = cy + d$.

Behauptung: Normalisierte Momente sind invariant unter Translation.
Beweis: siehe Anhang.

Sie sind jedoch nicht rotationsinvariant. Sogenannte normalisierte zentrale Momente η_{jk} von S ergeben sich entweder direkt aus

$$\eta_{jk} = \mu_{jk}, j + k = 2, 3, ... \tag{2.41}$$

oder aus

$$\eta_{jk} = \frac{\mu_{jk}}{\mu_{00}^{\gamma}}, \quad \gamma = \frac{j+k}{2} + 1, j + k = 2, 3, ... \tag{2.42}$$

Mit diesen normalisierten Momenten hat Hu [Hu62] sieben Funktionen definiert, die rotationsinvariant sind:

$$\begin{aligned}
\phi(1) &= \eta_{20} + \eta_{02} \\
\phi(2) &= (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2 \\
\phi(3) &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\
\phi(4) &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\
\phi(5) &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] \\
&+ (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} - \eta_{03})^2] \\
\phi(6) &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} - \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\
\phi(7) &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} - \eta_{03})^2] \\
&+ (3\eta_{12} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} - \eta_{03})^2].
\end{aligned} \tag{2.43}$$

Die letzte Funktion $\Phi(7)$ ist sogar scherungsinvariant. Manchmal ist es nützlich die Funktionen $log|\Phi(i)|$ zu verwenden. Wenn das Vorzeichen von $\Phi(i)$ wichtig ist, können die Funktionen $Sgn[\Phi(i)]log||\Phi(i)|$ verwendet werden.

Zusammenfassung Fourierdeskriptoren und Momente

Sowohl *Fourierdeskriptoren* (Abschnitt 2.2.2) als auch *Momente* (Abschnitt 2.2.3) liefern die Grundlage für skalierungs– und rotationsinvariante Formparameter. Die *Fourierdeskriptoren* sind die vielseitigeren Merkmale. Allerdings beschränken sie die Objektbeschreibung auf die Kontur, während *Momente* von Grauwertobjekten (von uns nur die *Momente* auf Binärobjekte behandelt) auch die räumliche Verteilung der Grauwerte in einem Objekt berücksichtigen. Die Schwierigkeit besteht darin, eine vollständige und eindeutige Beschreibung zu finden; d.h., daß unterschiedliche Formen nicht auf denselben Merkmalssatz abgebildet werden dürfen. Der Betrag der *Fourierdeskriptoren* liefert eine skalierungs– und rotationsinvariante, jedoch unvollständige Formbeschreibung. Nur der komplette Satz der *Fourierdeskriptoren* gewährleistet eine eindeutige Formbeschreibung.

Es ist ebenso zu erwähnen, daß für jede Invarianz die man gewinnt ein Freiheitsgrad verlorengeht. Bei der Tranlationsinvarianz läßt man den *Fourierdeskriptor* C_0 aus (zwei Freiheitsgrade). Dei der Skalierungsinvarianz setzt man den Betrag des *Fourierdeskriptors* C_1 auf eins (ein Freiheitsgrad), und für die Rotationsinvarianz setzt man alle Phasen zur Phase von C_1 in Beziehung (ein weiterer Freiheitsgrad). Es ist der Beschaffenheit der *Fourierdeskriptoren* zuzuschreiben, daß diese Invarianzen in den ersten beiden *Fourierdeskriptoren* enthalten sind. Normiert man alle anderen *Fourierdeskriptoren* auf die Phase und den Betrag des zweiten *Fourierdeskriptors*, erhält man eine vollständige translations–, rotations– und skalierungsinvariante Beschreibung der Form eines Objekts. Läßt man *Fourierdeskriptoren* höherer Ordnung weg, so kann man kontrolliert feine Details aus der Formbeschreibung entfernen, läßt man sie zu, so erhält man die detaillierte Konturbeschreibung. Ein Maß für Formunterschiede ergibt sich, wenn man berücksichtigt, daß *Fourierdeskriptoren* einen komplexwertigen Vektor bilden. Dann gilt für den Betrag des Differenzvektors:

$$d_{kk'} = \sum_{k=0}^{N} |C_k - C'_k|. \tag{2.44}$$

2.2.4 Weitere Merkmale

Nach den *Fourierdeskriptoren* und den sieben rotationsinvarianten Funktionen basierend auf *Momenten* beschäftigt sich dieser Abschnitt einerseits mit elementaren geometrischen Parametern wie *Fläche*, *Umfang* oder *Rundheit*, andererseits aber auch mit *Momenten zweiter Ordnung* und mit Merkmalen, wie z.B *Exzentrizität*, *Orientierung* u.a., die sich aus den Momenten ableiten lassen.

Fläche

Die *Fläche* A ist einer der simpelsten Formparameter eines Objekts. Sie ist bei einem diskreten Binärbild durch die Anzahl der Bildpunkte gegeben, die zum Objekt gehören. Bei der Matrixdarstellung eines Objekts bedeutet daher die Flächenberechnung einfach eine Zählung der Pixel. Da wir die Konturen der Objekte durch den *Kettencode* beschrieben haben, machen wir uns diese Kodierung der Kontur zunutze und verzichten auf das Aufsummieren der Pixel. Wir stellen fest, daß der Algorithmus der Flächenberechnung mit dem Kettencode viel schneller ist als das Aufaddieren der Objektpixel, da der Objektrand nur einen kleinen Teil der Pixel eines Objekts enthält und nur zwei Additionen pro Konturpixel erfordert. Zu Beginn werden die Flächen auf null gesetzt. Mit jedem Schritt werden die Fläche und der Umfang P entsprechend dem Wert des Kettencodes erhöht (nach [Zam89]).

Umfang

Der Umfang eines Objekts kann ebenfalls aus dem Kettencode berechnet werden. Man

Kettencode	Flächenzunahme	Zunahme von P
0	$+P$	0
1	$+P$	1
2	0	1
3	$-P$	1
4	$-P$	0
5	$-P$	-1
6	0	-1
7	$+P$	-1

Abbildung 2.27: *Berechnung der Fläche A aus dem Kettencode.*

muß dafür die Länge des Kettencodes zählen und berücksichtigen, daß diagonale Schritte um den Faktor $\sqrt{2}$ länger sind. Der Umfang P errechnet sich bei bei einem Kettencode, unter Berücksichtigung der 8er–Nachbarschaft nach folgender Formel:

$$P = n_g + \sqrt{2}n_u, \tag{2.45}$$

wobei n_g und n_u jeweils die Anzahl der Kettencodeschritte mit geradem bzw. ungeradem Code (sie gehen in Diagonalrichtung) sind. Der Umfang ist ein Parameter, der viel stärker vom Rauschen beeinflußt wird als z.B. die Fläche. Wenn man einen Objektvergleich aufgrund der extrahierten Konturen anstrebt, sollte man sich vergewissern, daß die Glattheit der Ränder in allen Bildern vergleichbar ist.

Rundheit

Um Objekte vergleichen zu können, die aus verschiedenen Entfernungen aufgenommen wurden, braucht man Formparameter, die von der Größe des Objekts in der Bildebene unabhängig sind. Insbesondere bei den Bildmustern aus unserer Datenbank, die sowohl handgezeichnete als auch elektronenmikroskopische Aufnahmen beinhaltet, ist es sinnvoll, außer der *Fläche* und dem *Umfang*, welche die Größe eines Objektes beschreiben, auch Parameter zu berücksichtigen, die nicht von der Orientierung des Objekts in der Bildebene abhängen. Die *Rundheit* ist ein solches Merkmal. Sie ist wie folgt definiert:

$$R = \frac{P^2}{A}. \tag{2.46}$$

Die *Rundheit* ist eine dimensionslose Zahl mit einem Minimalwert von $4\pi \approx 12,57$ für Kreise und 16 für Quadrate.

Fläche, *Umfang* und *Rundheit* eignen sich per Definition zu einer von der Orientierung unabhängigen Unterscheidung von Objekten.

Umgebendes Rechteck

Auf englisch: *bounding box* ist ein weiterer nützlicher Parameter für die grobe Größenapproximation eines Objekts. Es ist definiert als das Rechteck, was gerade groß genug ist, um alle Objektpunkte aufzunehmen. Im Gegensatz zur *Fläche* ist dieses Merkmal nicht rotationsinvariant.

Orientierung

Die *Orientierung* θ eines Objekts ist definiert als der Winkel zwischen der x–Achse und der Achse, um die das Objekt mit minimaler Trägheit gedreht werden kann. Dies ist der Eigenvektor zum minimalen Eigenwert. Wenn man die Momente zweiter Ordnung aus dem Abschnitt 2.2.3 betrachtet, dann ergibt sich fogender Orientierungswinkel:

$$\theta = \frac{1}{2} \arctan \frac{2M_{11}}{M_{20} - M_{02}}. \tag{2.47}$$

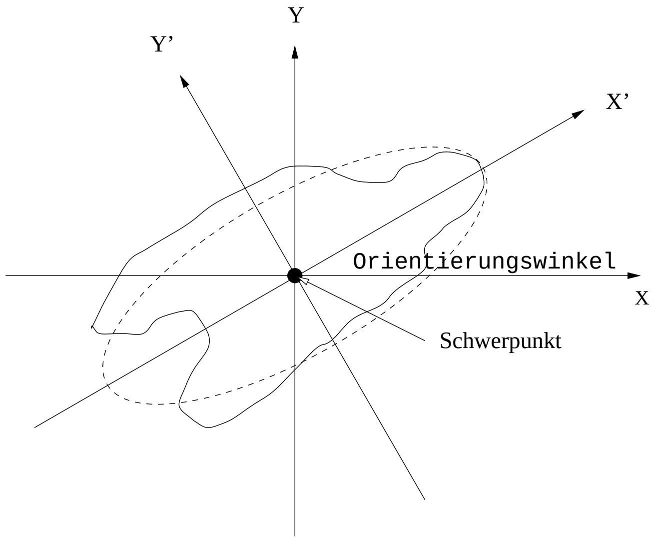


Abbildung 2.28: *Orientierungswinkel eines Objekts.*

Exzentrizität

Als Maß für die *Exzentrizität* ϵ können wir die Größe verwenden, die wir bei der lokalen Orientierung als Kohärenzmaß definiert haben (beim Trägheitstensor wird die Richtung der lokalen Orientierung durch den minimalen Eigenwert gegeben, für den Strukturtensor jedoch durch den maximalen Eigenwert):

$$\epsilon = \frac{(M_{20} - M_{02})^2 + 4M_{11}^2}{(M_{20} + M_{02})^2}. \tag{2.48}$$

Die Exzentrizität nimmt Werte von null bis eins an.

Die Abbildung 2.29 verdeutlicht die Varianz der oben aufgeführten Merkmale. Die Beispiele sind so gewählt, daß man sich aufgrund der unterschiedlichen morphologischen Gegebenheiten, ein Bild über die Spannbreite der Kombinationsmöglichkeiten, was die Gestaltung von Merkmalsvektoren betrifft, machen kann. Man hat also die Wahl, sowohl auf Momenten basierte Merkmale zurückzugreifen, die eine vollständige Formbeschreibung erlauben ([Rei93]), als auch die elementaren Formparameter, wie *Fläche* oder *Umfang* zu nutzen, und somit eine Informationskomprimierung in Form eines Merkmalsvektors zu ermöglichen. Wie man geschickt die Merkmale miteinander und untereinander kombiniert, welche unentbehrlich sind und auf welche man lieber verzichten sollte, werden wir im kommenden Abschnitt näher erläutern.

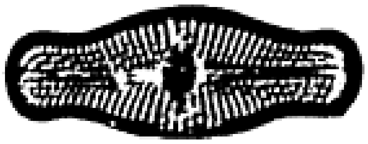
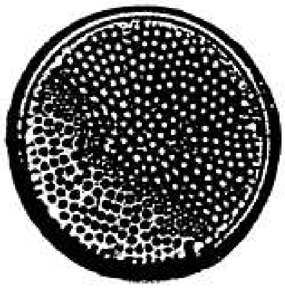
Merkmal		
xs	249	252
xy	256	252
Area	5388	18224
$M_{11}(fill)$	-63882	483072
$M_{20}(fill)$	801095	26761259
$M_{02}(fill)$	6944746	26172706
θ	-1.560400	0.511828
l	322	155
w	51	151
ratio	3.047884	1.284295
a–ellipse	24.253950	75.669363
b–ellipse	71.430956	76.115312
ϵ	0.339544	0.994141
$M_{11}(Kontur)$	-5501	16758
$M_{20}(Kontur)$	593572	1224637
$M_{02}(Kontur)$	107576	1251569

Abbildung 2.29: *Eine Vergleichstabelle für die Merkmale zweier verschiedener Algengattungen der 3. und der 1. Klasse (links bzw. rechts). Die Tabelle von oben nach unten: x–und y–Koordinaten des Schwerpunktes, Fläche, Momente zweiter Ordnung (auf der gefüllten Kontur berechnet), Orientierung θ im Bogenmaß, Länge und Breite des umfaßenden Rechtecks, ratio ($l \cdot w/Area$), Haupt– und Nebenachse der* Best–Fit–Ellipse *, Exzentrizität (ϵ) und die Momente zweiter Ordnung (auf der Kontur berechnet).*

2.2.5 Merkmalsvektoren

Aus den in den vorrigen Abschnitten beschriebenen Merkmalen werden die folgenden Merkmalsvektoren zusammengestellt. Dabei wird einerseits zwischen *reinen* und *gemischten* Vektoren unterschieden, andererseits zwischen solchen die invariante Eigenschaften (bzgl. Rotation, Translation und Skalierung) haben und solchen, die diese Invarianzen nicht aufweisen. Bei den *reinen* Vektoren handelt es sich **nur** um Fourierdeskriptoren bzw. **nur** um Momente. Bei den *varianten* Merkmalsvektoren wurde ebenso auf eine sinnvolle Einteilung in analoge Kategorien geachtet. So sind z.B. die Merkmale die auf den Momenten zweiter Ordnung basieren (berechnet auf dem Konturbild und auf dem gefüllten Bild) mit den Merkmalen *Exzentrizität* und *Orientierung* zusammengefaßt worden, da sich beide (siehe Definitionen 2.48 bzw. 2.47) auf diese Momente stützen. Die weiteren Merkmale, die sich auf *Umfang*, *Rundheit* etc. beziehen, wurden ebenso in einem Vektor zusammengefaßt. Auf diese Weise entstanden **13** verschiedene **Merkmalsvektoren**, die sich nach Inhalt und Dimension unterscheiden. Für eine übersichtlichere Darstellung der Merkmalsvektoren haben wir die Nomenklatur wie folgt abgekürzt:

1. **Die** *reinen* **Merkmalsvektoren:**
   - **4FD** für die ersten 4 *Fourierdeskriptoren* (**4–dim.**)
   - **8FD** für die ersten 8 *Fourierdeskriptoren* (**8–dim.**)
   - **16FD** für die ersten 16 *Fourierdeskriptoren* (**16–dim.**)
   - **32FD** für die ersten 32 *Fourierdeskriptoren* (**32–dim.**)
   - **MOM7f** für 7 invariante Momente auf dem gefüllten Bild ($f = fill$) (**8–dim.**)
   - **MOM7k** für 7 invariante Momente auf dem Konturbild ($k = Kontur$) (**8–dim.**)
   - **MET** setzt sich zusammen aus: M_{11}(f), M_{02}(f), M_{20}(f), M_{11}(k), M_{02}(k), M_{20}(k), ϵ (Exzentrizität) und θ Orientierung (**8–dim.**)
   - **VAR** setzt sich zusammen aus: **A** (Fläche), **P** (Perimeter), **R** (Rundheit), **l** (Länge des umfaßenden Rechtecks), **w** (Breite des umfaßenden Rechtecks), **ratio**, $\boldsymbol{a_{ellipse}}$ (Längsachse der *best–fit–ellipse*), $\boldsymbol{b_{ellipse}}$ (Querachse der *best–fit–ellipse*) (**8–dim.**)
2. **Die** *gemischten* **Merkmalsvektoren:**
   - **8FD–MET** setzen sich zusammen aus den Vektoren **8FD** und **MET** (**16–dim.**)
   - **8FD–VAR** setzen sich zusammen aus den Vektoren **8FD** und **VAR** (**16–dim.**)
   - **MET–VAR** setzen sich zusammen aus den Vektoren **MET** und **VAR** (**16–dim.**)
   - **8FD–MOM7f** setzen sich zusammen aus den Vektoren **8FD** und **MOM7f** (**16–dim.**)

- **8FD–MOM7f–MET–VAR** setzen sich zusammen aus den Vektoren **8FD** und **MOM7f** und **MET** und **VAR** (**32–dim.**)

Diese 13 Merkmalsvektoren werden in verschiedenen Versuchsreihen (siehe Kapitel 4) auf ihre Brauchbarkeit für unsere 8–Klassen–Klassifikationsaufgabe hin untersucht und im Kapitel 5 werden abschließend die Ergebnisse methodenkritisch diskutiert und hinsichtlich ihrer Anwendungseignung bewertet.

Kapitel 3

Klassifikation mittels Neuronaler Netze

3.1 Einführung

Die Aufgabe einer Klassifikation oder Mustererkennung besteht darin, Objekte in Klassen einzuteilen ([Abm94], S.277). Unter einer Klassifizierung wird in unserem Fall also eine Abbildung verstanden, die jeder Alge genau eine Klasse zuweist.

Merkmale und Merkmalsraum

Die Zuordnung erfolgt auf der Basis eines Satzes von n Merkmalen, deren jeweilige Ausprägungen für alle Bildelemente vorliegen. Ein Merkmal kann dabei verschiedenen Quellen entstammen (siehe z.B. Fourierdeskriptoren 2.2.2, Momente 2.2.3, weitere Merkmale 2.2.4).

Hat man n für die Klassifizierung als relevant erachtete Merkmale ausgewählt, so läßt sich aus ihnen ein Koordinatensystem mit paarweise senkrecht aufeinanderstehenden Achsen bilden. Diese spannen einen n-dimensionalen Raum, den sogenannten Merkmalsraum auf. Trägt man darin die jeweiligen n Werte eines Bildpunkts ein, so stellt jeder Bildpunkt einen n-dimensionalen Merkmalsvektor in diesem n-dimensionalen Raum dar.

Jedes Bildelement eines Datensatzes bekommt über seine Merkmalswerte eine eindeutige Position im Merkmalsraum zugewiesen. Bildelemente mit ähnlichen Merkmalseigenschaften bilden eine Objekt- oder Musterklasse.

Die Pixel einer Klasse fallen dabei nicht in einem Punkt zusammen, sondern aufgrund individueller Unterschiede der Flächenelemente in einem Punkthaufen oder Cluster. Unterscheiden sich Objektklassen bezüglich der gemessenen Merkmale, so liegen die einzelnen

Objektklassen in verschiedenen Bereichen des Merkmalsraumes ([Alb91], S. 140). Abbildung 3.1 (links) zeigt eine hypothetische Häufigkeitsverteilung der Merkmale von drei Objektklassen in einem durch die Merkmale **FD**, **MOM** und **VAR** (siehe 2.2.5) aufgespannten dreidimensionalen Merkmalsraum.

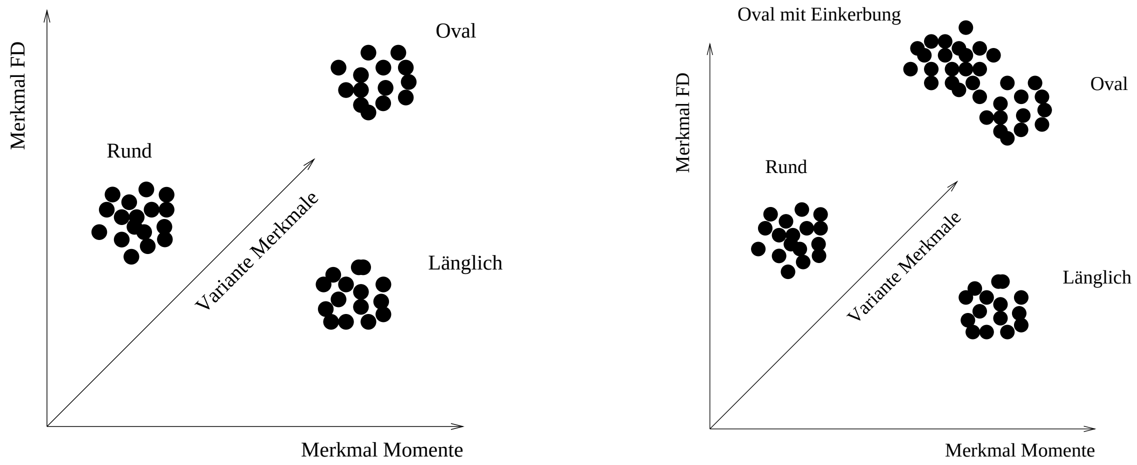


Abbildung 3.1: *Drei räumlich getrennte Objektklassen im selben dreidimensionalen Merkmalsraum (links) und vier räumlich z.T. nicht getrennte Objektklassen in einem dreidimensionalen Merkmalsraum (rechts).*

Die Cluster der einzelnen Objektklassen befinden sich hier signifikant voneinander getrennt in unterschiedlichen Bereichen des Merkmalsraumes. In der Praxis tritt eine derart deutliche Trennbarkeit der Objektklassen bezüglich der einbezogenen Merkmale allerdings nur selten auf. Meist liegen die Punktehaufen aufgrund nur beschränkt zur Verfügung stehender bzw. ungünstig gewählter Merkmale eng beinander oder sie überschneiden sich sogar (Abbildung. 3.1, rechts). Wird dann nicht mindestens ein weiteres zur Differenzierung der Objektklassen beitragendes Merkmal einbezogen, ist eine fehlerfreie Klassifizierung nicht möglich.

Grundsätzlich ist zunächst unbekannt, welche Punktewolke einer bestimmten Objektklasse zugeordnet ist und welche Bereiche des n-dimensionalen Merkmalsraumes die Objekte einnehmen ([Hab91], S. 251). Das Ziel der Klassifizierung besteht eher darin, Cluster aus ähnlichen Objekten im Merkmalsraum zu identifizieren und von benachbarten Clustern zu trennen ([Abm94], S. 277).

Für die Lösung dieser Aufgabe existieren zwei unterschiedliche Strategien, welche zu einer Einteilung der Klassifikationsverfahren in überwachte und unüberwachte Methoden führen.

Überwachte und unüberwachte Klassifikationsverfahren

Überwachten Verfahren liegt die Idee zugrunde, Vorinformationen über die Zugehörigkeit einzelner Objekte zu bestimmten Klassen in den Auswertungsprozeß einzubeziehen. Dazu wird die Häufigkeitsverteilung einer Objektklasse zunächst durch eine Stichprobe mit bekannten Objekten geschätzt ([Jäh97], S. 172). Hierfür wählt man Objekte aus, von denen man weiß, welcher Objektklasse sie angehören. Für jede zu unterscheidende Objektklasse muß mindestens ein derartiges Referenz- oder *Trainingsobjekt* vorliegen. Mit Hilfe solcher Referenzobjekte lassen sich dann numerische Kriterien ableiten, anhand derer die jeweilige Objektklasse charakterisiert und von anderen Klassen unterschieden werden kann. Es entsteht also eine Art *Interpretationsschlüssel* ([LK94], S. 586), über den jedes Objekt des Datensatzes der Klasse zugewiesen werden kann, zu der es die größte Ähnlichkeit hat.

Im Falle der Klassifikation von subantarktischen Algen stellen die Referenzobjekte Algen dar, deren Klassenzugehörigkeit bereits bekannt ist. Überwachte Verfahren setzen also ein *a priori*-Wissen über die zu klassifizierende Mikroorganismen voraus. Solche Vorinformationen können beispielsweise aus der Zusammenarbeit mit den Polarökologen oder aus den angegebenen literarischen Quellen stammen.

In einem ersten Schritt werden anhand der repräsentativ ausgewählten Muster numerische Daten über die Eigenschaften einzelner Klassen gesammelt. Für jede Klasse wird auf diese Weise eine für sie typische numerische Beschreibung, ein sog. Merkmalsvektor erzeugt. Diese Merkmalsvektoren können die im Kapitel 2 vorgestellten Fourierdeskriptoren enthalten oder die Momente oder weitere Merkmale, jedes für sich getrennt betrachtet aber auch als Kombination. Ein Merkmalsvektor kann also alle Merkmale umfassen, die zu einer Differenzierung der Objektklassen beitragen.

Nach der Erzeugung der Merkmalsvektoren sowohl für die Trainings- als auch für die Testmenge, erfolgt in einem zweiten Schritt ein Vergleich der Merkmalsvektoren der noch zu testenden Muster mit den Signaturen der Trainingsmenge. Ein noch nicht klassifiziertes Bild bekommt dann jene Klasse zugewiesen, zu der es (nach geeigneten Kriterien definiert), die größte Ähnlichkeit hat.

Unüberwachte Verfahren oder Cluster-Analysen beschreiten einen anderen Weg. Hier besteht die Aufgabe, die Gesamtheit der Bildelemente in eine Anzahl von Klassen zu unterteilen. Und zwar so, daß alle Bildelemente, die sich bezüglich der eingehenden Merkmale ähnlich sind, zu einer Klasse zusammengefaßt werden. Unüberwachte Verfahren verzichten auf ein Vorwissen über den Untersuchungsraum. Da somit auch keine Stichproben von bereits bekannten Objektklassen vorliegen, ist die Definition der Klassen lediglich von den Daten selber abhängig.

Cluster-Analysen haben zum Ziel, inhärente statistische Muster des Datensatzes aufzudecken. Dabei ist die Häufigkeitsverteilung jeder zu erzeugenden Klasse im Merkmalsraum unbekannt. Auch über die Anzahl der Klassen brauchen bei diesen Verfahren in der Regel keine Vorinformation vorliegen.

Unüberwachte Verfahren sind im Sinne der Klassenerzeugung objektiver als überwachte Verfahren ([Abm94], S. 172). Wie diese beinhalten sie jedoch für ihre sinnvolle Anwendung neben der rechnergestützten Klassenerzeugung auch einen vom Bearbeiter vorzunehmenden Interpretationsschritt. Denn mag es zwar in einigen Anwendungsfällen wichtiger sein, lediglich Gruppen von Bildern mit ähnlichen Merkmalsvektoren zu bilden, so ist man meist daran interessiert, den vorhandenen Mustern eindeutige Klassen zuzuordnen.

In eine überwachte Klassifikation fließt das Vorwissen des Bearbeiters dahingehend ein, daß dem eigentlichen Kategorisierungsvorgang eine Trainingsphase anhand bereits bekannter Musterklassen vorangestellt wird. Im Falle der unüberwachten Klassifikation dagegen ist das Vorgehen quasi umgekehrt. Hier erfolgt eine Integration der Kenntnisse des Bearbeiters nach der Zusammenfassung der Objekte, durch den Vergleich der klassifizierten Daten mit irgendeiner Form von Referenzdaten. Hierfür können – analog der Auswahl von Trainingsbeispielen bei der überwachten Klassifikation – Algenbeispiele aus Lehrbüchern, Elektronenmikroskopaufnahmen oder Handzeichnungen dienen.

Die Interpretation der unüberwacht erzeugten Klassen kann sich allerdings schwierig gestalten, da die generierten Muster lediglich Cluster von Pixeln mit ähnlichen Merkmalseigenschaften darstellen und nicht notwendigerweise mit erkennbaren realen Algen-Klassen korrespondieren müssen. Häufig werden Cluster-Analysen deshalb nicht als selbstständige Verfahren, sondern zur Vorbereitung einer überwachten Klassifizierung eingesetzt. Beispielsweise läßt sich mit ihnen überprüfen, ob die vorliegenden Meßdaten überhaupt eine Trennung in die gewünschten Objektklassen ermöglichen bzw. ob die gewählten Klassen sich nicht ihrerseits in mehrere Unterklassen zerlegen lassen ([Alb91], S. 143).

Neuronale Klassifikatoren

Neuronale Klassifikatoren basieren auf dem Konzept der künstlichen Neuronalen Netze, welches gerade in den letzten zehn Jahren in einer Vielzahl von Anwendungen Verbreitung gefunden hat (vgl. etwa die Beispiele in [Zel94], S. 493 ff).

Künstliche Neuronale Netze sind informationsverarbeitende Systeme, die eine abstrakte Simulation eines realen Nervensystems darstellen ([Kun93], S. 1). Wenn auch gegenüber biologischen neuronalen Systemen die Simulation der Neuronen (Nervenzellen) in künstlichen Neuronalen Netzen sehr stark idealisiert ist, so lassen sich denoch gewisse Ähnlichkeiten zum biologischen Vorbild erkennen. So bestehen künstliche Neuronale Netze aus einer Vielzahl von einfachen Prozesselementen, die mittels gerichteter und gewichteter Verbindungsstränge miteinander verknüpft sind und sich über diese Verbindungen Informationen in Form der Aktivierung der Zellen zusenden ([Zel94], S. 23). Die Stärke der Verbindungen bzw. die Gewichte dieser Verbindungen repräsentieren das Wissen im System, welches über die Interaktion zwischen den einzelnen Neuronen verarbeitet wird ([Kul94], S. 4). Wie diese Wissenspeicherung und –verarbeitung in einem Neuronalen Netz erfolgt, wird durch seine Verbindungsstruktur, die Verbindungsgewichte sowie die Arbeitsweise der einzelnen Prozesselemente bestimmt ([HAH$^+$94]). Für ausführliche Darstellungen vgl. z.B. [Roj93], [Was89], [Hay94], [Zel94]. Im folgenden werden zwei Arten von Neuronalen Netzen vorgestellt, das *Geometrische Netz* (basierend auf Clifford-Algebren) und das von Jörg Bruske am Lehrstuhl für *Kognitive Syteme* entwickelte *DCS-Netz* (**D**ynanic **C**ell **S**tructure). Das Geometrische Netz wurde in der Diplomarbeit von Sven Buchholz ([Buc97]) für Funktionsapproximation eingeführt und ist eine Erweiterung des MLP für Clifford–Algebren. Die DCS–Netze wurden im Rahmen zweier Diplomarbeiten für die Gesichtserkennung ([AM98]) und das visuell unterstützte Greifen ([Trö97]) bereits erfolgreich praktisch erprobt. Allerdings wurden diese beiden Netzarchitekturen noch nicht auf eine Klassifikationsaufgabe angewendet. Im folgenden Abschnitt stellen wir beide Netze theoretisch näher vor.

3.2 Motivation

Unter den vielen interessanten Eigenschaften Neuronaler Netze kommt insbesondere der Lernfähigkeit eine herausragende Bedeutung zu. Es ist dies die Fähigkeit eines Neuronalen Netzes, selbstständig aus seiner Umgebung anhand von Trainingsbeispielen zu lernen und seine Leistungsfähigkeit mit zunehmender Lerndauer zu verbessern, ohne daß das Netz dazu explizit programmiert werden muß ([Hay94], S. 45; [Zel94], S. 23).

Ein Neuronales Netz lernt aus seiner Umgebung über einen iterativen Prozeß. Hierzu wird dem Netz eine Menge von Trainingsdatensätzen an der Eingabeschicht präsentiert, d.h. jedes Eingabeneuron erhält einen Wert des Trainingsdatensatzes zugewiesen. Typischerweise ist für jeden dieser Trainingsdatensätze bereits vorab bekannt (überwachtes Lernen), welche Netzausgabe er produzieren soll, d.h. welcher der N Klassen er zugeordnet werden soll.

Das Lernziel besteht darin, die vorgegebene Zuordnung möglichst genau zu reproduzieren.

Beim Lernen liefert das Netz für jede Trainingseingabe eine zugehörige Ausgabe, welche mit der gewünschten Ausgabe verglichen wird. Der Fehler zwischen aktueller und gewünschter Ausgabe dient dazu, die Gewichte der Verbindungen im Netz zu modifizieren. Die Netzparameter werden nur solange angepaßt, bis der Fehler einen vorher festgelegten Grenzwert unterschreitet bzw. sich nicht mehr weiter reduzieren läßt.

Graphisch läßt sich dies wie folgt veranschaulichen. Trägt man den Fehler eines Neuronalen Netzes als Funktion der Gewichte in ein Koordinatensystem ein, so erhält man eine Fehlerfläche, die sich im zweidimensionalen Fall anschaulich darstellen läßt (Abbildung 3.2 zeigt dir Fehlerfläche aus zwei verschiedenen Perspektiven).

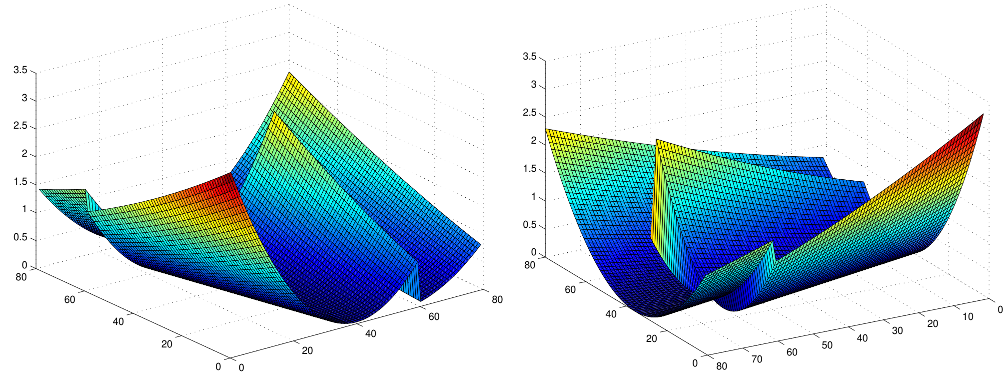


Abbildung 3.2: *Fehlerfläche eines Neuronalen Netzes als Funktion der Gewichte w_1 und w_2 (links) und um -120 Grad gedreht (rechts).*

Die Fehlerfunktion gibt den Fehler an, den das Netz, bei gegebenen Gewichten w_1 und w_2 sowie über alle Trainingsmuster aufsummiert, besitzt. Mit einem Gradientenabstiegsverfahren kann nun versucht werden, möglichst schnell ein globales Minimum der Fehlerfunktion zu finden. d.h. eine Konfiguration der Gewichte, bei der die Fehlersumme über alle Trainingsmuster minimal wird ([Zel94], S. 106). Dies ist das Prinzip von *Backpropagation* ([RGW86]), des am meisten eingesetzten Lernverfahrens zum Training von mehrschichtigen Neuronalen Netzen. Allerdings weisen Gradientenverfahren mitunter ein schlechtes Konvergenzverhalten, und damit die zugehörigen Netze eine lange Trainingszeit auf, da das Abstiegsverfahren etwa in lokalen Minima hängenbleiben kann oder zu lange auf flachen Plateaus verweilt. Insbesondere in den letzten Jahren kamen modifizierte Versionen von Backpropagation –z.B. Abstieg mit variabler Schrittweite (vgl. z.B. [Zel94], S. 115 ff.)– zum Einsatz. Aber auch ganz andere Netztopologien finden – so auch in der vorliegenden Arbeit – zunehmend Verwendung.

Wurde das Training des Netzes erfolgreich abgeschlossen, so können anschließend für den eigentlichen Klassifikationsprozeß dem Netz dann die übrigen Datensätze (Testmenge) präsentiert werden, so daß – im Idealfall – jeder Datensatz genau einer der N Klassen zugeordnet wird.

Neben ihrer Lernfähigkeit ist der Erfolg Neuronaler Netze auf folgende drei Charakteristika zurückzuführen ([Lip87], [Hay94], [HH93]):

- Neuronale Netze besitzen, sofern sie richtig trainiert wurden, aufgrund ihrer Struktur die inhärente Eigenschaft der automatischen Generalisierung. Neuronale Netze reagieren gegenüber verrauschten Daten oder Störungen in den Eingabemustern meist weniger empfindlich als konventionelle Algorithmen ([Zel94], S. 27). Gerade für die Mustererkennung unter realen Bedingungen kommt der Eigenschaft, ein bestimmtes Muster trotz Rauschen und Verzerrungen identifizieren zu können, eine große Bedeutung zu ([Was89], S.2). Bezogen auf die Klassifikation von subantarktischen Algen bedeutet die Fähigkeit zur Generalisierung die Möglichkeit, zu klassifizierende Objekte auch dann bestimmten Klassen richtig zuzuordnen, wenn die zur Entscheidung anstehenden Muster nicht vollständig mit den vorher gelernten Trainingsmustern übereinstimmen.

- Neuronale Netze machen schwächere *a piori*–Annahmen über die statistische Verteilung der einzelnen Klassen im Datensatz als ein parametrischer Bayes-Klassifikator. Der Erfolg herkömmlicher statistischer Klassifikationsverfahren wie die *Maximum–Likelihood–Methode* hängt stark davon ab, inwieweit die Verteilung der Grauwerte einer Klasse der einer Normalverteilung entspricht. Weicht die Verteilung von einer Normalverteilung signifikant ab, so sind Fehlklassifizierungen nicht auszuschließen. Gerade für viele reale Algenkategorien läßt sich die Annahme einer Normalverteilung jedoch häufig nicht bestätigen. Neuronale Netze dagegen können auch Klassen identifizieren, die nicht der Normalverteilungsvoraussetzung genügen.

- Neuronale Netze sind in der Lage, auch nichtlineare Entscheidungsgrenzen im Merkmalsraum nachzubilden. Die Abbildungen 3.3 und 3.4 verdeutlichen diese Fähigkeit.

In Abbildung 3.3 kann man keine Gerade im zweidimensionalen Merkmalsraum so plazieren, daß die beiden Klassen A und B exakt voneinander getrennt werden. Die beiden Mengen sind nicht linear separierbar. Mit Hilfe von mehrschichtigen Neuronalen Netzen ist es dagegen möglich, auch konvexe und konkave Polygone zu klassifizieren [1] (Abbildung 3.4.)

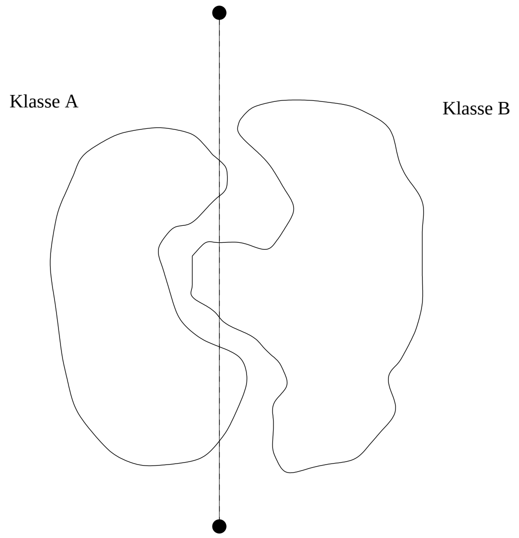


Abbildung 3.3: *Nicht-linear separierbare Klassen A und B.*

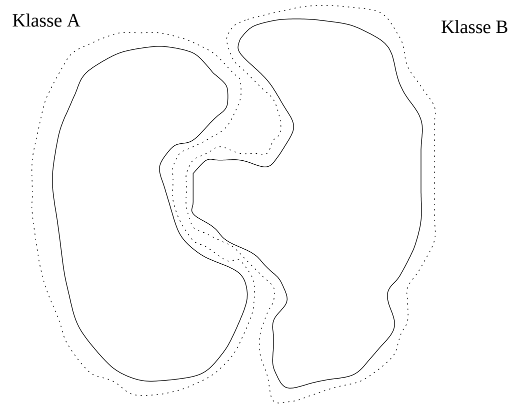


Abbildung 3.4: *Abgrenzung durch mehrschichtige Neuronale Netze.*

3.3 Geometrisches Netz

3.3.1 Vorwärtsgerichtete Neuronale Netze

Die Aufgabe, die ein Neuronales Netz lernen soll, kann allgemein als Problem der Approximation einer Funktion formuliert werden. Die Klassifikation ist dabei ein Spezialfall,

[1] Für eine ausführliche Darstellung der Trennbarkeit von Neuronalen Netzen vgl. z.B. [Was89], S. 33 ff., [Kra90], S. 145 ff. und [Zel94], S. 99 ff.

bei dem der Bildbereich der zu lernenden Funktion eine endliche Menge ist. Die zu lernende Funktion $F : \mathbb{R}^N \to \mathbb{R}^P$ ist nicht analytisch beschrieben, sondern lediglich durch K Abtastpunkte (x^k) und zugehörige Funktionswerte (y^k) gegeben, welche von einem (häufig unbekannten) Prozeß stammen. Die Menge all dieser Prozeßdaten sei mit S bezeichnet. Wie wir erst später sehen werden, ist ein Neuronales Netz mit einer versteckten Schicht und linearen Ausgabeknoten, wie es Abbildung 3.5 zeigt, ausreichend, um universelle Approximatoren zu erhalten. Wir können uns also ganz auf die versteckte Schicht konzentrieren.

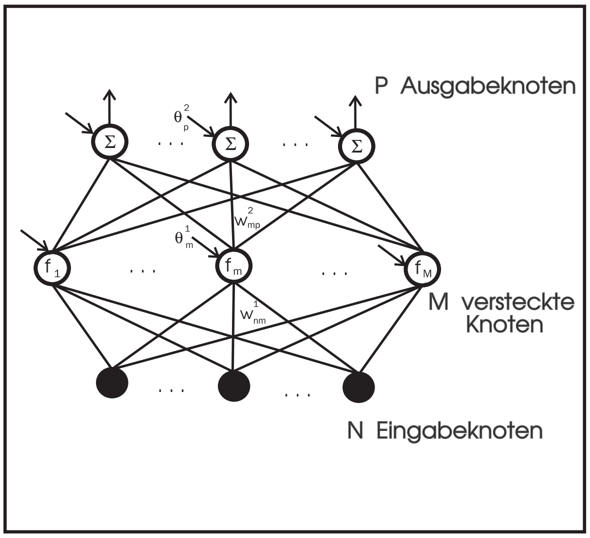


Abbildung 3.5: *Vorwärtsgerichtetes Neuronales Netz*

Jeder der M versteckten Knoten repräsentiert eine *Basisfunktion*

$$f_m : \mathbb{R} \to \mathbb{R} \qquad (1 \leq m \leq M)\,. \tag{3.1}$$

Die Gewichte w^1_{nm} und die Schwellwerte θ^1_m sind die Parameter dieser Basisfunktionen. Mit den Verbindungen von der versteckten Schicht zu den Ausgabeknoten sind Gewichte w^2_{mp} assoziiert, θ^2_p sind die Schwellwerte der Ausgabeknoten. Die oberen Indizes geben die Schicht im Netzwerk an, für die unteren gilt: $(1 \leq n \leq N)$, $(1 \leq m \leq M)$ und $(1 \leq p \leq P)$. Also bezeichnet w^2_{mp} das Gewicht vom m–ten Knoten der versteckten Schicht zum p–ten Ausgabeknoten und so weiter. Ferner sei $w^1 := \{w^1_{nm}, \theta^1_m\}$, $w^2 := \{w^2_{mp}, \theta^2_p\}$ und schließlich $\omega := \{w^1, w^2\}$ die Menge aller Parameter des Netzes. Desweiteren wollen wir noch wie üblich als Konvention vereinbaren, daß Mengenbildung über fehlende Indizes zugelassen ist.

Die Funktionalität des Netzes ist durch

$$\mathcal{N}^S_\omega : \mathbb{R}^N \to \mathbb{R}^P\,, \tag{3.2}$$

gegeben, das durch $\mathcal{N}^S_\omega$ berechnete p–te Ausgabedatum zu einem Eingabedatum $x^k = (x^k_1, \ldots, x^k_N)$ ist

$$\mathcal{N}^S_\omega(x^k)_p = \sum_{m=1}^{M} w^2_{mp} \cdot f_m(w^1_m; \theta^1_m; x^k) + \theta^2_p\,. \tag{3.3}$$

Im folgenden sei mit $\mathcal{N}_\omega^S$ stets auch das Netz selbst bezeichnet, d.h. wir identifizieren es mit seiner Berechnungsfunktion. Zur besseren Unterscheidung und als Abkürzung für eine Netzausgabe definieren wir zusätzlich noch $o_p := \mathcal{N}_\omega^S(x^k)_p$.

Für unsere Experimente haben wir zwei spezielle Typen vorwärtsgerichteter Neuronaler Netze benutzt. Zum einen das DCS (Dynamic Cell Structures) und eine am Lehrstuhl entwickelte Erweiterung des MLPs — das Geometrische Netz. Bei diesem sind im Gegensatz zum MLP alle Größen Elemente einer Clifford Algebra, anstelle reeller Werte. Wir beginnen zunächst mit einer kurzen Vorstellung des MLPs.

Ein reelles MLP können wir mit den zu Beginn eingeführten Notationen wie folgt angeben:

$$\mathcal{N}_\omega^S(x^k)_p = \sum_{m=1}^{M} w_{mp}^2 \cdot f_m\Big(\sum_{n=1}^{N}(w_{nm}^1 \cdot x_n^k) + \theta_m^1)\Big) + \theta_p^2 \,. \tag{3.4}$$

Die Funktionen f_m heißen Aktivierungsfunktionen. Sie ermöglichen eine nicht–lineare Modellierung, im Falle der Klassifikation insbesondere die Bildung nicht–linearer Trennfunktionen durch das Netz.

Die am häufigsten verwendeten Aktivierungsfunktionen sind die sogenannten *Sigmoiden Funktionen*. Sie lassen sich über

$$\sigma_\beta(x) := \frac{1}{1+\exp(-\beta x)} \qquad \beta \in \mathbb{R} \,. \tag{3.5}$$

generieren. Für $\beta = 1$ erhalten wir die sogenannte logistische Funktion, die mit σ bezeichnet werde. Ein weitere Vertreter ist die tanh–Funktion, die sich aus 3.5 wie folgt ergibt

$$\tanh(x) = 2\sigma_2 - 1 \,. \tag{3.6}$$

Die Graphen dieser beiden Funktionen sind in Abbildung 3.6 dargestellt.

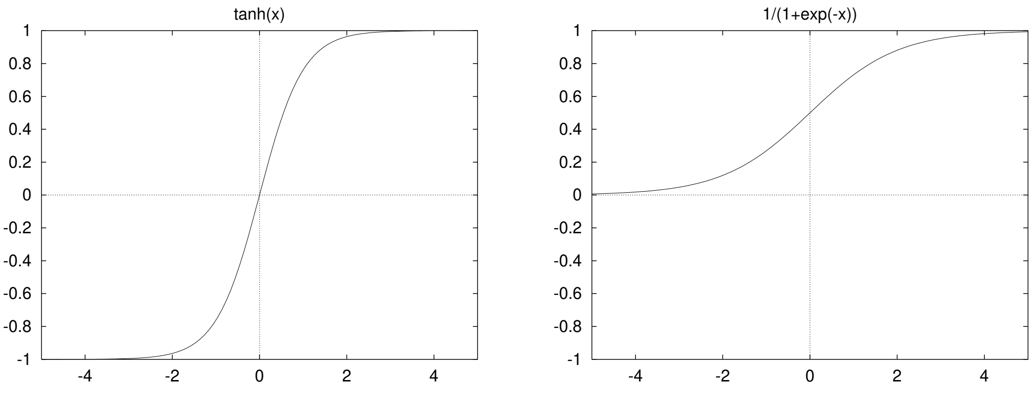


Abbildung 3.6: *Graphen gebräuchlicher Aktivierungsfunktionen*

Das meist angewandte Trainingsverfahren für MLPs ist der sogenannte *Backpropagation–Algorithmus*. Es setzt die Existenz der partiellen Ableitungen der verwendeten Aktivierungsfunktionen voraus, welche für sigmoide Funktionen stets gegeben ist. Mehr noch, im Falle der logistischen Funktion σ ist die Ableitung besonders einfach

$$\dot{\sigma}(\cdot) \; = \; \sigma(\cdot) \; \cdot \; (\, 1 - \sigma(\cdot)\,)\;. \tag{3.7}$$

Ziel des Backpropagations ist die Minimierung einer vorgegebenen Fehlerfunktion, meist des quadratischen Fehlers

$$E(y,o) = \frac{1}{2}\,\|y - o\|^2 \tag{3.8}$$

zwischen erwarteter Netzausgabe y und tatsächlicher Netzausgabe o. Die dazu notwendige Anpassung der Gewichte erfolgt lokal und iterativ. Die Herleitung der konkreten Änderungsvorschrift ergibt sich dabei durch Gradientenabstieg

$$\Delta\,\omega = -\nabla_{\omega} E(y,o)\,. \tag{3.9}$$

Bezeichne o_j jetzt allgemein die Ausgabe des j–ten Neurons in einer beliebigen Schicht und net_j dessen Netto–Eingabe.

Zunächst erfolgt die Adaption der Gewichte der Ausgabeschicht gemäß

$$\frac{\partial\,E}{\partial\,w^2_{mp}} = \delta_p\,\frac{\partial\,net_p}{\partial\,w^2_{mp}} \quad \text{mit} \quad \delta_p := \frac{\partial\,E}{\partial\,o_p}\cdot\frac{\partial\,o_p}{\partial\,net_p}\,, \tag{3.10}$$

danach werden die partiellen Ableitungen der Gewichte der versteckten Schicht berechnet

$$\frac{\partial\,E}{\partial\,w^1_{nm}} = \delta_m\,\frac{\partial\,net_m}{\partial\,w^1_{nm}} \quad \text{mit} \quad \delta_m := \sum_{p=1}^{P}\delta_p\cdot\frac{\partial\,net_l}{\partial\,o_m}\cdot\frac{\partial\,o_m}{\partial\,net_m}\,. \tag{3.11}$$

Der Backpropagation–Algorithmus für MLPs soll hier nicht weiter vertieft werden. Wir wollen uns nun den Geometrischen Netzen zuwenden. Deren Trainingsalgorithmus, der auch als eine Erweiterung des einfachen Backpropagation–Algorithmus aufgefaßt werden kann, werden wir dann etwas genauer betrachten.

Der Vollständigkeit halber sei hier noch erwähnt, daß sich mit sigmoiden Aktivierungsfunktionen jede stetige reellwertige Funktion in einer n–dimensionalen reellen Variablen beliebig genau approximieren läßt [Cyb89].

3.3.2 Das Geometrische Netz

Das Geometrische Netz verarbeitet keine reellen Daten sondern Elemente (*sogenannte Multivektoren*) einer Clifford Algebra. Diese können z.B. Vektoren sein, aber auch weitere

Entitäten höherer Ordnung. Für die Klassifikation könnten sich daher interessante Vorteile ergeben.

Die Funktionalität eines Geometrischen Netzes ist gegeben durch:

$$\mathcal{CN}_{\omega}^{S} : (\mathcal{C}_{n,s})^N \to (\mathcal{C}_{n,s})^P \,. \tag{3.12}$$

Wir betrachten hier Geometrische Netze mit der selben Architektur wie in Abbildung 3.5, d.h. mit linksseitiger Gewichtsmultiplikation. Der durch ein derartiges Netz $\mathcal{CN}_{\omega}^{S}$ berechnete p–te Ausgabemultivektor zu einem Eingabemultivektor $x^k = (x_1^k, \ldots, x_N^k)$ ist

$$\mathcal{CN}_{\omega}^{S}(x^k)_p = \sum_{m=1}^{M} w_{mp}^2 \otimes_{n,s} F_m(w_m^1; \theta_m^1; x^k) + \theta_p^2 \,, \tag{3.13}$$

wobei die Aktivierungsfunktionen die Funktionalität

$$F_m : \mathcal{C}_{n,s} \to \mathcal{C}_{n,s} \qquad (1 \leq m \leq M) \tag{3.14}$$

haben. In unserem Fall findet als Aktivierungsfunktion die logistische σ Funktion in jeder Komponente Verwendung. Diese sei daher mit $\boldsymbol{\sigma}$ bezeichnet.

Im folgenden geben wir den Backpropagation–Algorithmus für Geometrische Netze aus [Buc97] wieder.

Bezeichne nun N, M und P die Anzahl der Eingabeknoten, der versteckten Knoten und der Ausgabeknoten des Netzes, respektive. Weiterhin sei w_{nm}^1 das Gewicht, welches das n–te Neuron der Eingabeschicht mit dem m–ten Neuron der versteckten Schicht verbindet. Analog dazu, sei w_{mp}^2 das Gewicht vom m-ten versteckten Neuron zum p–ten Neuron der Ausgabeschicht. Alle Gewichte sind dabei Elemente einer Clifford Algebra.

Unter Benutzung der obigen Notationen erhalten wir:

- Aktivierungs– und Ausgabewert eines versteckten Knotens

$$S_m^1 := \sum_{n=1}^{N} w_{nm}^1 \otimes x_n + \theta_m^1 \tag{3.15}$$

$$h_m := \boldsymbol{\sigma}(S_m^1) \tag{3.16}$$

- Aktivierungs– und Ausgabewert eines Ausgabe–Knotens

$$S_p^2 := \sum_{m=1}^{M} w_{mp}^2 \otimes h_m + \theta_p^2 \tag{3.17}$$

$$o_p := \boldsymbol{\sigma}(S_p^2) \,. \tag{3.18}$$

Die Minimierung der Fehlerfunktion

$$E = \frac{1}{2} \sum_{p=1}^{P} (y_p - o_p)^2 , \tag{3.19}$$

wobei $y = (y_1, \ldots, y_p)$ den erwarteten Ausgabewert bezeichne, erfolgt nun durch Anwendung des Gradientenabstieges in den Gewichten. Im ersten Schritt bestimmen wir dazu die partiellen Ableitungen der Gewichte der Ausgabeschicht

$$\nabla E_{w^2_{mp}} = \sum_{A \in \mathcal{A}} \frac{\partial E}{\partial [w^2_{mp}]_A} \, e_A \, . \tag{3.20}$$

Die Anwendung der Kettenregel liefert dabei (3.20)

$$\frac{\partial E}{\partial [w^2_{mp}]_A} = \sum_{B \in \mathcal{A}} \frac{\partial E}{\partial [S^2_p]_B} \frac{\partial [S^2_p]_B}{\partial [w^2_{mp}]_A} \, . \tag{3.21}$$

Die partiellen Ableitungen der Fehlerfunktion nach den Aktivierungen S^2_p der Ausgabeknoten ergeben sich zu

$$\frac{\partial E}{\partial [S^2_p]_B} = \frac{\partial E}{\partial [y_p]_B} \frac{\partial [y_p]_B}{\partial [S^2_p]_B} = ([y_p]_B - [o_p]_B) \, \dot{\boldsymbol{\sigma}}([S^2_p]_B) \, . \tag{3.22}$$

Die Berechnung der verbleibenden partiellen Ableitungen ist formal etwas schwieriger, da sie direkt abhängig von der verwendeten Clifford Algebra ist. Wir beschränken uns daher nur auf die Angabe des Ergebnisses

$$\frac{\partial [S^2_p]}{\partial w^2_{mp}} = \overline{h_m} \, , \tag{3.23}$$

wobei $\overline{h_m}$ eine ausgezeichnete und eindeutig bestimmte Involution ist. Für diesbezüglich weiterführenden Details sei auf [Buc97] verwiesen.

Insgesamt, unter Benutzung des Symbols $\odot$ für die komponentenweise Multiplikation, erhalten wir damit die folgende Formel für die Änderung der Gewichte der Ausgabeschicht

$$\Delta w^2_{mp} = [\, \underbrace{(y_p - o_p) \odot \dot{\sigma}(S^2_p)}_{\delta^2_p} \,] \otimes \overline{h}_m \, . \tag{3.24}$$

Analog zur obigen Herleitung ergibt sich für die Änderung der Gewichte der versteckten Schicht

$$\Delta w^1_{nm} = [\, \underbrace{(\sum_{p=1}^{P} \overline{w}^2_{mp} \otimes \delta^2_p) \odot \dot{\sigma}(S^1_m)}_{\delta^1_m} \,] \otimes \overline{x}_n \, . \tag{3.25}$$

Die Änderung der Schwellwerte der Neuronen der einzelnen Schichten berechnet sich schließlich zu

$$\Delta \, \theta^2_p = \delta^2_p \quad \text{and} \quad \Delta \, \theta^1_m = \delta^1_m \, . \tag{3.26}$$

3.3.3 Integration des Merkmalsvektors in die Geometrische Algebra

Eine Geometrische Algebra hat, als reeller Vektorraum aufgefaßt, stets eine Zweierpotenz 2^n als Dimension. Für die Geometrischen Netze hat dies die unmittelbare Konsequenz, daß alle Ein– und Ausgabeknoten, sowie natürlich auch alle Gewichte als Multivektoren 2^n Komponenten besitzen. Eine einfache Kodierung jedes Merkmals in genau einem Eingabeneuron wie bei herkömmlichen MLPs ist daher nicht möglich. So muß beispielsweise der 7-dimensionale Merkmalsvektor geeignet in eine Geometrische Algebra eingebettet werden.

Da es mehrere Geometrische Algebren der Dimension 2^n gibt — genau $n+1$ —, ist mit der Wahl der Dimension als Parameter des Geometrischen Netzes allein noch nicht eindeutig ein solches bestimmt. Auch auf diese über die Topologie des Netzes hinausgehende Fragen zur Integration der Daten werden wir nun im folgenden eingehen.

Eine Geometrische Algebra ist grob gesagt, eine spezielle Einbettung eines metrischen n–dimensionalen Vektorraumes in einen 2^n–dimensionalen Vektorraum. Dieser enthält eine Kopie des Ursprungsvektorraums aber auch daraus generierte Entitäten höherer Dimension. Unsere Merkmalsvektoren sind im Prinzip als Elemente (Punkte) des $\mathbb{R}^n$ anzusehen. Für den Extremfall der 32 Fourierdeskriptoren bedeutete dies, daß bei obiger Interpretation eine 2^{32} dimensionale Geometrische Algebra erforderlich wäre. Dies wäre natürlich hier, aber auch schon bei kleineren Dimensionen des Merkmalsvektors, völlig unpraktikabel.

Eine derartige Einbettung wäre aber nur notwendig, wenn man wirklich die Daten entsprechend weiter durch das Netz verarbeiten wollte, z.B. gezielt orthogonale Transformationen als Propagierungsfunktionen in den Neuronen verwenden wollte. Dafür gibts es aber in unserem Fall keine Notwendigkeit. Daß man auch bei jeder anderen als der obigen Einbettung sich theoretisch korrekt verhält, ist durch die universelle Approximationseigenschaft der Geometrischen Netze garantiert [Buc97]. Das heißt, wir können einfach die bezüglich der Nutzung der Parameter des Netzes effektivste Einbettung wählen.

Unter diesem Gesichtspunkt können wir auch unter den Algebren einer festen Dimension frei wählen, da unsere Daten keine Präferenzen aufweisen. An dieser Stelle sei vorweggenommen, daß sich dies bei stichprobenartigen Experimenten auf den verschiedenen Merkmalsvektoren bestätigt hat.

Durch das Vorliegen eines 8–Klassenproblems, d.h. die Ausgabe ist automatisch eine Zweierpotenz, ist natürlich maximal eine 8–dimensionale Algebra zu verwenden. Bei Merkmalsvektoren, deren Dimension ungleich acht ist, sind entsprechend viele Eingabeneuronen zu verwenden, wobei beim letzten die Eingabe gegebenenfalls mit Nullen aufgefüllt wird. Damit ergibt sich folgende beispielhafte Integration eines Merkmalsvektors $\vec{x}$ und Ausgabevektors $\vec{y}$:

$$(x_{11}, x_{12}, \ldots, x_{18}), \ldots, (x_{n1}, x_{n2}, 0, 0, 0, 0, 0, 0) \mid (y_1, \ldots, y_8) \tag{3.27}$$

für $n \in \mathbb{N}$.

Aus der Darstellung ist ersichtlich, daß für $n > 2$ eine durch die Eingabeneuronen stärkere Zusammenfassung bestimmter Merkmale erfolgt. Dies kann in Hinblick auf die vom Netz zu leistende Diskriminierung vorteilhaft sein. Um dies genauer untersuchen zu können, haben wir auch Experimente in einer $4-$*dimensionalen* Algebra durchgeführt, wobei sich mit obigen Notationen folgende beispielhafte Repräsentation für das Geometrische Netz ergibt:

$$(x_{11}, x_{12}, x_{13}, x_{14}), \ldots, (x_{n1}, x_{n2}, 0, 0) \mid (y_1, y_2, y_3, y_4), (y_5, y_6, y_7, y_8)\,. \tag{3.28}$$

Bei den Experimenten fanden die Algebren $\mathcal{C}_{2,0}$ und $\mathcal{C}_{3,0}$ Verwendung.

3.4 Das DCS-Netz

Dynamic Cell Structures, im folgenden kurz DCS–Netze genannt, sind eine Erweiterung *Radialer Basisfunktionsnetze* (RBF–Netze) um die Verwendung *topologischer Karten* [Koh95]. Das DCS–Netz wurde von J. Bruske [BS94] am Lehrstuhl entwickelt. Eines seiner Hauptcharakteristika ist seine dynamische Topologie während des Trainings. Dies und seine Wurzeln in der RBF–Architektur unterscheiden es sehr wesentlich von den schon vorgestellten Geometrischen Netzen, die auf der MLP–Architektur basieren.

Wie werden im folgenden nur einen groben Überblick über DCS–Netze geben, da sie schon mehrfach Gegenstand von Diplomarbeiten waren. Dabei werden wir uns auch an diesen orientieren, beginnend mit einer kurzen Erläuterung der RBF–Architektur.

Ein RBF–Netz ist ein vorwärtsgerichtetes Neuronales Netz, und wird damit auch durch die Abbildung 3.5 wiedergeben. Im Gegensatz zu einem MLP hat es stets nur eine verdeckte Schicht und eine lineare Ausgabeschicht. Seine Funktionalität läßt sich daher analog zu (3.4) mit den dort verwendeten Notationen beschreiben durch

$$\mathcal{N}_\omega^S(x^k)_p = \sum_{m=1}^{M} w_{mp} \cdot f_m(c_m; \sigma_m; x^k) + \theta_p\,. \tag{3.29}$$

In einem RBF–Netz ist f nun eine sogenannte *Radiale Basisfunktion* der Paramter $c_m; \sigma_m$. Am häufigsten kommen dabei meist Gaußfunktionen

$$\exp\left(-\frac{||x^k - c_m||^2}{\sigma_m^2}\right) \tag{3.30}$$

zum Einsatz. Die Aktivierung eines RBF–Knotens ist also lokal, genauer umso stärker je näher die Eingabe am Zentrum c_m liegt, unabhängig von der Richtung. Der Normalisierungsparameter σ_m (oder auch die Varianz) bestimmt, wie weit eine Eingabe entfernt sein kann, um noch eine signifikante Aktivierung auszulösen.

In einem RBF–Netz wird über alle Knoten summiert, obwohl aufgrund der Lokalität der Aktivierungsfunktionen nur wenige Knoten jeweils einen signifikanten Beitrag leisten werden. Es ist daher sehr natürlich, nur diese Knoten zu betrachten. Dies sind vereinfacht jene Knoten, die zum Neuron, das die Eingabe am besten repräsentiert, der sogenannten *best matching unit* (bmu) in Nachbarschaft liegen. Diese seien mit $N(bmu)$ bezeichnet. Die so gegenüber (3.30) modifizierte Aktivierungsfunktion lautet dann

$$\begin{cases} \exp\left(-\frac{||\mathbf{x^k}-\mathbf{c_m}||^2}{\sigma_m^2}\cdot(-\ln p)\right) & : \quad m \in \{bmu(x^k)\} \cup N(bmu(x^k)) \\ 0 & : \quad sonst \end{cases}, \tag{3.31}$$

wobei p ein konstanter Faktor ist.

Die Fallunterscheidung in (3.31) bewirkt nun gerade die gewünschte Eigenschaft, daß nur der Eingabe nahe Knoten zur Ausgabe beitragen. Die Breite der modifizierten Gaußfunktionen ergibt sich optimal in Abhängigkeit von der mittleren quadratischen Entfernung zu den benachbarten Knoten

$$\sigma_m^2 = \frac{\sum_{j\in N(i)} ||c_i - c_j||^2}{|N(i)|}. \tag{3.32}$$

Daraus kann man ableiten, daß der Faktor p die Aktivierung des Neurons i in der Nähe der Zentren benachbarter Neuronen angibt. Er bestimmt also letztendlich die Überlappung benachbarter Neuronen.

Bei einem RBF–Netz wird die Topologie des Netzes, d.h. im wesentlichen die Anzahl der Knoten in der verdeckten Schicht, nicht automatisch vorgegeben, sondern meist unüberwacht während des Trainings gelernt. Bei einem DCS–Netz ist aber nicht nur die Knotenzahl sondern auch die Verbindungen selbst dynamisch. Auf die Algorithmen zum Lernen der Topologie eines DCS–Netzes werden wir hier nicht detaillierter eingehen, es sei dazu auf [Trö97], [AM98] verwiesen.

Die auf diese Art gelernte Topologie kann noch weiter verwendet werden, um das Netz effizienter zu gestalten. Oftmals, ist es nicht nötig den gesamten Eingaberaum zu betrachten. Vielfach sind nur wenige Komponenten der Daten für die Klassifizierung bereits ausreichend. Dann spannen die Daten nicht den gesamten Eingaberaum auf, sondern nur einen niedriger dimensionalen Teilraum. Dessen Dimension wird auch als sogenannte *intrinsischen Dimensionalität* der Daten bezeichnet. Die Berechnungen im intrinsischen Teilraum lassen sich also nahezu verlustfrei und weit effizienter durchführen. Die Bestimmung dieses Raumes läßt sich sowohl global (z.B. Hauptkomponentenanalyse) als auch lokal durchführen, indem man dort die Nachbarschaftsbeziehungen der Daten auswertet. Das sogenannte SDCS–Netz (*Subspace Dynamic Cell Structures*) realisiert dies nun gerade als Erweiterung des DCS–Netzes.

Dazu ist im wesentlichen folgende Anpassung der Aktivierungsfunktion eines Knotens i (3.31) notwendig

$$\begin{cases} \exp\left(-\frac{||E_i x - E_i c_i||_2^2}{\sigma_i^2}\cdot(-\ln p)\right)\cdot\frac{1}{sumAct(x)} & : \quad i \in \{bmu(x)\} \cup N(bmu(x)) \\ 0 & : \quad sonst \end{cases}, \tag{3.33}$$

wobei

$$sumAct(x) = \sum_{l \in \{bmu(x)\} \cup N(bmu(x))} \exp\left(-\frac{||E_l x - E_l c_l||^2}{\sigma_l^2} \cdot (-\ln p)\right) \tag{3.34}$$

die Summe der unnormierten Aktivierungen bezeichnet und

$$\sigma_i^2 = \frac{\sum_{j \in N(i)} ||E_i c_i - E_i c_j||^2}{|N(i)|} \tag{3.35}$$

ist und E_i die Matrix der normierten Eigenvektoren der Korrelationsmatrix des Knotens i ist.

Das SDCS–Netz wurde im Hinblick auf diese mögliche Datenreduktion für die Experimente gewählt.

3.5 Vorverarbeitung der Merkmalsvektoren

Bevor wir uns dem experimentellen Teil dieser Arbeit widmen und im Kapitel 4 die Versuchsreihen beschreiben, ist es notwendig, die zu testenden Merkmalsvektoren einigen Vorverarbeitungsschritten zu unterziehen. Hierzu zählen eine Skalierung der Daten, eine Reduktion von Komponenten innerhalb der Merkmalsvektoren und die Transformation der Datendarstellung für die verschiedenen Netztypen.

3.5.1 Skalierung

Neuronale Netze und andere statistische Verfahren setzten die Existenz einer n-dimensionalen Wahrscheinlichkeitsverteilung

$$p(\boldsymbol{d}),\ \boldsymbol{d} \in \mathbb{R}^{\mathrm{n}} \tag{3.36}$$

oder einer endlichen Datenmenge aus n-dimensionalen Vektoren

$$D = \{\boldsymbol{d}_1, \ldots, \boldsymbol{d}_m\},\ \boldsymbol{d}_i \in \mathbb{R}^{\mathrm{n}} \tag{3.37}$$

vorraus. Diese Daten stehen nicht 'einfach so' zur Verfügung, sondern werden durch geeignete Vorverarbeitungsprozesse (in vorrigen Abschnitten vorgestellt) aus den Rohdaten gewonnen. Die Trainingsdaten entstehen also häufig aus einer viel höherdimensionalen Wahrscheinlichkeitsverteilung

$$p(\boldsymbol{x}),\ \boldsymbol{x} \in \mathbb{R}^l,\ l > n \tag{3.38}$$

oder Datenmenge

$$D_{orig} = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_m\},\ x_i \in \mathbb{R}^l. \tag{3.39}$$

Die von uns verwendeten Rohdaten stammen z.B. aus den Elektronenmikroskopaufnahmen, oder aus verschiedenen Polarökologischen Lehrbücher. Rohdaten können kontinuierlich oder diskret sein und sind mit verschiedenen Maßeinheiten behaftet.

Die Vorverarbeitung dieser Daten ist oft von entscheidender Bedeutung für den Erfolg der weiteren Bearbeitungsschritte; dort gemachte Fehler können in der Regel nicht mehr korrigiert werden. Nur in den seltensten Fällen kann man auf eine Vorverarbeitung gänzlich verzichten und die Rohdaten unverändert als Trainingsdaten verwenden.

Neben der *Binarisierung*, der Bearbeitung mit *morphologischen Operatoren*, *Kontur–extraktions–* oder *Füllalgorithmen* hat sich eine **Skalierung** der einzelnen Komponenten der Datenvektoren als sinnvoll erwiesen. Abhängig von der Bedeutung der einzelnen Komponenten und den verwendeten Maßeinheiten können sie völlig unterschiedliche Wertebereiche haben. Dies wird am deutlichsten bei den in der Abbildung 2.29 aufgeführten Merkmalen, die z.B für die Länge und Breite des umfaßenden Rechtecks Werte zwischen 0 und 512 annehmen können, für die Exzentrizität Werte zwischen null und eins und die Momente belaufen sich bei der gefüllten Kontur im Millionenbereich. Ohne Vorverarbeitung bildet jede dieser Komponenten eine der Koordinaten des Vektorraumes, in dem vektorbasierte Neuronale Netze (z.B RBF-Netz, Nächster Nachbar) Distanzen messen. Bei den Merkmalsvektoren, die Momente zweiter Ordnung beinhalten, würden sich die Komponenten *Momente auf die gefüllte Kontur* völlig dominant auswirken, wodurch das Ziel der *Gleichberechtigung*, bzw. der *gleichen Gewichtung* der einzelnen Komponenten des Merkmalsvektors verfehlt wäre. Es ist also sinnvoll, sich von eher zufälligen Einflüssen, wie der Wahl der Maßeinheiten unabhängig zu machen. Ein übliches Mittel ist die Skalierung aller Komponenten derart, daß sie eine gemeinsame Varianz haben, z.B den Wert eins. Meistens wird aus Darstellungsgründen zusätzlich noch von jeder Komponente der Mittelwert abgezogen, so daß die transformierten Daten in jeder Komponente den Mittelwert null und die Varianz eins haben. Für die i-te Komponente $\boldsymbol{x}_i$ des Datenvektors $\boldsymbol{x}$ ergibt sich so die Transformation

$$x_i' = \frac{x_i - \mu_i}{\sigma_i} \tag{3.40}$$

wobei μ_i der Mittelwert und σ_i die Standardabweichung der i-ten Komponente aller Vektoren ist (im Falle einer endlichen Datenmenge).

Manchmal existiert Vorwissen über die Daten, welches besagt, daß die Bedeutung der einzelnen Komponenten unterschiedlich sein kann, daß einigen Komponenten des Merkmalsvektors eine größere Bedeutung zugesprochen wird. In solchen Fällen können die jeweiligen normierten Komponenten noch mit entsprechenden Faktoren g_i multipliziert (gewichtet) werden:

$$x'_i = g_i \, \frac{x_i - \mu_i}{\sigma_i} \tag{3.41}$$

Vektorbasierte Neuronale Netze sind sehr empfindlich bezüglich der Skalierung der Daten, da sie Distanzen in dem ihnen vorgegebenen Vektorraum messen. Es sei angemerkt, daß andere Neuronale Netze, z.B. mehrschichtige Perzeptrons die Fähigkeit besitzen, durch Wahl geeigneter Gewichte, selbst eine Skalierung der Daten vorzunehmen. Somit wäre eine Vorskalierung nicht unbedingt erforderlich.

3.5.2 Reduktion der Komponenten

Es müssen nicht alle Komponenten des ursprünglichen Vektors $\boldsymbol{x}_i$ in den korrespondierenden Datenvektor $\boldsymbol{d}_i$ übernommen werden. Eine kleinere Dimension des Merkmalsvektors verringert den Berechnungsaufwand und ist daher anzustreben. Um dies näher zu untersuchen, haben wir uns bei der Zusammensetzung des Merkmalsvektors bestehend aus den Fourierdeskriptoren auf drei Untermengen konzentriert. Der ursprüngliche Algorithmus hatte bei einer äquidistanden Abtastung der Kontur bei 64 Konturpunkten, abzüglich des Null-ten Fourierdeskriptors und unter Berücksichtigung der *Spiegelung* der Daten ab der Hälfte des Vektors, einen 31–dimensionalen Merkmalsvektor als Ergebnis geliefert. Beim Training der Netze wurden probeweise Trainingsläufe mit 16–, 8– bzw. 4–dimensionalen Eingabevektoren gemacht. Es wurden also jeweils nur die ersten 16, 8 bzw. 4 Fourierdeskriptoren für das Training verwendet. Ziel war es, herauszufinden, inwieweit die *höheren* Fourierdeskriptoren, also die, die in der Fourierreihe am weitesten waren, eine Relevanz bezüglich der Klassifikationsfähigkeit haben. Eine optimale Unterauswahl von Komponenten zu treffen, ist ein extrem schwieriges Problem, da es bei l-dimensionalen ursprünglichen Datenvektoren eine Anzahl von 2^l möglichen Untermengen dieser Komponenten gibt. *A priori* Wissen kann hier helfen, eine geeignete Auswahl zu treffen. Manchmal wird vor der Ausblendung von Komponenten eine Koordinatentransformation (z.B. eine Hauptkomponentenanalyse) durchgeführt.

3.5.3 Transformation der Datendarstellung

Datum ist nicht gleich Datum! Für eine gegebene Information gibt es eine Mannigfaltigkeit, was die unterschiedlichen Arten der Darstellung betrifft, die (obwohl prinzipiell äquivalent) unterschiedlich gut geeignet sein können für die vorzunehmende Verarbeitung. Bei einem Klassifikationsproblem mit acht verschiedenen Klassen (wie in unserem Fall), mögen die Daten vorliegen als n-dimensionale Eigabevektoren mit einer Klassenbezeichnung, die als Zahl zwischen eins und acht gegeben ist. Die Dimension des Eingabevektors haben wir variabel bezeichnet, da sie je nach Art des Merkmalsvektors, die Werte 4 (**4FD**), 8 (**8FD**, **MOM7f**, **MOM7k**, **MET**, **VAR**, 16 (**16FD**, **FD–MOM7f**, **FD–MET**, **FD–VAR**, **MET–VAR**, **FD**), 32 (**32FD**, **FD–MOM7f–MET–VAR**) annehmen kann. Im allgemeinen ist es nicht ratsam, die Kodierung für die Klassenzugehörigkeit (eins bis acht, als Dezimalzahlen) so zu belassen und von einem adaptiven Verfahren die Ausgabe eines Wertes zwischen eins und acht zu erwarten. Dies würde nämlich implizieren, daß die Klasse 1 der Klasse 2 ähnlicher ist als der Klasse 7 (nur weil sie *näher dran ist*). Ist dies nicht der Fall, sollte man unbedingt eine Kodierung wählen, die sicherstellt, daß in der verwendeten Metrik je zwei Klassenbezeichnungen den gleichen *Abstand* voneinander haben. Bei dem Klassifizierungsproblem mit acht Klassen bietet sich eine 1–*aus*–8 *Kodierung* an, die die Klassenbezeichnung k als einen 8-dimensionalen Vektor darstellt, der an der k-ten Stelle mit einer eins und ansonsten mit nullen besetzt ist. In diesem Abschnitt ist noch ein kleiner Ausschnitt aus dem Eingabevektor eines Geometrischen Netzes bestehend aus vier Fourierdeskriptoren und der Klassenzugehörigkeit (sie ist noch dezimal), zu sehen. Die Datei enthält je ein Beispiel für jede Klasse. Erst nach der Trennung der Files in ein Eingabe– und Ausgabefile für das Training des Geometrischen Netzes, modellieren wir mit der oben beschriebenen Methode die unäre Kodierung der Markierung und damit der Ausgabe.

```
1.1409300000 0.3661450000 0.5292630000 0.3146330000 1
1.5562800000 6.9808100000 0.1797160000 0.9607200000 2
3.3876500000 12.4087000000 0.5010670000 1.3222900000 3
2.3491400000 5.2898800000 0.1046200000 1.2344400000 4
5.3612000000 15.5009000000 2.1049100000 1.4453800000 5
5.2380600000 14.9051000000 1.4876400000 1.0788700000 6
13.7878000000 14.8771000000 2.4727700000 2.3398700000 7
3.4841200000 16.1216000000 0.9672310000 0.6203780000 8
```

Das Ausgabefile hat dann die Form:

```
1 0 0 0 0 0 0 0
0 1 0 0 0 0 0 0
0 0 1 0 0 0 0 0
0 0 0 1 0 0 0 0
0 0 0 0 1 0 0 0
0 0 0 0 0 1 0 0
0 0 0 0 0 0 1 0
0 0 0 0 0 0 0 1
```

Bezüglich der Euklid'schen Vektornorm haben dann je zwei Klassenbezeichnungen stets den gleichen Abstand zueinander ($\sqrt{2}$). Sollte jedoch *a priori* Wissen über die Verwandschaft verschiedener Klassen zueinander existieren, kann durchaus eine Kodierung gewählt werden, die dieses Vorwissen wiederspiegelt. Dadurch ist es wahrscheinlicher, daß bei einer Fehlklassifizierung zumindest eine *ähnliche* Klasse gewählt wird.

Für die DCS-Netze muß wiederum eine andere Art der Darstellung implementiert werden. Die Eingabevektoren für das Training (und später für den Test) beinhalten sowohl die Zahlenvektoren als auch die Ausgabe, getrennt durch das Sonderzeichen $\Longrightarrow$. Ein Ausschnitt aus den Eingabevektoren der 1. Klasse verdeutlicht die unterschiedlichen Darstellungsformen:

Number_of_examples: 10
INPUTS: 4
OUTPUTS: 8
1.1409300000 0.3661450000 0.5292630000 0.3146330000 $\Longrightarrow$ 1 0 0 0 0 0 0 0
1.1045100000 1.1252100000 0.3868970000 0.4048330000 $\Longrightarrow$ 1 0 0 0 0 0 0 0
1.4381700000 0.2108860000 0.1202030000 0.1727020000 $\Longrightarrow$ 1 0 0 0 0 0 0 0
1.0857200000 0.1670140000 0.2761330000 0.2806900000 $\Longrightarrow$ 1 0 0 0 0 0 0 0
1.1943600000 0.3817880000 0.1851190000 0.3454690000 $\Longrightarrow$ 1 0 0 0 0 0 0 0
1.3553700000 1.6745800000 0.4219310000 0.0747186000 $\Longrightarrow$ 1 0 0 0 0 0 0 0
1.4439200000 2.6518600000 0.5452310000 0.4025930000 $\Longrightarrow$ 1 0 0 0 0 0 0 0
2.1437200000 0.7179450000 0.1549720000 0.1728140000 $\Longrightarrow$ 1 0 0 0 0 0 0 0
1.7116600000 1.1237400000 0.0450507000 0.2260080000 $\Longrightarrow$ 1 0 0 0 0 0 0 0
1.4295800000 3.5303000000 0.6367710000 0.3115390000 $\Longrightarrow$ 1 0 0 0 0 0 0 0

Die DCS-Files haben zusätzlich noch eine Art *header*, einen Dateikopf, der dem Netz Informationen über die Beschaffenheit der Daten liefert. In diesem Beispiel handelt es sich um zehn Testbeispiele, die Eingabedimension ist vier (4FD) und die Ausgabedimension ist acht (ebenfalls unäre Kodierung der acht Klassen).

Kapitel 4

Experimentelle Ergebnisse

In diesem Kapitel werden die Versuchsreihen vorgestellt, die durchgeführt worden sind, um die zwei Netztopologien auf ihre Tauglichkeit für eine 8-Klassen–Klassifikationsaufgabe zu testen. Anschließend werden die Ergebnisse graphisch ausgewertet und merkmals– bzw. netzbezogen analysiert. Die methodenkritische Interpretation der Ergebnisse und die darauf basierende Diskussion erfolgen erst im letzten Kapitel dieser Arbeit.

4.1 Klassifikation mittels Geometrischer Netze

Bei den Geometrischen Netzen benutzen wir zwei verschiedene Netzarchitekturen:

- Das $C_{2,0}$–Geometrische Netz basierend auf der 4–dimensionalen Clifford–Algebra der Quaternionen
- Das $C_{3,0}$–Geometrische Netz basierend auf einer 8–dimensionalen Clifford–Algebra.

Mit diesen zwei Netztypen wurden für die nachstehend aufgelisteten dreizehn Merkmalsvektoren jeweils fünf Durchläufe mit variierender Anzahl von Knoten in der versteckten Schicht durchgeführt (je nach Dimension des Merkmalsvektors haben wir fünf, acht oder sechzehn Knoten verwendet).

1. **4FD (4–dim.)**
2. **8FD (8–dim.)**
3. **16FD (16–dim.)**
4. **32FD (32–dim.)**

5. **MOM7f (8–dim.)**
6. **MOM7k (8–dim.)**
7. **MET (8–dim.)**
8. **VAR (8–dim.)**
9. **FD–MET (16–dim.)**
10. **FD–VAR (16–dim.)**
11. **MET–VAR (16–dim.)**
12. **FD–MOM7f (16–dim.)**
13. **FD–MOM7f–MET–VAR(32–dim.)**

An dieser Stelle sei vermerkt, daß diese Merkmalsvektorenliste auch für das Training der DCS-Netze im folgenden Abschnitt ihre Gültigkeit behält.

Darüberhinaus wurde eine auf dem Zufallsprinzip gestüzte Unterteilung der Gesamtdaten in Test– und Trainingsmenge für drei Kategorien vorgenommen. Diese teilen sich wie folgt auf:

- 20/80 (Test/Training)
- 50/50 (Test/Training)
- 80/20 (Test/Training).

Der Grund für diese Aufteilung ist die Beurteilung der Generalisierungsfähigkeit der Netze aufgrund von verschiedenen Datenvolumina. Somit ergibt sich eine Gesamtanzahl von 780 Testläufen für die Geometrischen Netze. Pro Merkmal wurde dann der Mittelwert aus diesen gebildet. Die so ermittelten Ergebnisse für die Geometrischen Netze sind in Form von Balkendiagrammen visualisiert worden und stellen die Klassifizierungsrate der Netze für die jeweiligen Merkmalstypen dar. In jeder Abbildung sind zum besseren Vergleich die beiden Netze $C_{2,0}$ und $C_{3,0}$ (von links nach rechts) nebeneinander aufgebaut. Die Lernparameter *Momentum* und *Lernrate* sind nach empirischen Gesichtspunkten plausibel gewählt und daher in den Graphiken nicht einzeln aufgeführt. Die Anzahl der Iterationen pro Versuch ist auch variiert worden, wobei die dargestellten Graphiken annähernd gleich viele (der Durchschnittswert liegt bei 10000 It.) hiervon enthalten. Die Ergebnisgraphiken sind in der Reihenfolge der Merkmalsvektoren wie in der obigen Liste nacheinander abgebildet und werden direkt im Anschluß wiedergegeben.

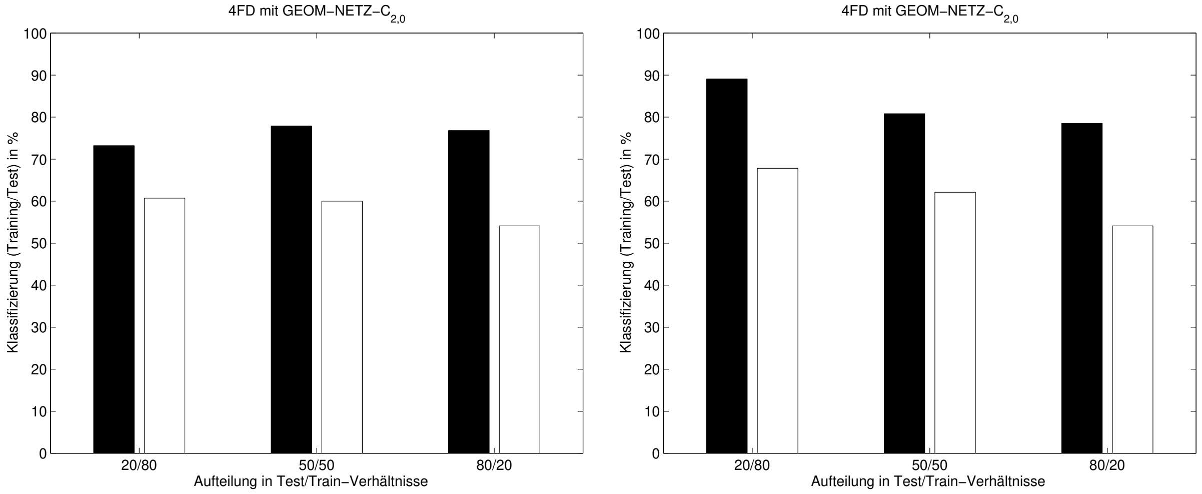


Abbildung 4.1: *Merkmal* **4FD** *trainiert mit dem Geometrischen Netz* $C_{2,0}$ *mit 4 (links) und 8 (rechts) versteckte Knoten. Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

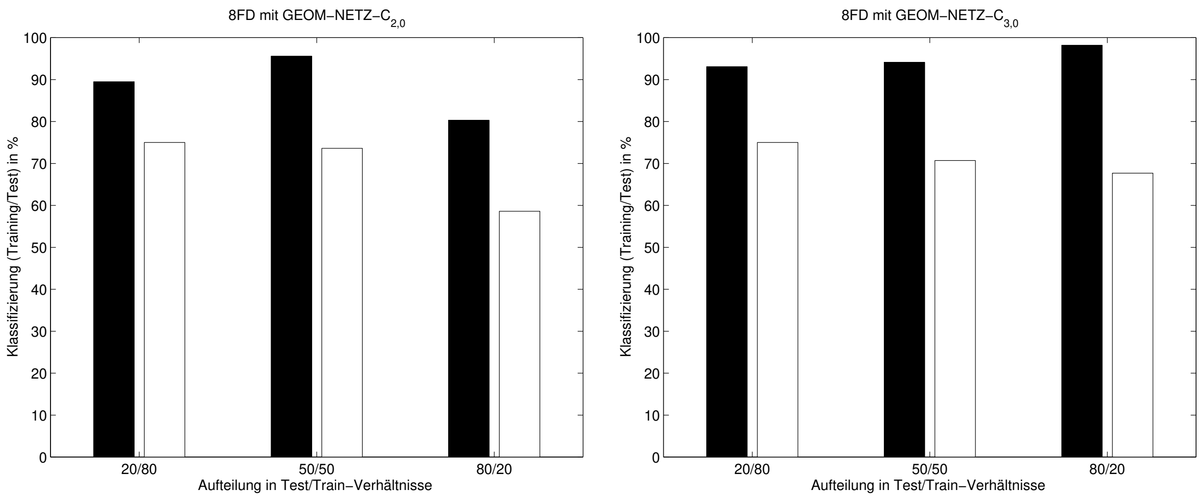


Abbildung 4.2: *Merkmal* **8FD** *trainiert mit den zwei Geometrischen Netze* $C_{2,0}$ *(links) und* $C_{3,0}$ *(rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

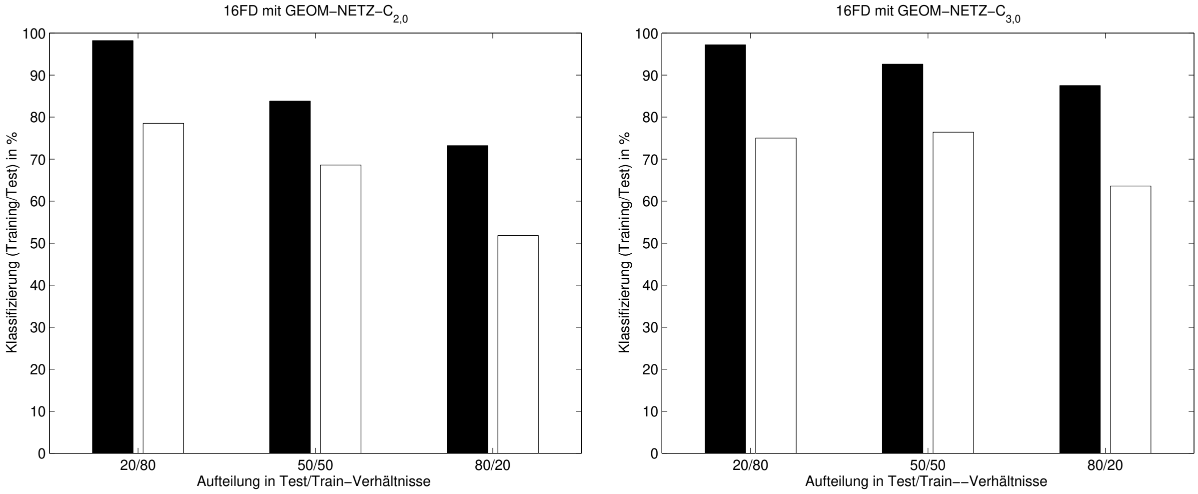


Abbildung 4.3: *Merkmal* **16FD** *trainiert mit den zwei Geometrischen Netze $C_{2,0}$ (links) und $C_{3,0}$ (rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

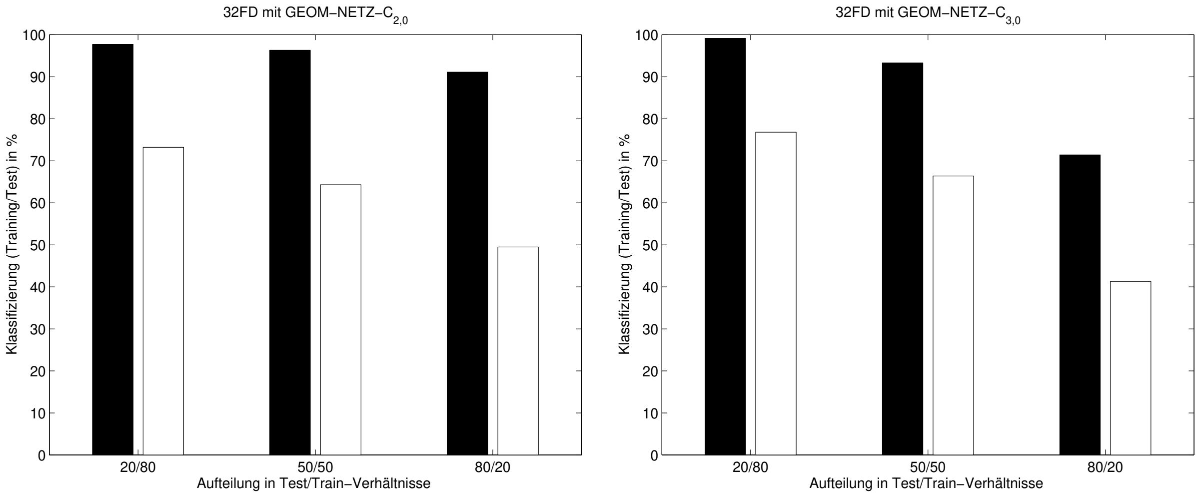


Abbildung 4.4: *Merkmal* **32FD** *trainiert mit den zwei Geometrischen Netze $C_{2,0}$ (links) und $C_{3,0}$ (rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

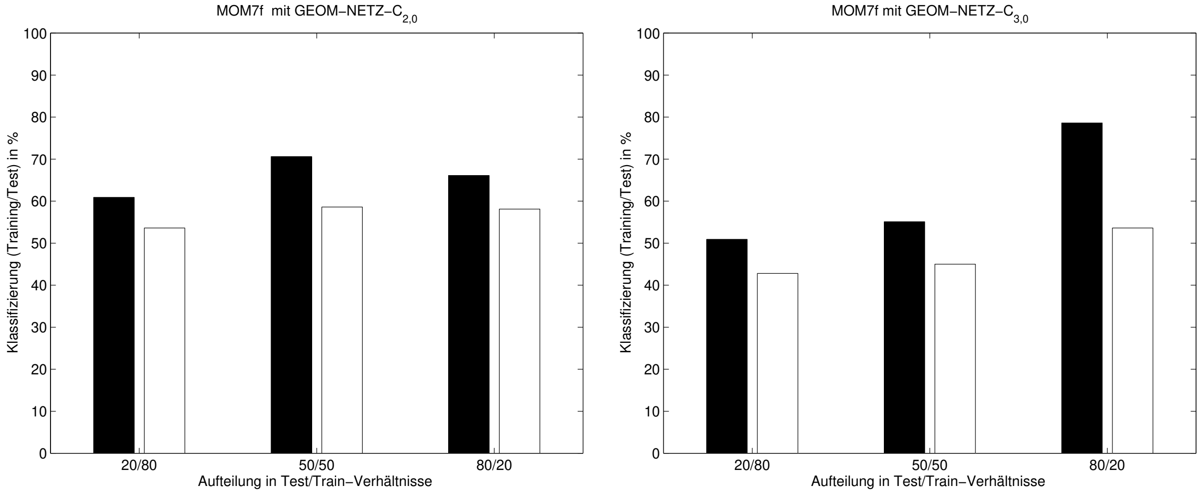


Abbildung 4.5: *Merkmal* **MOM7f** *trainiert mit den zwei Geometrischen Netze* $C_{2,0}$ *(links) und* $C_{3,0}$ *(rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

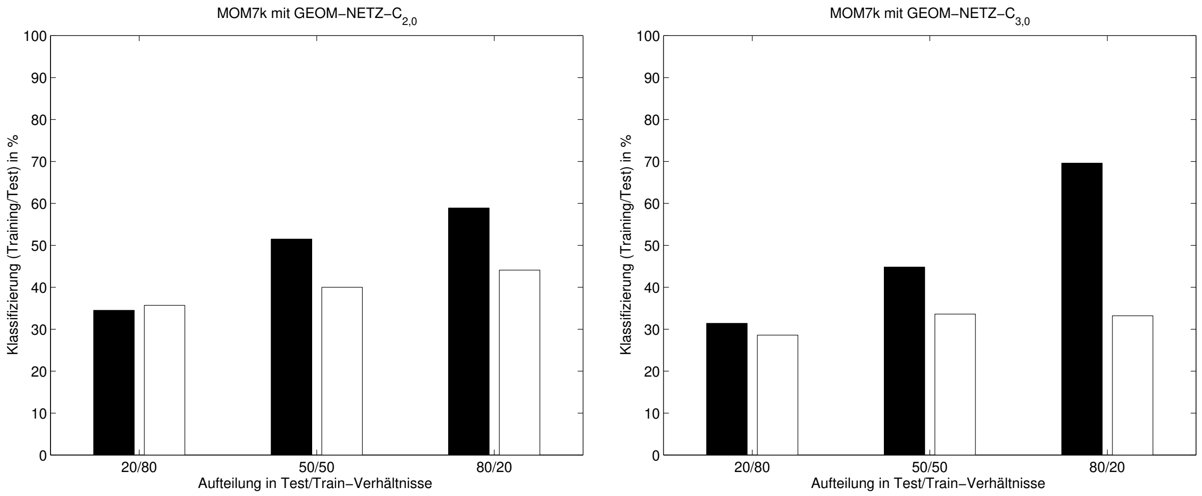


Abbildung 4.6: *Merkmal* **MOM7k** *trainiert mit den zwei Geometrischen Netze* $C_{2,0}$ *(links) und* $C_{3,0}$ *(rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

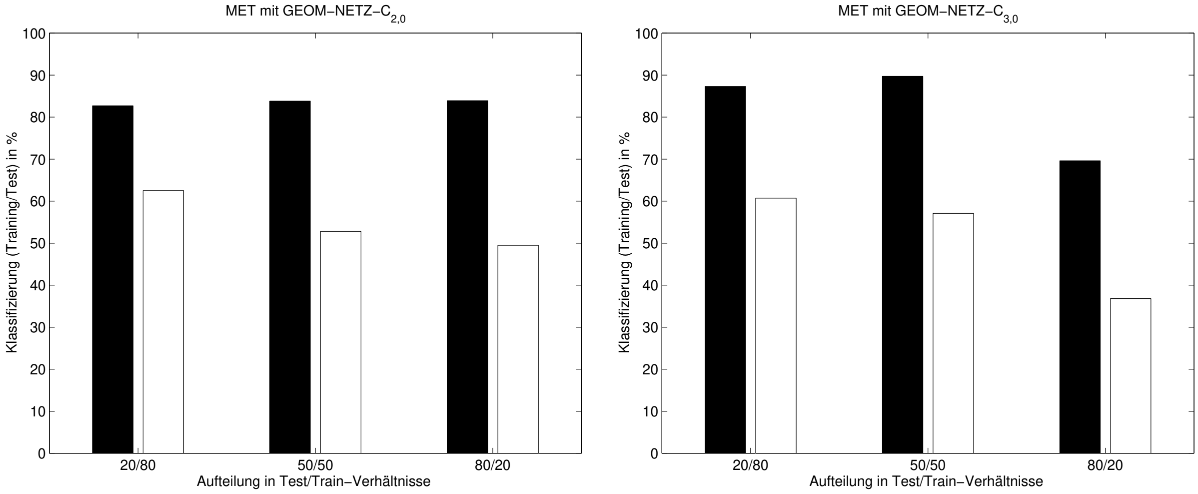


Abbildung 4.7: *Merkmal* **MET** *trainiert mit den zwei Geometrischen Netze $C_{2,0}$ (links) und $C_{3,0}$ (rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

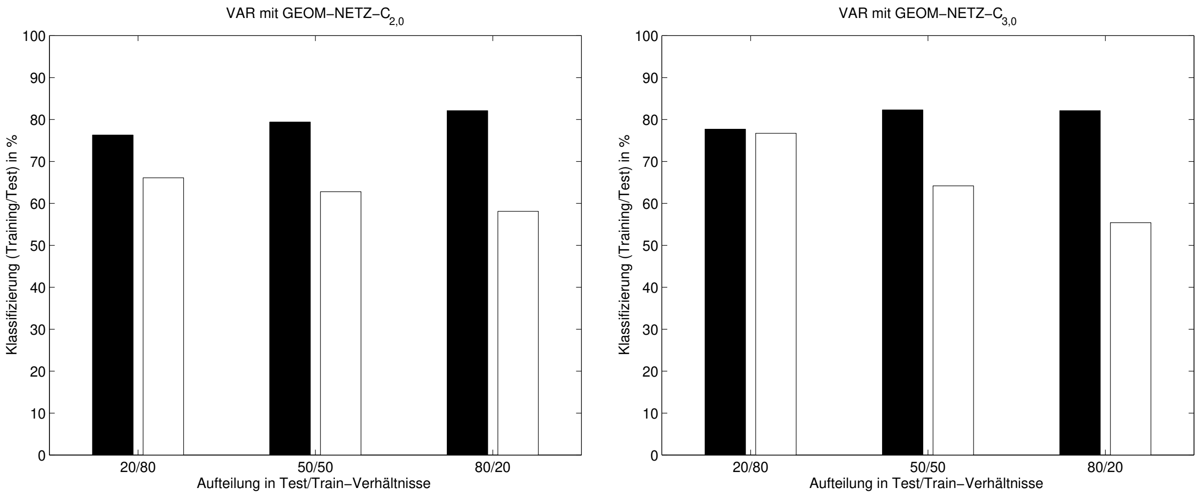


Abbildung 4.8: *Merkmal* **VAR** *trainiert mit den zwei Geometrischen Netze $C_{2,0}$ (links) und $C_{3,0}$ (rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

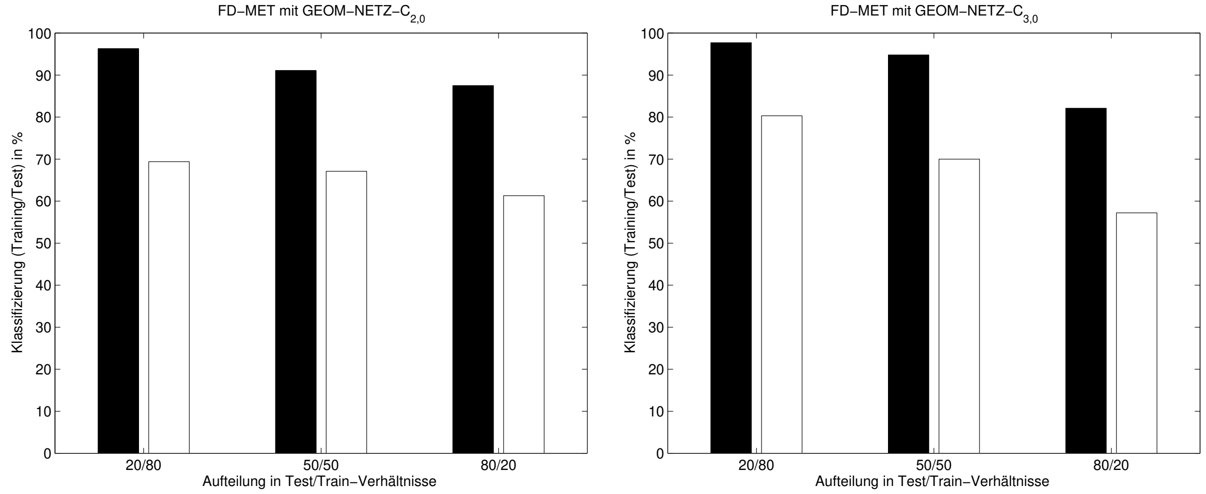


Abbildung 4.9: *Merkmal* **FD–MET** *trainiert mit den zwei Geometrischen Netze $C_{2,0}$ (links) und $C_{3,0}$ (rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

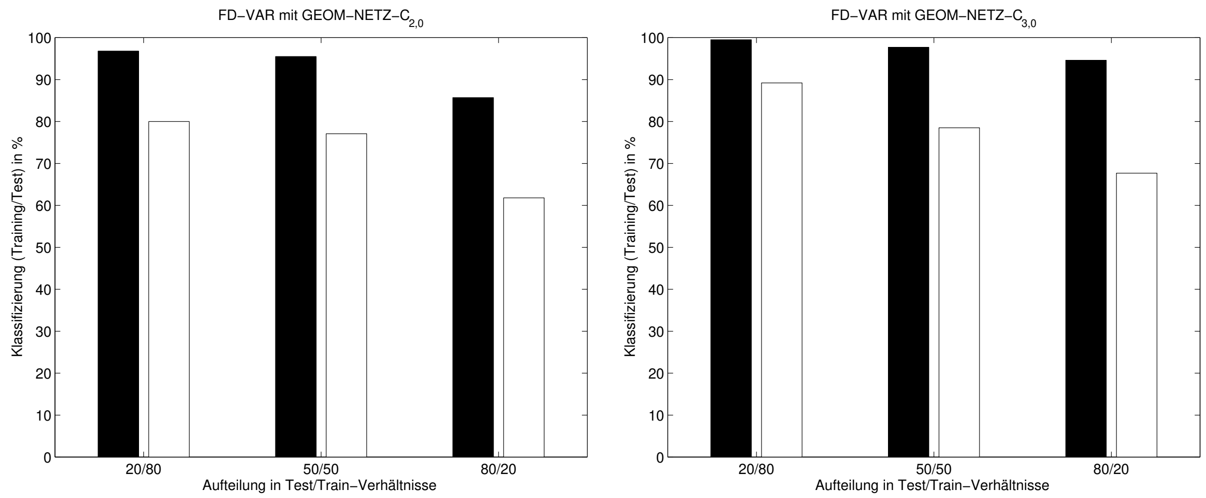


Abbildung 4.10: *Merkmal* **FD–VAR** *trainiert mit den zwei Geometrischen Netze $C_{2,0}$ (links) und $C_{3,0}$ (rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

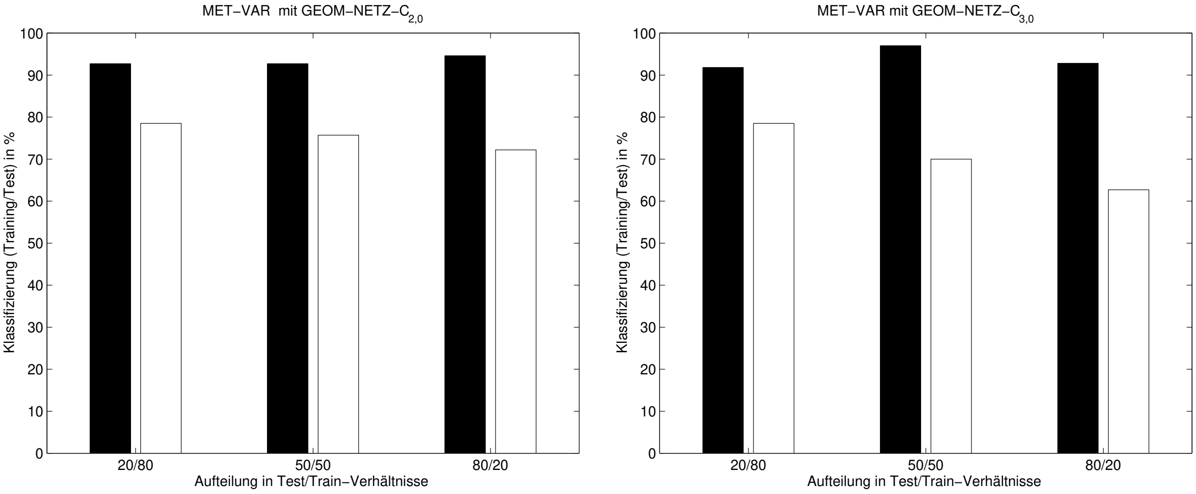


Abbildung 4.11: *Merkmal* **MET–VAR** *trainiert mit den zwei Geometrischen Netze $C_{2,0}$ (links) und $C_{3,0}$ (rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

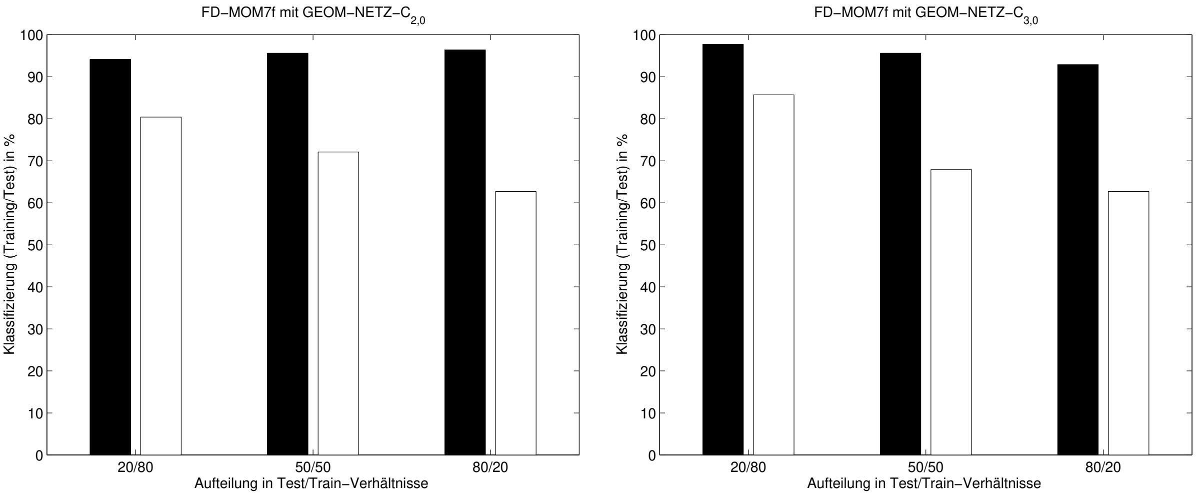


Abbildung 4.12: *Merkmal* **FD–MOM7f** *trainiert mit den zwei Geometrischen Netze $C_{2,0}$ (links) und $C_{3,0}$ (rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

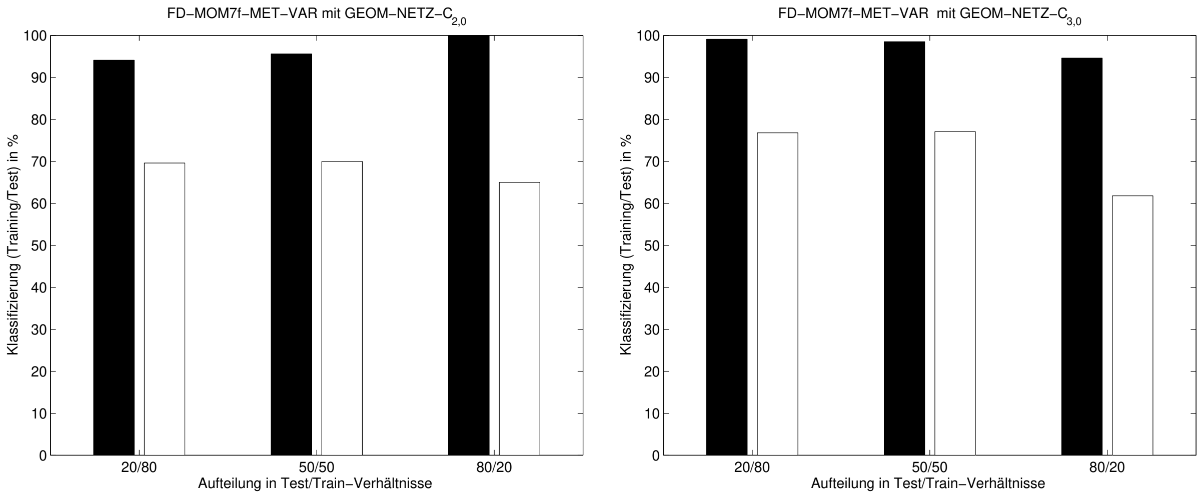


Abbildung 4.13: *Merkmal* **FD–MOM7f–MET–VAR** *trainiert mit den zwei Geometrischen Netze $C_{2,0}$ (links) und $C_{3,0}$ (rechts). Die Balken repräsentieren die Klassifizierung (Training/Test) in Prozent für 3 verschiedene Aufteilungen der Test/Train–Verhältnisses 20/80, 50/50, 80/20 (von links nach rechts).*

Die Ergebnisse für die Geometrischen Netze sind wie folgt zu beurteilen:

Für keinen Merkmalsvektor konnte eine fehlerfreie Klassifizierung der Testdaten erreicht werden. Sogar in den Versuchsreihen, wo das Training nahezu hundertprozentig erfolgreich war, blieb eine vollständig korrekte Erkennung der Testdaten aus. Bezüglich der verschiedenen Aufsplittungen der Gesamtdaten in Test– und Trainingsmenge ist eindeutig festzustellen, daß eine 20/80 Aufteilung durchgehend für alle Testreihen und alle Merkmale die deutlich beste Performance ergeben hat. Das bedeutet, daß ein hohes Datenvolumen im Training notwendig ist, um Generalisierung überhaupt erst zu ermöglichen. Lediglich 20% Trainingsdaten waren dazu stets völlig unausreichend. Im folgenden gehen wir daher genauer auf die erzielten Ergebnisse für das 20/80 Verhältnis ein.

Die Ergebnisse unterscheiden sich sehr stark für die verschiedenen Merkmalsgruppen. Es ist zu beobachten, daß die Klassifizierung der zusammengesetzten, und somit höher dimensionalen Merkmalsvektoren und der an sich schon hochdimensionalen Vektoren des Merkmals **32FD**, durch das Geometrische Netz $C_{3,0}$ besser generalisert worden ist. Für die geringer dimensionalen Merkmalsvektoren der Momente oder der **4FD** sind die $C_{2,0}$ Netze in ihrer Leistung (auch wenn nur geringfügig) besser.

Am besten war die Klassifizierungsrate für die gemischten Vektoren **FD–VAR** mit im Mittel 89,2% , **FD–MOM7f** mit im Mittel 85,7% und **FD–MET** mit im Mittel 80,3%.

Am schlechtesten schnitten die auf den Momenten basierenden Merkmale ab (beim $C_{2,0}$ Netz sogar mit nur 35,7%).

Die Fourierdeskriptoren sind mit ihren bis zu 78% Erkennungsrate im mittleren Bereich der Generalisierungsleistung.

In der Regel betrug der Abfall der Klassifizierungsrate auf der Testmenge gegenüber der Trainingsmenge ca. 20%. Größtenteils war ein perfektes Lernen der Trainingsmenge nicht möglich.

4.2 Klassifikation mittels DCS-Netze

Für das Training der DCS–Netze haben wir uns der selben Merkmalsvektoren bedient, wie sie im Abschnitt 4.1 aufgelistet sind. Damit unterscheiden wir zwischen 4–, 8–, 16–, und 32–dimensionalen Eingabevektoren bei einer unär kodierten 8–dimensionalen Ausgabe. Die Aufteilung des zur Verfügung stehenden Datensatzes in drei Varianten (20/80, 50/50 und 80/20 Test/Training) wurde auch bei den DCS–Netzen beibehalten und als Klassifizierungsgrundlage für die verschiedenen Versuche verwendet. Diese Aufsplittung wurde ebenfalls mit einem Zufallsgenerator realisiert, damit man eine dynamische und damit willkürliche Aufspaltung der zu erkennenden Muster erhält. Somit wurde insbesondere in der Klasse 2 herrschende Mannigfaltigkeit der Datenmuster auf ihre Eignung als

Testobjekt dem Zufall überlassen. Bei einer statischen, fest vorgenommenen Erzeugung der Testmenge durch das Abspalten der z.B. letzten 20 Repräsentanten der Klasse, hätte man nur einen einseitigen und damit unausreichenden Lerneffekt erzielt.

Im Gegensatz zu den Geometrischen Netzen wurden jeweils zehn Durchläufe pro Merkmal und pro prozentuale Aufteilung durchgeführt. Der daraus errechnete Mittelwert wurde in den hierauf folgenden Diagrammen graphisch dargestellt. Damit ergibt sich für das Training der DCS–Netze ein Gesamtvolumen von 390 Experimenten.

Die Parametereinstellung für das Training und Testen der DCS–Netze wurde wie folgt vorgenommen:

- INPUT DIMENSION: 4, 8, 16 oder 32
- OUTPUT DIMENSION: 8
- REZEPTIVE FIELDTYPE: Sphere
- APPROXIMATION TYPE: zero order
- EIGENVALUE PRUNNING: fraction of largest
- INTERPOLATION TYPE: normalization
- ACTIVE SET: bmu_and_direct
- ACTIVATION FUNCTION: exp_activation
- ERROR FUNCTION: squared_error
- OFFLINE LEARNING FUNCTION: superv_sdcs

Auf die detaillierte Bedeutung der einzustellenden Parameter für das Training der DCS–Netze wird hier nicht näher eingegangen, sondern auf die Arbeiten von [AM98] und [Trö97] verwiesen.

Es ist leicht zu erkennen, daß die Freiheitsgrade der Parametereinstellung bei den DCS–Netzen höher liegen als bei den Geometrischen Netzen.

Für eine visuelle Beurteilung der Testergebnisse, sortiert nach der Reihenfolge der Merkmale (siehe Liste) und analog zu den Geometrischen Netzen in drei Aufteilungen, betrachte man die folgenden Graphiken.

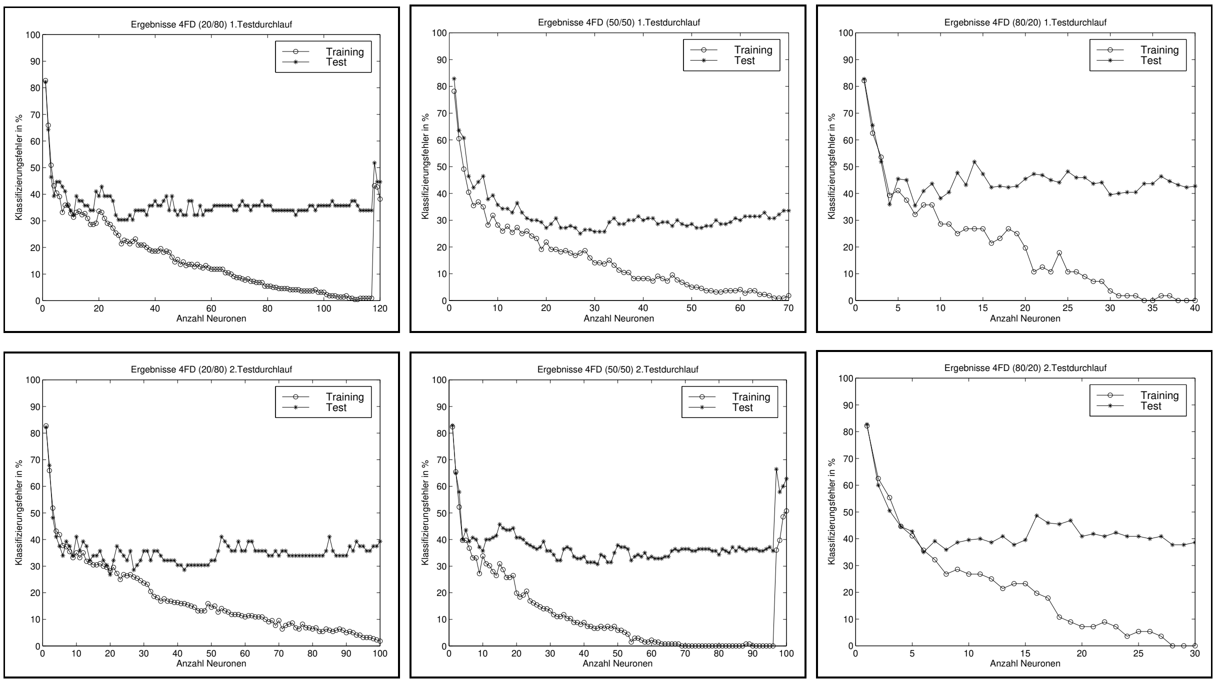


Abbildung 4.14: *Ergebnisse der DCS-Netze für das Merkmal* **4FD** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

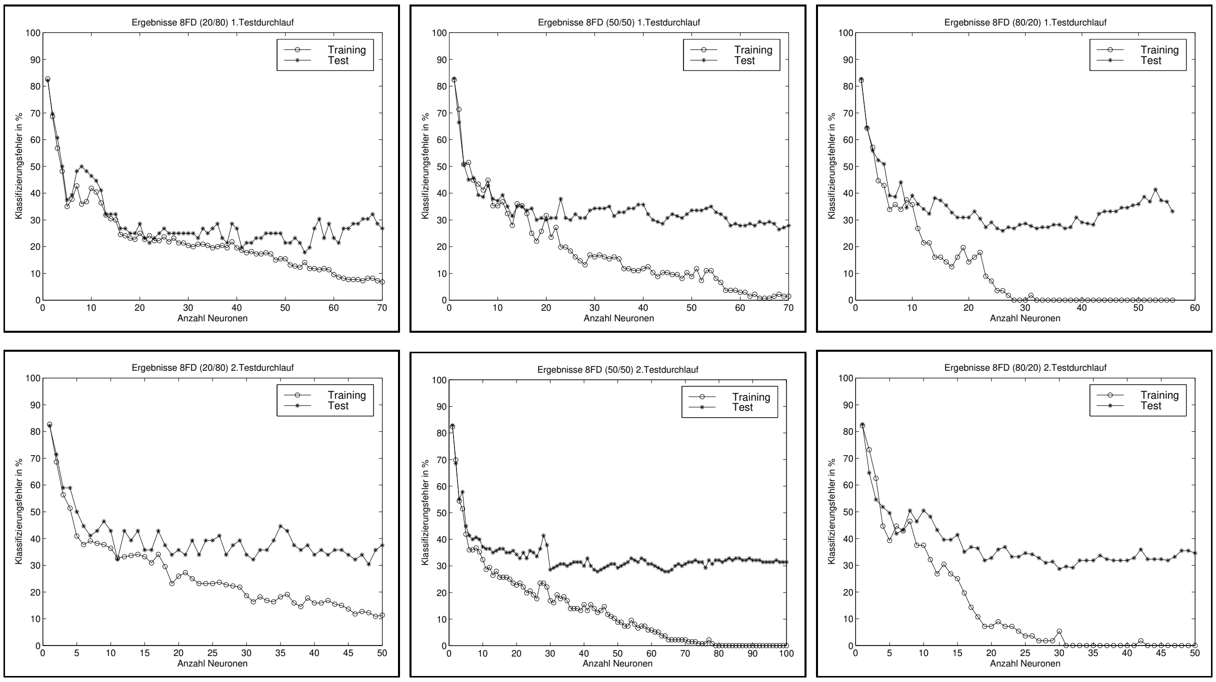


Abbildung 4.15: *Ergebnisse der DCS-Netze für das Merkmal* **8FD** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

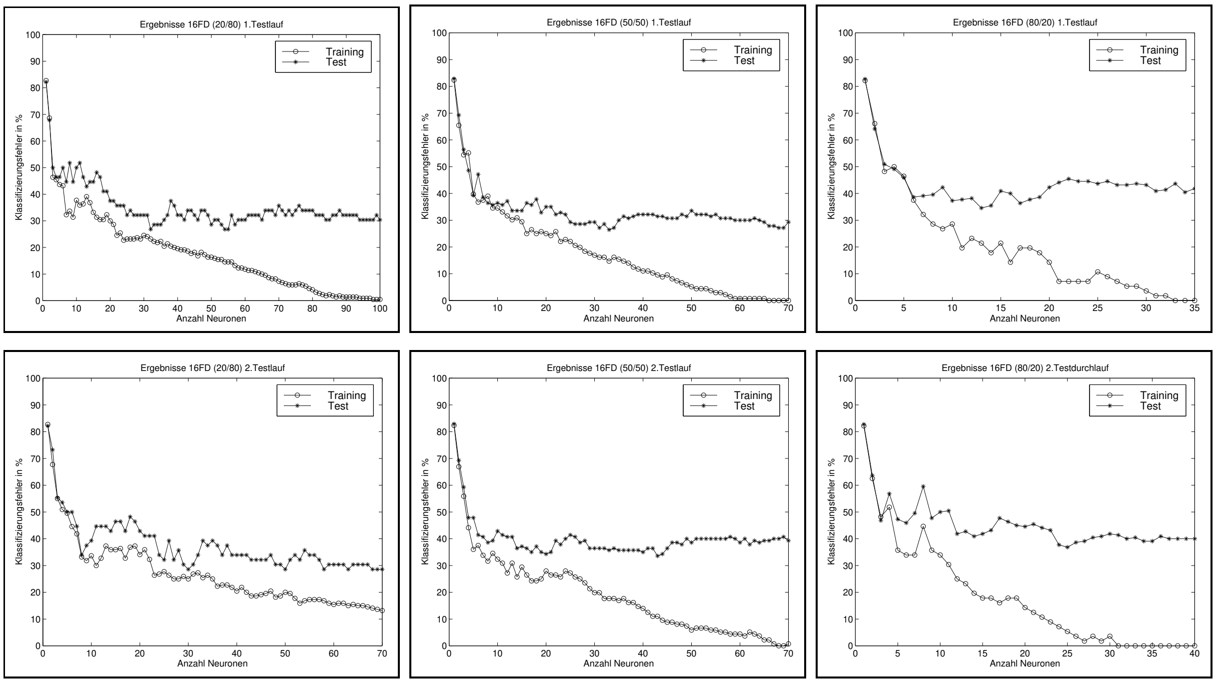


Abbildung 4.16: *Ergebnisse der DCS-Netze für das Merkmal* **16FD** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

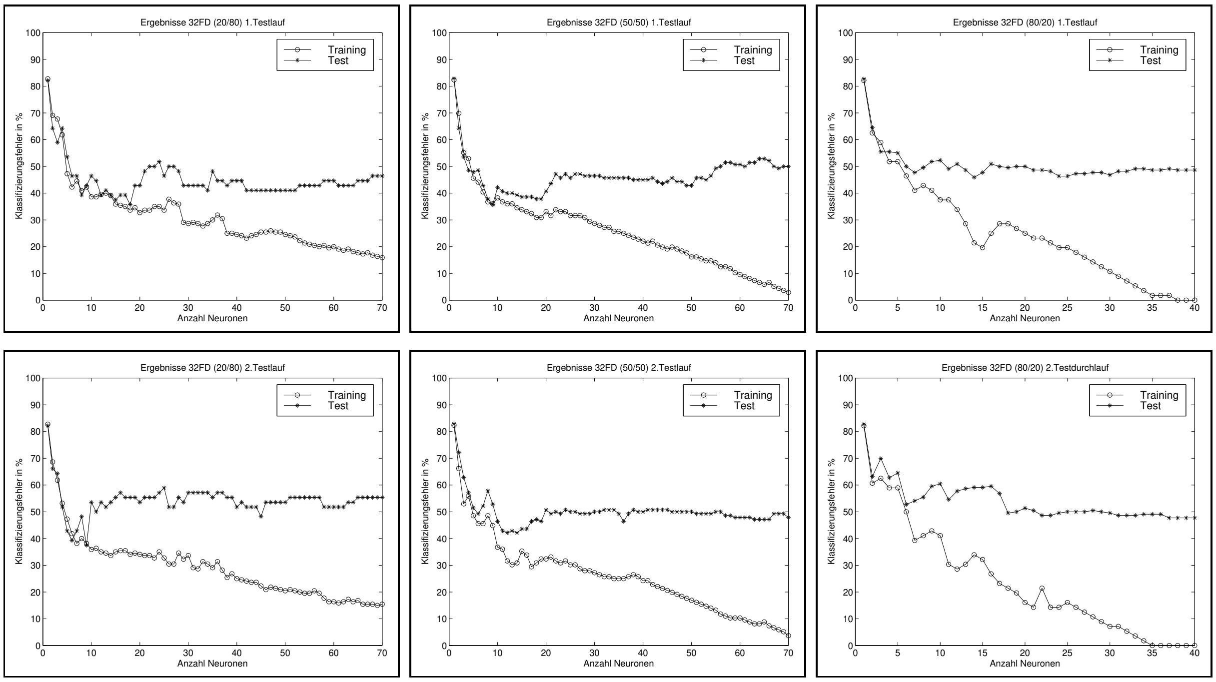


Abbildung 4.17: *Ergebnisse der DCS-Netze für das Merkmal* **32FD** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

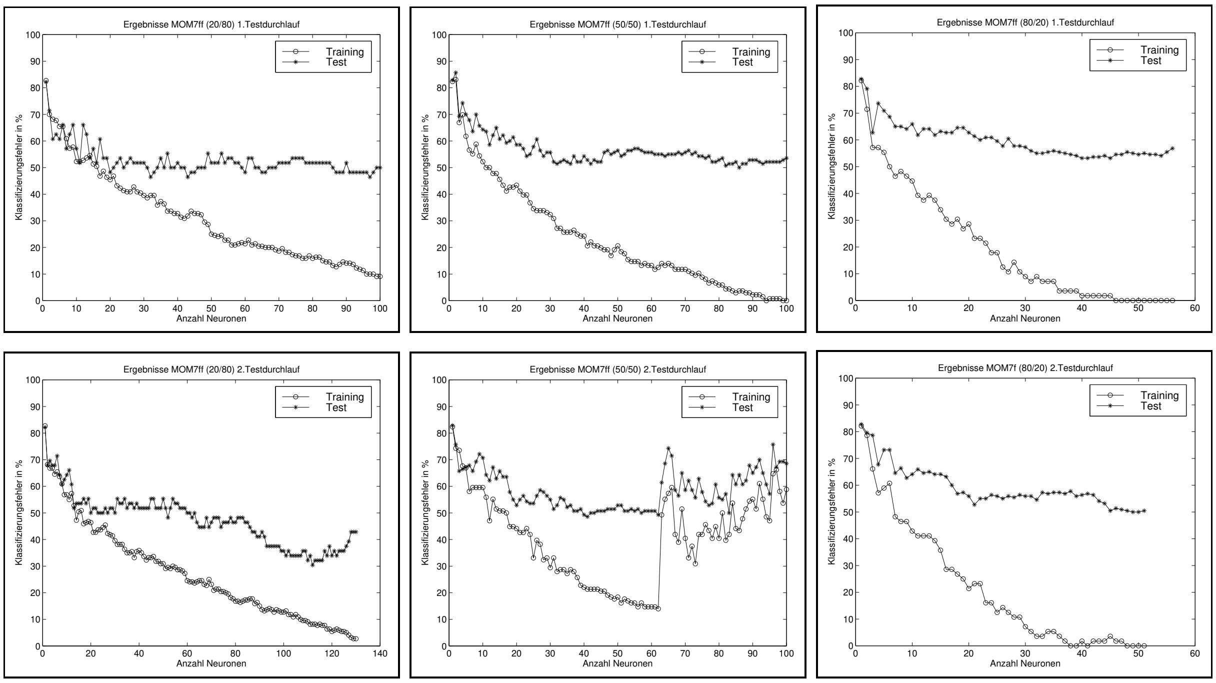


Abbildung 4.18: *Ergebnisse der DCS-Netze für das Merkmal* **MOM7f** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

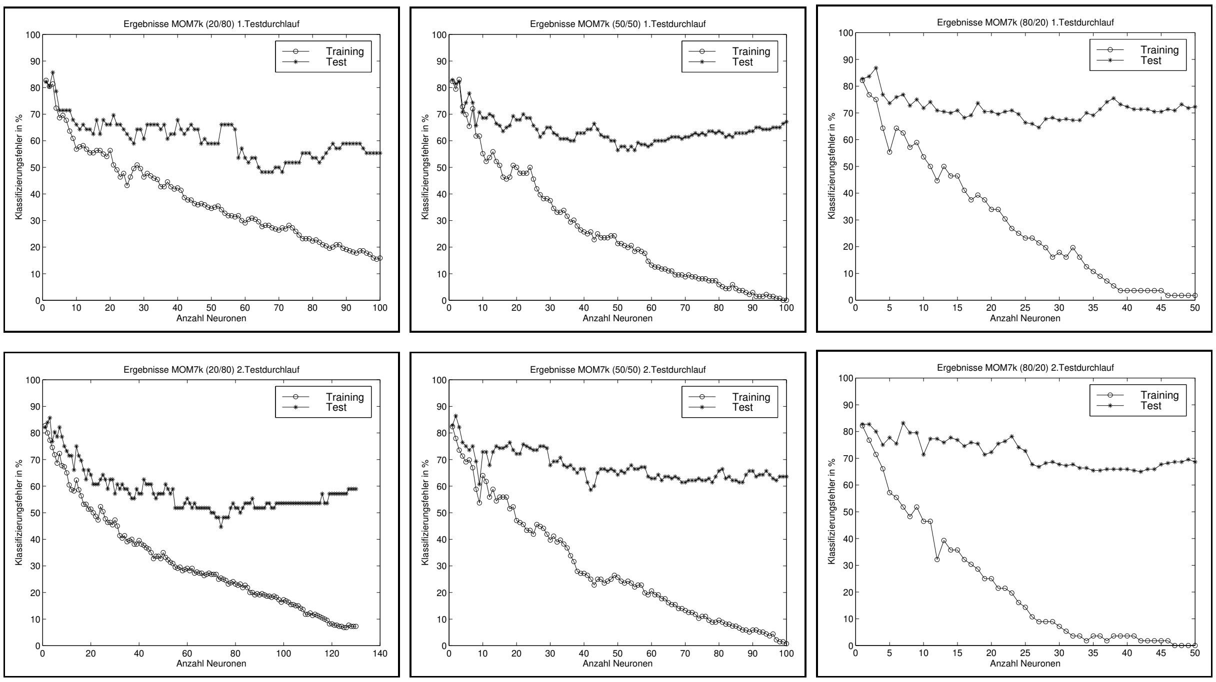


Abbildung 4.19: *Ergebnisse der DCS-Netze für das Merkmal* **MOM7k** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

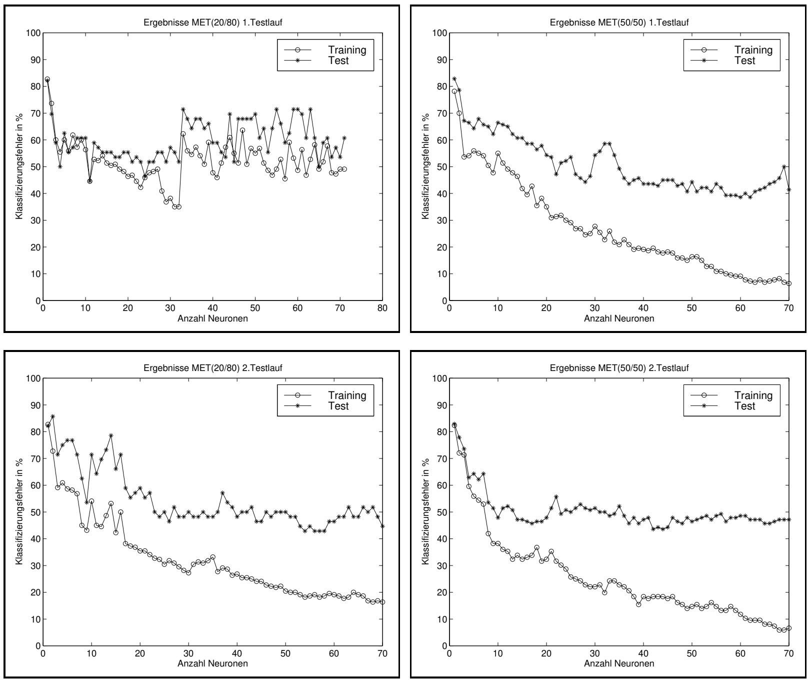


Abbildung 4.20: *Ergebnisse der DCS-Netze für das Merkmal* **MET** *für jeweils (20/80) und (50/50) Test/Train–Verteilung (2 Versuchsreihen).*

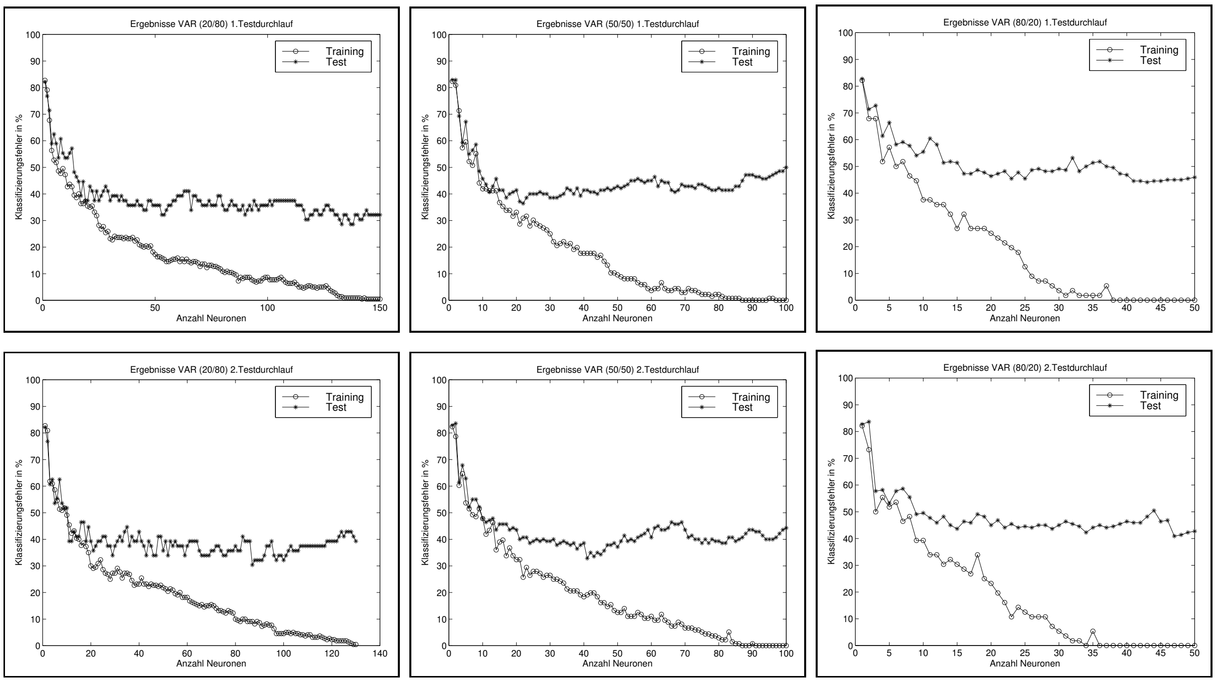


Abbildung 4.21: *Ergebnisse der DCS-Netze für das Merkmal* **VAR** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

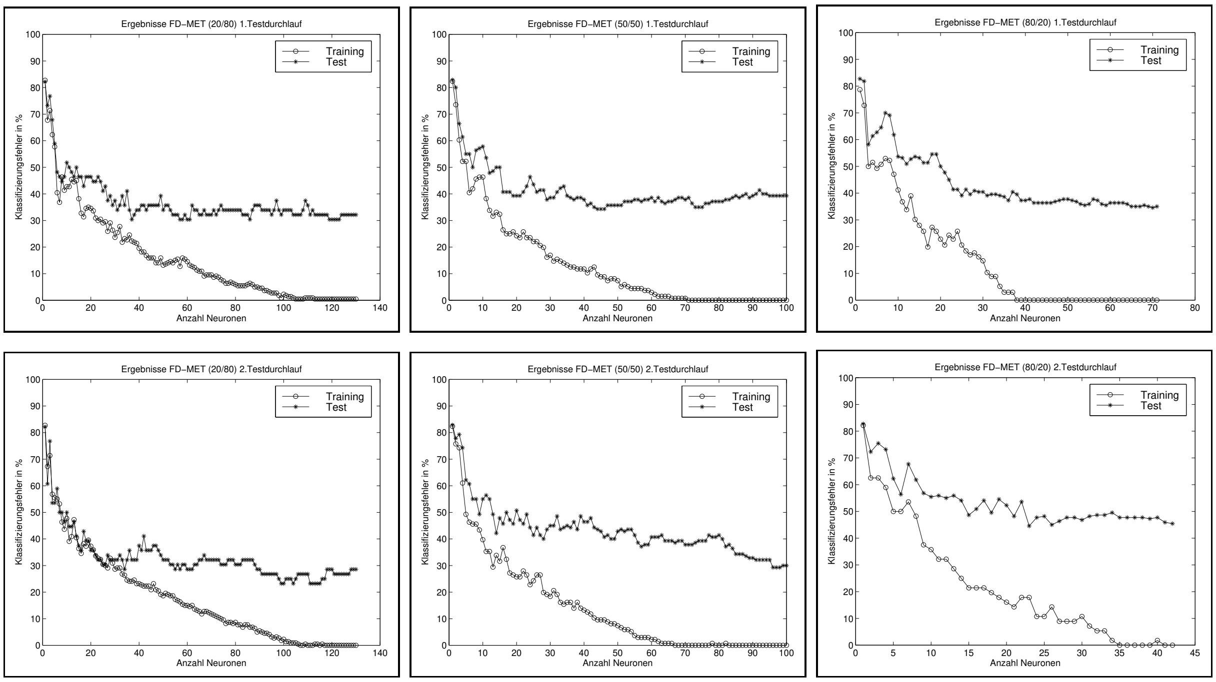


Abbildung 4.22: *Ergebnisse der DCS-Netze für das Merkmal* **FD–MET** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

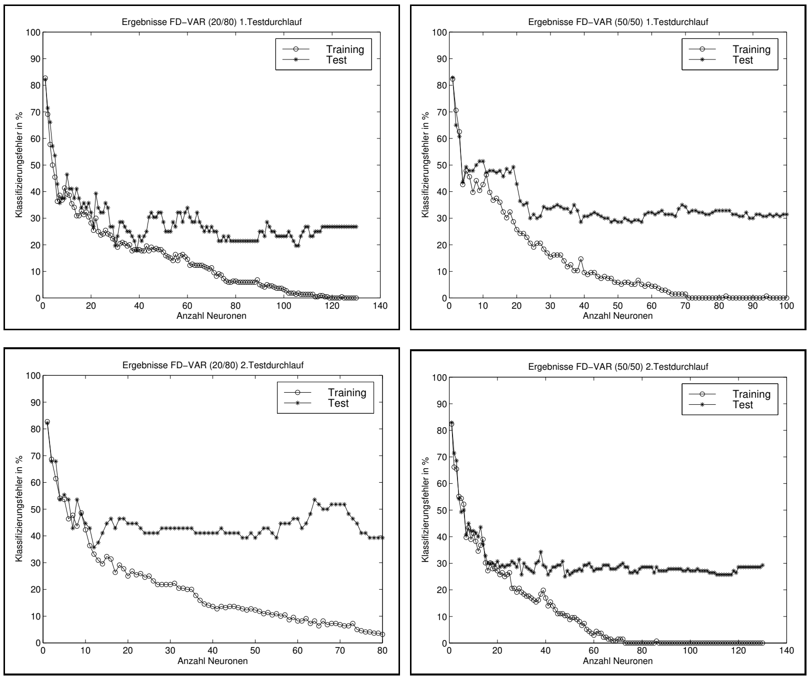


Abbildung 4.23: *Ergebnisse der DCS-Netze für das Merkmal* **FD–VAR** *für jeweils (20/80) und (50/50) Test/Train–Verteilung (2 Versuchsreihen).*

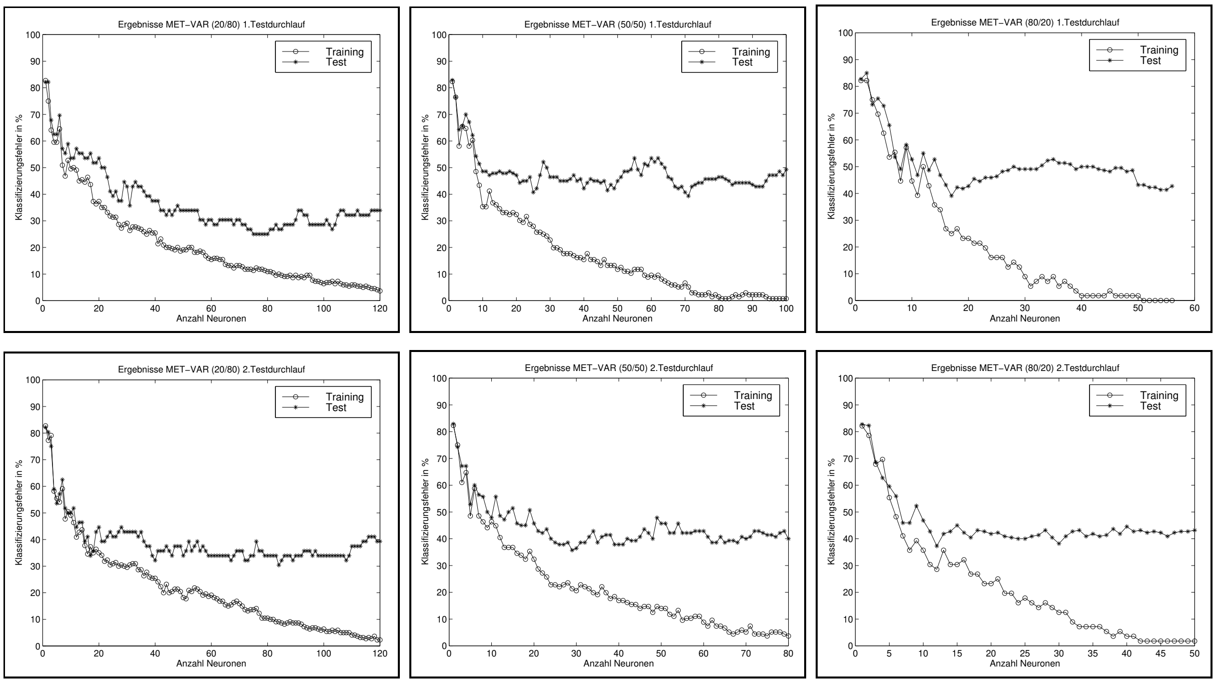


Abbildung 4.24: *Ergebnisse der DCS-Netze für das Merkmal* **MET–VAR** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

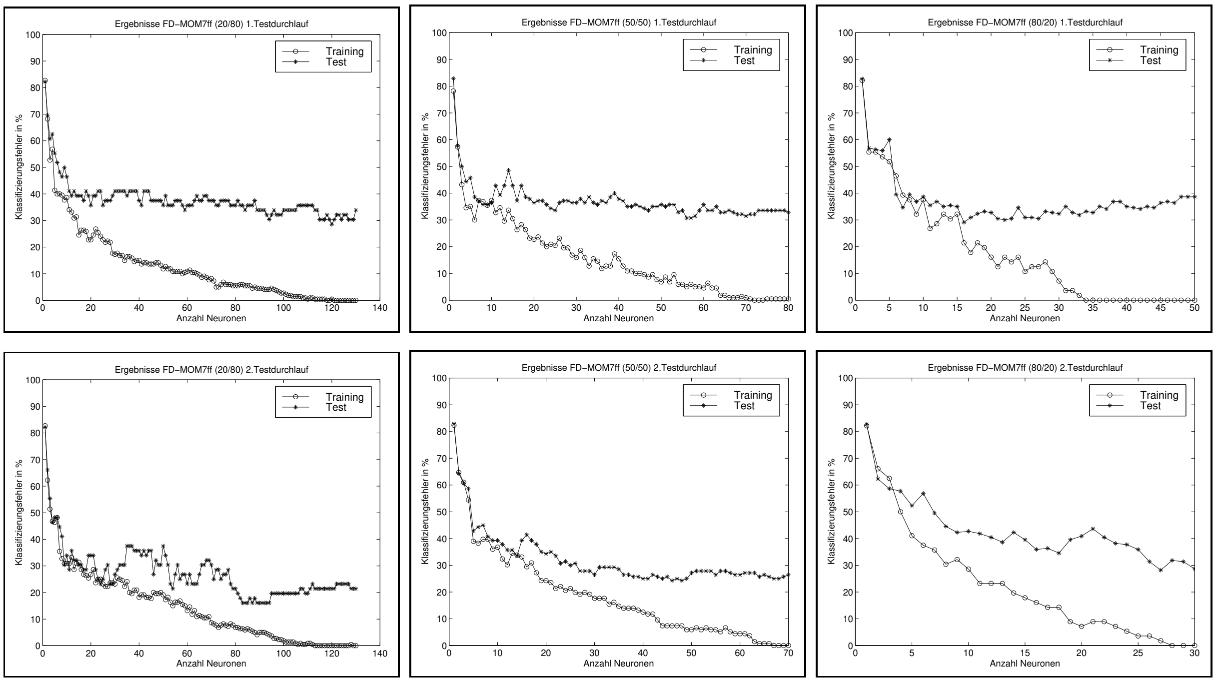


Abbildung 4.25: *Ergebnisse der DCS-Netze für das Merkmal* **FD–MOM7f** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

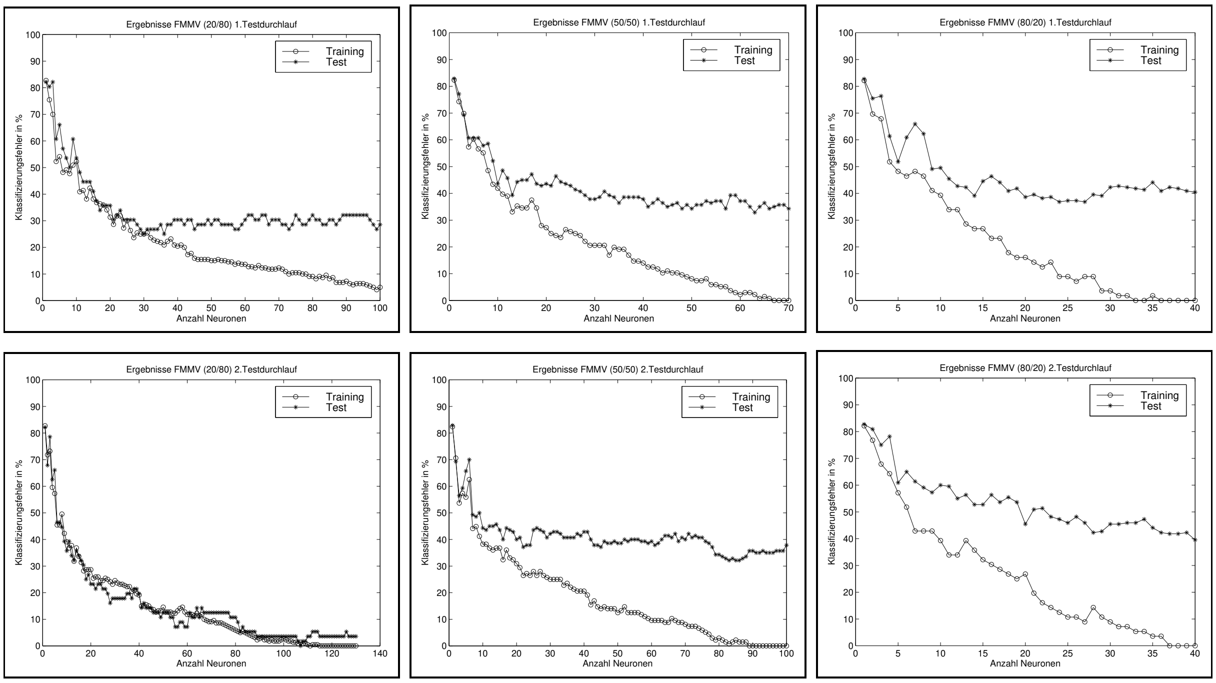


Abbildung 4.26: *Ergebnisse der DCS-Netze für das Merkmal* **FD–MOM7f–MET–VAR** *für jeweils (20/80), (50/50), (80/20) Test/Train–Verteilung (2 Versuchsreihen).*

Die Ergebnisse der DCS–Netze sind wie folgt zu beurteilen.

Eines der Merkmale wurde hundertprozentig generalisert. Es handelt sich dabei um den aus den Einzelmerkmalen **8FD**, **MOM7f**, **MET** und **VAR** zusammengesetzten 32–dimensionalen Merkmalsvektor. Auch andere Merkmale wurden hundertprozentig gelernt, ergaben aber an das Optimum nur heranreichende Testergebnisse (wenn überhaupt). Der aus den Fourierdeskriptoren und den Momenten zusammengesetzten Merkmalsvektor ist bei den DCS–Netzen die zweitbeste Kombination von Merkmalen. Die im mittleren Bereich der Dimensionalität liegenden (8– und 16–dimensionalen) Fourierdeskriptoren liefern eine zufriedenstellende Leistung der Klassifizierung (80%). Auch hier sind die zusammengesetzten, höher dimensionalen Vektoren geeigneter für die gestellte Aufgabe. Die auf den Momenten basierenden Vektoren besitzen wiederrum eine schlechte Generalisierung.

Die Diskrepanz zwischen Trainings– und Testlernrate beträgt bei den DCS–Netzen bis zu 40%. Die Aufteilung die drei Teilverhältnisse betreffend ist, mit der einzigen Ausnahme des hundertprozentig korrekt gelernten Merkmals, aufgrund der Natur des Netzes nicht signifikant, worauf wir in der Diskussion eingehen werden. Die Anzahl der Neuronen ist um ein Vielfaches höher als bei den Geometrischen Netzen. Sie beträgt 30 bis 100 Knoten. Es schließt sich nun ein Gesamtvergleich der erzielten Ergebnisse an.

4.3 Vergleich der Experimente

Die Beurteilung der Ergebnisse für die beiden Netzarchitekturen erfolgt nach den einzelnen Merkmalsgruppen gegliedert. Als generelle Beobachtung ist festzustellen, daß die Mehrzahl der Merkmale eine Generalisierung zwischen 70% und 80% erlaubt. Zwei Ausnahmen fallen beim Betrachten der Gesamtgraphik der Ergebnisse 4.27 dabei sofort ins Auge. Zum einen sind es die beiden schwach erlernten Momente auf der Kontur (**MOM7k**) und auf dem gefüllten Bild (**MOM7f**), und zum anderen der 32-dimensional zusammengesetzte Vektor, der von dem DCS–Netz zu hundert Prozent erlernt wurde. Nun zu den einzelnen Merkmalsgruppen.

Bei den *Fourierdeskriptoren* (die ersten vier Balkengruppen) erkennen wir eine nahezu gleich starke Lernleistung der drei Netze $C_{2,0}$, $C_{3,0}$ und DCS–Netz. Die Quote liegt dabei bei allen in den mittleren Wertebereichen und stellt ein zufriedenstellendes Ergebnis dar.

Bei den *Momenten–Merkmalen*, die auf den sieben invarianten Moment–Funktionen nach [Hu62] beruhen, haben die DCS–Netze eine gegenüber den Geometrischen Netzen $C_{2,0}$ und $C_{3,0}$ eine um 30% bessere Erkennung der Testdaten erzielt. Nichtsdestotrotz ist bei beiden die Generalisierung auf niedrigem Niveau.

Der auf den *Momenten zweiter Ordnung*, der *Orientierung* und der *Exzentrizität* basierende **MET**–Vektor, der invariante Eigenschaften besitzt, ist schlechter erkannt worden, als der Vektor **VAR**, der sich auf Merkmale stützt, die stark von der Form und Größe

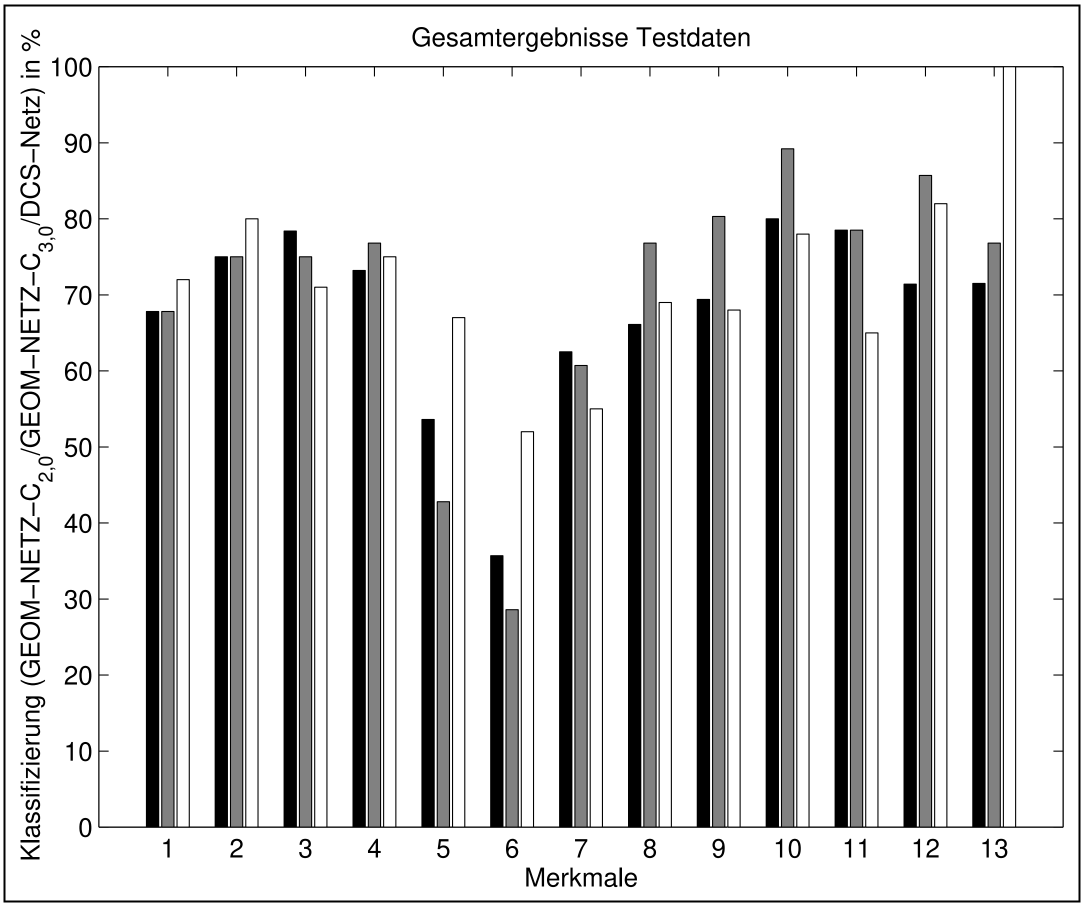


Abbildung 4.27: *Gesamtergebnisse der Geometrischen Netze $C_{2,0}$ $C_{3,0}$ und des DCS-Netzes (von links nach rechts) sortiert nach den einzelnen Merkmalsvektoren 1–13 :* **4FD**, **8FD**, **16FD**, **32FD**, **MOM7f**, **MOM7k**, **MET**, **VAR**, **FD–MET**, **FD–VAR**, **MET–VAR**, **FD–MOM7f** *und der 100%-ig gelernte* **FD–MOM7f–MET–VAR** *(links außen).*

bzw. vom Umfang des Objekts abhängen. An den Auswertungen des **VAR**– Vektors ist die gute, 80%–ige Erfolgsquote des $C_{3,0}$ zu bemerken. Dies setzt sich als Tendenz für die kommenden drei Merkmalstypen fort.

Bei den ersten sieben Vektoren handelt es sich um *reine* Merkmale. Die genannten Vektoren beinhalten also lediglich einen Typus eines Merkmals. Daher können wir uns die Schlußfolgerung erlauben, daß Momenten–basierte Erkennung durch die DCS–Netze besser generalisiert wird.

Die *kombinierten* 16–dimensionalen Merkmalsvektoren, die zur Hälfte aus Fourierdeskriptoren bestehen, werden alle durch das Geometrische Netz $C_{3,0}$ am besten erkannt. Hierbei ragt die Leistung beim zehnten Merkmalsvektor, bestehend aus acht *Fourierdeskriptoren* und acht *nicht invarianten* Merkmalen, sogar an die 90%–Marke heran.

Der 13. Vektor (und man sollte noch meinen, daß diese Zahl kein Glück bringt!) ist für eine fehlerfreie, optimale Generalisierungsfähigkeit des DCS-Netzes als einziger ausreichend geeignet. Er vereint in sich sowohl invariante, als auch von der Ausdehnung des Objekts abgeleitete Kriterien, und ist mit 32 Dimensionen die umfangreichste Merkmalsakkumulation in der gesamten Versuchsreihe. Daher können wir sagen, daß eine Diversität der Merkmalsextraktion bei den DCS–Netzen zu einem besseren Klassifikationsergebnis führt. Bei den Geometrischen Netzen ist die Kombination von invarianten und morphologie–abhängigen Merkmalen am Erfolg versprechendsten.

Hiermit schließen wir die Methodologie der vergleichenden Studien und wenden uns abschließend einer methodenkritischen Diskussion der hier erzielten Ergebnisse hinsichtlich ihrer Anwendungseignung zu.

Kapitel 5

Diskussion und Ausblick

In dieser Diplomarbeit wurde ein merkmalsbasiertes Klassifikationsverfahren für antarktische Algen mittels Neuronaler Netze vorgestellt und auf seine Praktikabilität hin bewertet und untersucht. Der Schwerpunkt der Arbeit lag dabei in der Konstruktion eines rotations–, translations– und skalierungsinvarianten Merkmalsvektors. Dies war aus dem Grunde von zentraler Bedeutung, weil sich durch die genannten invarianten Eigenschaften eine Reduktion der Mannigfaltigkeit der Beschreibung der einzelnen Algen ergab. Der Verzicht auf die Invarianzen hätte bei jeder neuen Alge eine neue Berechnung des Merkmalsvektors bedeutet. Um diese Merkmale extrahieren zu können, waren eine Reihe von Vorverarbeitungsschritten notwendig, die letztendlich dazu beigetragen haben, daß die auf diese Weise gewonnenen Daten als Eingabe für neuronale Klassifikatoren dienen konnten. Nebst Fläche und Rundheit haben wir uns zum einen den Fourierdeskriptoren [PF77], [Bur79], [Arb90], und zum anderen den Momenten nach [Hu62] gewidmet. Die auf Grundlage dieser mathematischen Methoden errechneten Merkmale wurden zu einfachen und kombinierten Merkmalsvektoren zusammengetragen. Diese wurden anschließend als Trainingsbasis für zwei unterschiedliche Netztypen verwendet. Bei den Neuronalen Netzen handelt es sich um das Geometrische Netz [BCB97] und um das DCS–Netz [BS94]. Die Herausforderung bestand darin, diese Netze hinsichtlich ihrer Klassifikationsperformance zu untersuchen und miteinander zu vergleichen.

Das vom Institut für Polarökologie gestellte Datenmaterial wurde zu einer sogar im Internet präsenten Datenbank aufgebaut und diente als Trainings– und Testbasis für unsere Versuche. Die qualitative und quantitative Auswertung der Gesamtdaten hinsichtlich ihrer Eignung als Trainings– oder Testmuster führte zu einer Reduktion der ursprünglich 2000 Algenbilder auf lediglich 300 Muster, die die gewünschten morphologischen Kriterien erfüllten. Mit diesem selektierten Bestand wurden eine Reihe von experimentellen Studien nach einzelnen Merkmalstypen durchgeführt. Die Ergebnisse dieser Versuche sind statistisch und in Form von Graphiken resümiert worden. Ihre Bedeutung und Interpretation, sowie eine mögliche Begründung dieser Resultate sollen Gegenstand dieses Kapitels sein.

Die methodenkritische Interpretation der Ergebnisse wird in zwei Kategorien aufgeteilt.

Zum einen handelt es sich um die Gestaltung der Merkmalsvektoren und ihre Eignung als Klassifikationsgrundlage, zum anderen werden die beiden Netzarchitekturen miteinander verglichen.

5.1 Modellierung der Merkmalsgestaltung

Unter Klassifikationsverfahren versteht man Verfahren, die dazu verwendet werden, Objekte verschiedenen Klassen zuzuordnen. Dabei soll im Idealfall die tatsächliche Klassenzugehörigkeit der einzelnen Objekte unbekannt sein, so daß eine Klassifizierung allein aufgrund der Merkmale der Objekte vorgenommen werden kann. Um diese Leistung erst zu ermöglichen, mußten im Vorfeld die von uns verwendeten Netze angelernt werden. Um die merkmalsbasierte Erkennung durch die Netze zu erzielen, war es notwendig, Merkmale aus Bildern zu extrahieren, die invariante Eigenschaften besitzen. Von jedem Algenmuster stand uns eine einzige Aufnahme zur Vefügung, in Form einer Strichzeichnung oder eines Elektronenmikroskopbildes. Da für jede der acht Klassen eine Vielfalt von unterschiedlichen Algenvariationen in Frage kam, lag es nahe, sich auch auf Merkmale zu konzentrieren, die bei einer Veränderung des Skalierungsverhälnisses oder einer Rotation konstant blieben. Damit sollte eine Reduktion der Mannigfaltigkeit der Beschreibung der einzelnen Bilder erreicht werden, die nur durch invariante Merkmale abgedeckt werden kann. Aus dieser Motivation heraus haben wir ein Kontingent an Merkmalstypen zusammengetragen, die, wie sich durch die Versuche herausstellte, fast immer ausreichen, um ein Klassifikationskriterium zu erfüllen. Bei den invarianten Merkmalen handelt es sich um die in der Dissertation von [Arb90] vorgestellten Fourierdeskriptoren und um die sieben invarianten Momente nach [Hu62]. Wir haben feststellen können, daß die Momente allein betrachtet im Schnitt ein viel schlechteres Ergebnis liefern als die Fourierdeskriptoren, eine Kombination der beiden aber ein besseres Ergebnis des $C_{3,0}$–Netzes geliefert hat, als die einzelnen Merkmale singulär betrachtet. Die Ergebnisse waren in den Fällen aber am befriedigendsten, wo wir eine Mischung aus den beiden (variant/invariant) betrachtet haben. Dies könnte ein Hinweis sein, nicht gänzlich auf Merkmale wie Rundheit oder Exzentrizität zu verzichten, da sie *typisch* für viele Algenklassen sind (siehe hierfür die Abbildungen der einzelnen Klassen im Methodikteil), und somit auch eine statistische Abdeckung der Vorkommen vieler Arten repräsentieren. Bei der automatisierten qualitativen und quantitativen Auswertung von Algenbildern (z.B. für die Bestimmung des Biovolumens) ist es also ratsam, die Grundstruktur dieser Morphologien nicht aus den Augen zu lassen und die Fourierdeskriptoren oder die Momente als ergänzende Information über die Kontur hinzuzuziehen.

In dieser Arbeit hat sich die Merkmalsgestaltung als schwierigste Aufgabe herausgestellt. Die Vorverarbeitung unterliegt strengen mathematischen Vorgaben, denen man sich nicht entziehen kann, derer man sich aber auch bedient um eine Quintessenz aus den Rohdaten zu gewinnen. Die Beantwortung der Frage, welche Merkmale relevant sein können und auf welche man auch aus Motivation der Datenreduktion verzichten kann, ist uns nicht eindeutig und absolut gelungen. Die Variationsmöglichkeiten der beschriebenen Merkmalsvektoren, steigen exponentiell mit jedem neu hinzugefügten Merkmal.

5.2 Geometrische Netze versus DCS-Netze

Wie bereits erwähnt und kurz begründet war die Wahl der Netzwerkarchitektur in der Mehrzahl der getesteten Merkmalsvektoren nicht relevant. Die beiden Netzwerkarchitekturen waren dabei sehr verschieden. Das DCS–Netz basiert auf dem RBF–Modell und erweitert dieses um den dynamischen Verbindungsaufbau zwischen den Neuronen verbunden mit der Berücksichtigung lokaler Nachbarschaften. Das Geometrische Netz erweitert das herkömmliche MLP für Clifford–Algebren. Trotz dieser zweier gänzlich unterschiedlichen Architekturen lagen die Generalisierungsleistungen, bis auf eine Ausnahme, meist im Bereich zwischen 70% und 80%. Berücksichtigt man, daß das DCS–Netz aufgrund seiner Natur stets in der Lage ist (mit einer genügend großen Anzahl von Neuronen) die Trainingsdaten fehlerfrei zu klassifizieren, so bedeutet dies, daß mit den erzielten Werten durchaus konkrete obere Schranken gefunden wurden. Die Unterschiede zwischen den einzelnen Geometrischen Netzen bezüglich gemischter Vektoren (wobei das 8–dimensionale stets im Vorteil gegenüber dem 4–dimensionalen Netz war) erklären sich dadurch, daß bei ersterem die 16–dimensionalen Eingaben genau in ihre semantischen Einheiten (in je ein Neuron) zerlegt wurden. Dies führte sogar zu besserer Generalisierung als beim DCS–Netz, trotz der im Gegenzug nicht perfekten Trainingsquoten.

Das Resultat der hunderprozentig korrekten Generalisierung des DCS–Netzes liefert den Beweis, daß eine genügend große Anzahl von verschiedenen Merkmalen letztendlich ausreichend ist, bestätigt also unser methodisches Vorgehen insgesamt. Mit der dabei benötigten Anzahl von 100 Knoten wird aber auch die Komplexität unseres realen Klassifikationsproblems vollständig deutlich. Aus diesem Grunde war ein Verifizieren, ob dies prinzipiell mit Geometrischen Netzen ebenfalls möglich wäre, in Anbetracht der zu erwartenden Knotenzahl nicht realisierbar. Hier zeichnet sich ein Vorteil der dynamischen Architektur des DCS–Netzes und seiner Lokalität ab, die es erlauben auch solche Komplexitäten zu handhaben.

Um zu zeigen, daß die Klassifikationsaufgabe aufgrund des Datenmaterials nur an eine 80% Erkennung heranreichen konnte, nicht aber an den Klassifizierungsfähigkeiten der beiden Netze generell lag, haben wir im folgenden für je ein Merkmal pro Merkmalsgruppe (**16FD**, **MET** und **VAR**) noch eine beispielhafte Klassifikation mit drei idealen Musterklassen durchgeführt.

5.3 Klassifikation mit idealen Daten

Bei den drei idealen Klassen handelt es sich um *künstlich* hergestellte Daten, die aber in ihrer Beschaffenheit die Kriterien erfüllen, die wir mit den beiden Netzen differenzieren wollen. Die Klassen enthalten Muster in drei verschiedenen Grundformen (länglich, oval und rund) und decken mit einer Gesamtanzahl von 32 Bildbeispielen pro Klasse die rotatorischen, translatorischen und skalierungsabhängigen Variationsmöglichkeiten ab, die wir bei den realen Daten auch vorgefunden haben. Jeweils drei Repräsentanten aus diesen idealen Klassentypen sind in den Abbildungen 5.1, 5.2 und 5.3 zu sehen.

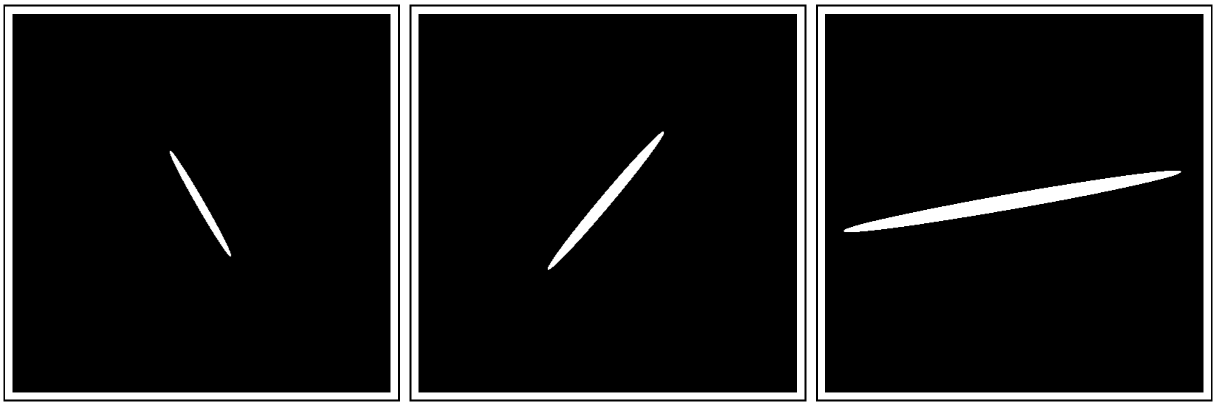


Abbildung 5.1: *Drei Repräsentanten der 1. Idealklasse. Hauptmerkmal ist die längliche Form, die skaliert und um verschiedene Rotationswinkel gedreht wurde.*

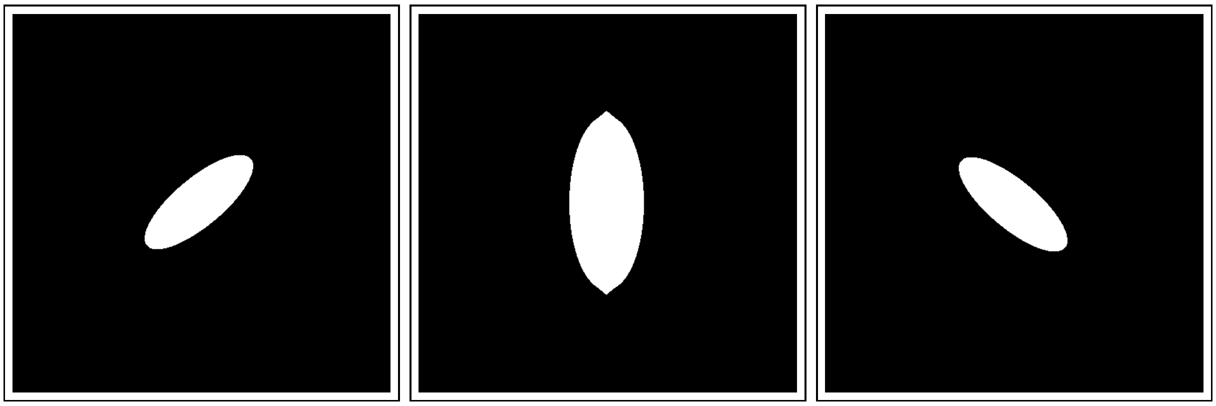


Abbildung 5.2: *Drei Repräsentanten der 2. Idealklasse. Hauptmerkmal ist die ovale Form, die skaliert und um verschiedene Rotationswinkel gedreht wurde.*

Mit derselben Methodik wie bei den anderen acht Klassen, wurden die Merkmalsvektoren aus den Bildern extrahiert und als Grundlage für das Training der beiden Netze genommen. Das Ergebnis war eine hundertprozentige Erkennung aller Merkmalstypen mit beiden Netzen. Das bedeutet sowohl eine sinnvolle Methodik der Merkmalsgewinnnung basierend auf den oben benannten Kriterien, als auch eine nahezu perfekte Eignung beider Netze für eine Klassifikationsaufgabe. Mehr noch, das Geometrische Netz hat die Trennung der drei Klassen mit Hilfe eines einzelnen Neurons in der versteckten Schicht (nach 1000

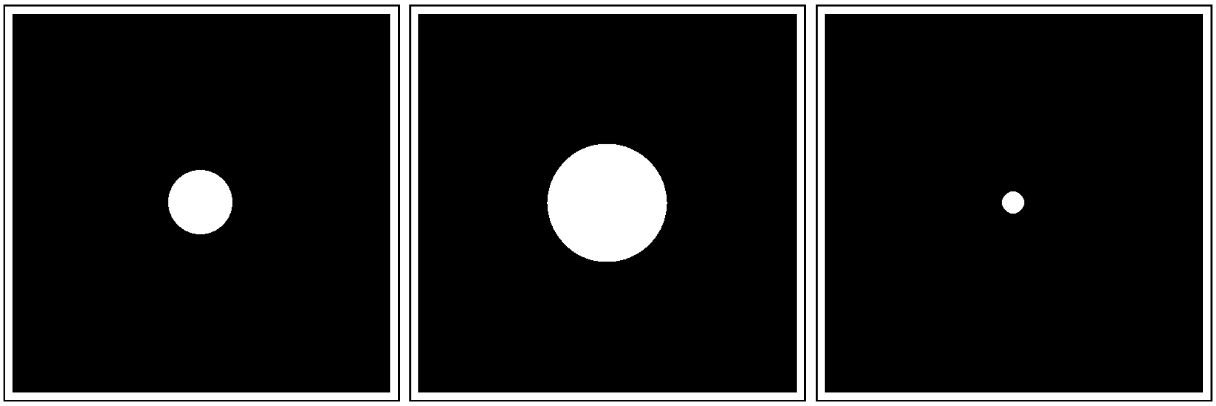


Abbildung 5.3: *Drei Repräsentanten der 3. Idealklasse. Hauptmerkmal ist die runde Form, die skaliert und verschiedene Rotationswinkel gedreht wurde.*

Iterationen) realisieren können. Bei den DCS-Netzen ist eine schnelle Konvergenz (nach maximal zehn Neuronen) des Testfehlers gegen null festzustellen. Die Ergebnisse sind in den folgenden Abbildungen graphisch dargestellt und beinhalten drei Versuchsreihen für repräsentative Merkmale basierend auf **16FD**, **MET** und **VAR**. Auf einer Kombination der einzelnen Merkmale in gemischten Vektoren wurde aufgrund der optimalen Generalisierfähigkeit der einzelnen Merkmale verzichtet.

Sicher ist das 3–Klassen–Problem gegenüber dem 8–Klassen–Problem in seiner Komplexität einfacher, wir können aber die Aussage wagen, daß bei einer ausreichend hohen Anzahl von Trainingsmustern und bei einer ausgewogenen Belegung der Klassen mit gleich vielen Beispielen, ein weit verbessertes Klassifikationsergebnis hätte erzielt werden können. Die Methodik der Merkmalsgewinnung ist stabil, da sie viele Invarianzen gewährleistet. Die Leistung der Netze ist bei realistisch hoher Anzahl von Trainingsmustern durch die zu lernenden Parameter noch weiter optimierbar. Uns standen z.B. in der Klasse 6 lediglich 13 Muster zur Verfügung. Bei der Fehlklassifikation zweier Algen dieser Klasse, hätte dies einen Testfehler von nahezu 20% bedeutet. Da die bei den Polarexpeditionen entnommenen Proben von einem eher größeren Volumen sind, ist es denkbar, daß eine automatische Erkennung aus solchen Datensätzen eher von Erfolg gekrönt seien könnte, als unsere auf 300 Exemplaren reduzierte selektierte Datenbank.

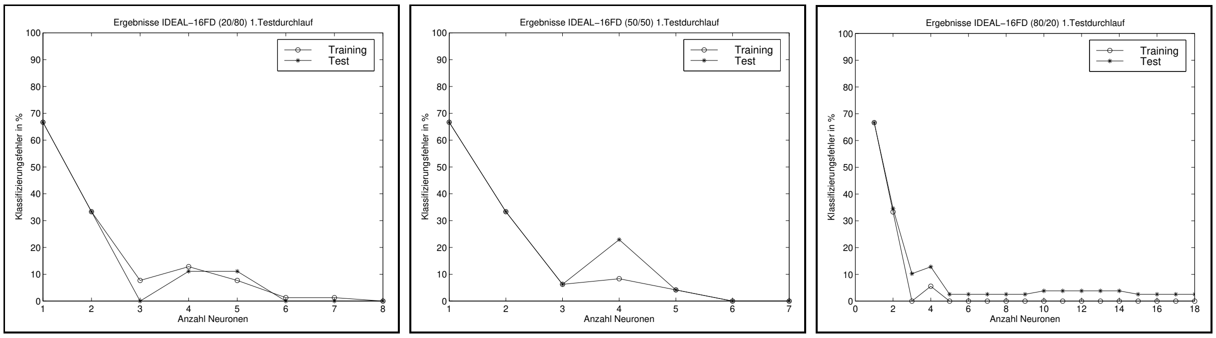


Abbildung 5.4: *Ergebnisse der DCS-Netze für das Merkmal* **IDEAL–16FD**.

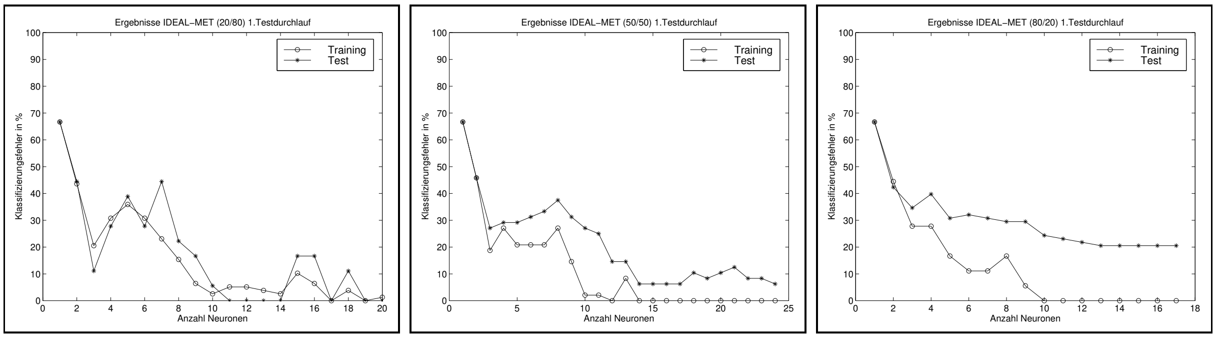


Abbildung 5.5: *Ergebnisse der DCS-Netze für das Merkmal* **IDEAL–MET**.

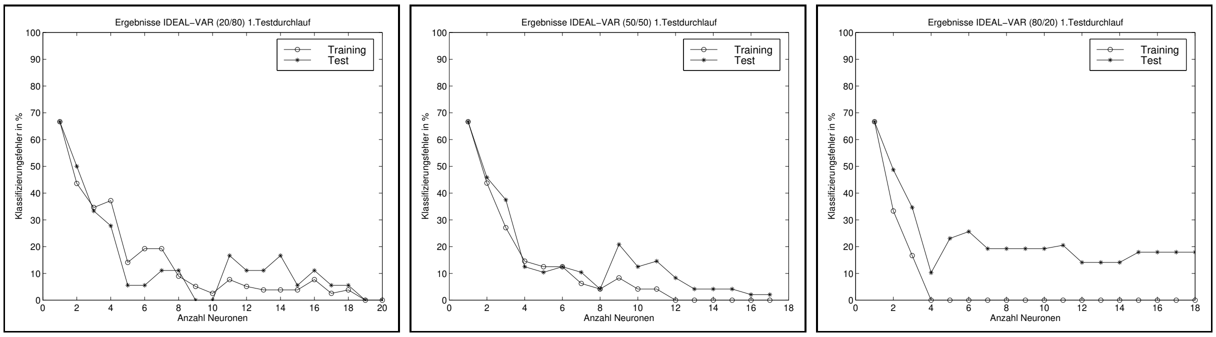


Abbildung 5.6: *Ergebnisse der DCS-Netze für das Merkmal* **IDEAL–VAR**.

5.4 Ausblick

Eine mögliche Erweiterung dieser Arbeit ist die Spezialisierung der beiden Netze auf reine Epifluoreszenzmikroskopiebilder. Um gegenüber Approximationsschwankungen des Fluoreszenzmarkers unempfindlich zu werden, könnte ein adaptives Binarisierungsverfahren über einen geeigneten Farbraum entwickelt werden. Eine Klassifikation nach Grauwertbildern, die auch die Textur der Algen miteinbeziehen ist ebenfalls denkbar. Dabei könnten insbesondere die Momente nach [Hu62] auf Grauwertbilder erweitert werden und auf ihre Eignung als singuläres Klassifikationsmerkmal hin untersucht werden. Bei den Neuronalen Netzen, die schon für die Gesichtserkennung aufgrund von Grauwertbildern eingesetzt worden sind, besteht in dieser Richtung kein Forschungsbedarf. Ihre Eignung als Klassifikator ist das Novum in ihrer bisherigen Anwendungshistorie.

Anhang A

Beweise der Translationsinvarianz

Behauptung: Die zentralen Momente sind invariant unter Translation.
Beweis: Sei $S^* = \{(x^*, y^*) | x^* = x + a, y^* = y + b, (x, y) \in S\}$.
Es gilt:

$$\bar{x}(S^*) = \frac{\sum_{(x^*,y^*)\in S^*} \ x^*}{M_{00}(S)} = \frac{\sum_{(x,y)\in S} \ x + a}{M_{00}(S)} = \bar{x}(S) + a \tag{A.1}$$

und analog

$$\bar{y}(S^*) = \bar{y}(S) + b\,. \tag{A.2}$$

$$\begin{aligned} \mu_{jk}(S^*) &= \sum_{(x^*,y^*)\in S^*} [x^* - \bar{x}(S)]^j \ [y^* - \bar{y}(S)]^k \\ &= \sum \{(x + a - [\bar{x}(S) + a]^j) \ (y + b - [\bar{y}(S) + b]^k)\} \\ &= \sum (x - \bar{x})^j \ (y - \bar{y})^k \\ &= \mu_{jk}(S) \end{aligned} \tag{A.3}$$

q.e.d.

Behauptung: Die normalisierten Momente sind invariant unter Translation
Beweis: Sei $S^* = \{(x^*, y^*) | x^* = ax + b, y^* = cy + d, (x, y) \in S\}$

$$\begin{aligned} m_{jk}(S^*) &= \frac{\sum_{(x^*,y^*)\in S^*} \left(\frac{x^* - \bar{x}(S^*)}{\sigma_x(S^*)}\right)^j \left(\frac{y^* - \bar{y}(S^*)}{\sigma_y(S^*)}\right)^k}{M_{00}(S^*)} \\ &= \frac{\sum_{(x,y)\in S} \ \frac{a^j [x - \bar{x}(S)]^j}{a^j (\sigma_x)^j (S)} \ \frac{c^k [y - \bar{y}(S)]^k}{c^k (\sigma_y)^k (S)}}{M_{00}(S)} \\ &= m_{jk}(S) \end{aligned} \tag{A.4}$$

q.e.d.

Literaturverzeichnis

[Abm94] W. Abmayr. *Einführung in die digitale Bildverarbeitung.* Teubner, Stuttgart, 1994.

[Alb91] J. Albertz. *Grundlagen der Interprätation von Luft und Satellitenbildern. Eine Einführung in die Fernerkundung.* Wissenschaftliche Buchgesellschaft, Darmstadt, 1991.

[AM98] E. Abraham-Mumm. Bestimmung der Gesichtspose mit künstlichen neuronalen Netzen. Master's thesis, Cognitive Systems Group, Inst. of Comp. Sci., Univ. of Kiel, Germany, 1998.

[Arb90] K. Arbter. *Affininvariante Fourierdeskriptoren ebener Kurven.* PhD thesis, Technische Universität Hamburg, 1990.

[BB93] H.-P. Blume and M. Bölter. Soils of casey station (wilkes land, antarctica). In D. Gilichinski, editor, *Joint Russian-American Seminar on Cryopendology and Global Change*, Post-Seminar Proceedings, pages 96–104. Pushchino, 1993.

[BCB97] E. Bayro-Corrochano and S. Buchholz. Geometric neural networks. In G. Sommer and J. J. Koenderink, editors, *Algebraic Frames of the Perception–Action Cycle*, Lecture Notes in Computer Science, 1315, pages 379–394. Springer–Verlag, Berlin, Heidelberg, 1997.

[BMD93] M. Bölter, R. Möller, and W. Dzomla. Determination of bacterial biovolume with epifluorescence microscopy: Comparison of size distributions from image analysis and size classification. *Micron 24*, pages 31–40, 1993.

[Böl94] M. Bölter. Estimations of microbial biomass by direct and indirect methods with special respect to monitoring programs. *Proc XV NIPR Symp. Polar Biol. 7*, pages 199–209, 1994.

[Böl95] M. Bölter. Distributions of bacterial numbers and biomass in soils and on plants from king george island (arctowski station, maritime antarctica). *Polar Biol. 15*, pages 115–124, 1995.

[Böl96] M. Bölter. A data base for arctic and antarctic non-marine algae (abstract). *Proc. Inter. Workshop Antarctic Research on Taxonomy and Ecology of Algae, Cracow, Poland*, pages 4–5, 1996.

[BS94] J. Bruske and G. Sommer. Dynamic cell structure learns perfectly topology preserving map. *Neural Computation*, 7:845–865, 1994.

[BSK98] M. Bölter, G. Sommer, and R. Kloos. *Eine Algendatenbank für antarktische Böden.* Miller, H. (Hrsg.) Koordiniertes Programm Antarktisforschung, Berichtskolloquium im Rahmen des Koordinierten Programms Antarktisforschung mit vergleichenden Untersuchungen in arktischen Eisgebieten, Ber. z. Polarforsch. 277, 1998.

[Buc97] S. Buchholz. Algebraische Einbettungen Neuronaler Netze. Master's thesis, Cognitive Systems Group, Inst. of Comp. Sci., Univ. of Kiel, Germany, 1997.

[Bur79] K. Burghardt. Transformationen zur lageinvarianten merkmalsgewinnung, vdi–fortschritt–bericht, reihe 10, nr. 7. Technical report, VDI–Verlag, Düsseldorf, 1979.

[Car13] G. W. F. Carlson. *Süßwasseralgen aus der Antarktis und den Falkland Inseln.* PhD thesis, Mathematisch–Naturwissenschaftliche Sektion der Philosofischen Fakultät zu Uppsala, 1913.

[Cyb89] G. Cybenko. Approximation by superposition of a sigmoidal function. *Mathematics of Control, Signals and Systems*, 2:303–314, 1989.

[DF97] W. Dunger and H. J. Fiedler. *Methoden der Bodenbiologie 2. neubearbeitete Auflage.* Verlag Fischer, Jena, Stuttgart, Lübeck, 1997.

[Din90] D. L. Dindal. *Soil Biology Guide.* Wiley-Intesince Publication, New York, Brisbane, Toronto, Singapore, 1990.

[EG95] H. Ettl and G. Gärtner. *Syllabus der Boden–, Luft– und Flechtenalgen.* Gustav Fischer Verlag, Stuttgart, New York, 1995.

[Gam69] H. Gams. *Makroskopische Süßwasser– und Luftalgen.* Gustav Fischer Verlag, Stuttgart, 1969.

[Hab91] R. Haberäcker. *Digitale Bildverarbeitung: Grundlagen und Anwendungen.* Hanser, München, Wien, 1991.

[Hab95] R. Haberäcker. *Praxis der digitalen Bildverarbeitung und Mustererkennung.* Hanser, München, Wien, 1995.

[HAH+94] Y. Hara, R. G. Atkins, S. H.Yueh, R. T.Shin, and J. A. Kong. Application of neural networks to radar image classification. *IEEE Transactions on Geoscience and Remote Sensing 32*, pages 100–109, 1994.

[Ham93] P. Hamilton. *Künstliche neuronale Netze. Grundprinzipien, Hintergründe, Anwendungen.* vde Verlag GmbH, Berlin, Offenbach, 1993.

[Hay94] S. Haykin. *Neural Networks. A Comprehensive Foundation.* Macmillan College Publishing Company, Inc., New York, 1994.

[HH93] D. R. Hush and B. G. Horne. Progress in supervised neural networks. *IEEE Signal Processing Magazine 1*, pages 8–39, 1993.

[HJM93] van den C. Hoek, H. M. Jahms, and D. G. Mann. *Algen.* Georg Thieme Verlag, Stuttgart, New York, 1993.

[Hu62] M.-K. Hu. Visual pattern recognition by moment invariants. *Proc. IRE–49*, pages 113–121, 1962.

[Jäh97] B. Jähne. *Digitale Bildverarbeitung. 4. Auflage.* Springer, Berlin, Heidelberg, New York, 1997.

[Koh95] T. Kohonen. *Self-Organizing Maps.* Springer Verlag, Berlin Heidelberg, 1995.

[Kra90] K. P. Kratzer. *Neuronale Netze. Grundlagen und Anwendungen.* Hanser, München, Wien, 1990.

[Kul94] A. D. Kulkarni. *Artificial Neural Networks for Image Understanding.* Van Nortrand Reinhold, New York, 1994.

[Kun93] S. Y. Kung. *Digital Neural Networks.* Prentice Hall, Inc., Englewood Cliffs, 1993.

[Lip87] R. P. Lippmann. An introduction to computing with neural nets. *IEEE Acoustics, Speech and Signal Processing Magazine*, pages 4–22, 1987.

[LK94] T.M. Lillesand and R.W. Kiefer. *Remote Sensing and Image Interpretation. 3. Auflage.* John Wiley and Sons, New York, 1994.

[MZ92] J. Mundy and A. Zisserman. *Geometric Invariance in Computer Vision.* MIT Press, England, 1992.

[Nan95] F. Nansen. *In Nacht und Eis. Die Norwegische Polarexpedition 1893–1896.* Heinrich Albert Verlag, Wiesbaden, 1995.

[PF77] E. Persoon and K.-S. Fu. Shape discrimination using fourier descriptors. *IEEE Trans., SMC–7(3)*, pages 170–179, 1977.

[PHR91] H. Pankow, D. Haendel, and W. Richter. *Die Algenflora der Schirmacheroase (Ostantarktika).* J. Cramer in der A. R. Gantner Verlag Kommanditgesellschaft, 1991.

[Rei93] T. H. Reiss. *Recognition Planar Objects Using Invariant Image Features.* Springer-Verlag, Heidelberg,, 1993.

[RGW86] D.E. Rumelhart, G. E. Hinton G.E, and R. J. Williams. *Learning Internal Representations by Error Propagation.* MIT Press, MA, 1986.

[Roj93] R. Rojas. *Theorie der neuronalen Netze. Eine systematische Einführung.* Springer-Verlag, Berlin, Heidelberg, New York, 1993.

[Ser82] J Serra. *Image Analysis and Mathematical Morphology.* Academic Press Inc., London, New York, 1982.

[Som97] G. Sommer. *Computer Vision Vorlesungsskript.* Lehrstuhl für Kognitive Systeme an der Christian–Albrechts–Universität zu Kiel, 1997.

[SPPW79] K. G. Seaburg, B. C. Parker, G. W. Prescott, and L. A. Whitford. *The Algae of Southern Victorialand, Antarctica, A Taxonomic and Distributional Study.* J. Cramer in der A. R. Gantner Verlag Kommanditgesellschaft in Biblithecia Phycologia, Band 46, Vaduz, 1979.

[Trö97] J. Tröster. Visuell basiertes Greifen gestützt auf neuronale netze. Master's thesis, Cognitive Systems Group, Inst. of Comp. Sci., Univ. of Kiel, Germany, 1997.

[Wan77] K. Wang. *Affin–invariant moment method of three dimensional object identification.* PhD thesis, Syracuse University, 1977.

[Was89] P. D Wasserman. *Neural Computing. Theory and Practice.* Van Nortrand Reinhold, New York, 1989.

[Wil24] N Wille. *Bericht der Deutschen Südpolar–Expedition 1901–1903, Band VIII, Heft IV (Botanik).* Erich von Drygalski (Hrsg.), Berlin und Leibzig, 1924.

[Zam89] P. Zamperoni. *Methoden der digitalen Bildsignalverarbeitung.* Vieweg, Braunschweig, 1989.

[Zel94] A. Zell. *Simulation Neuronaler Netze.* Addison-Wesley, Inc., Bonn, 1994.

Danksagung

An dieser Stelle ist es an mir all denen zu danken, die mir mit Rat und Tat zur Seite standen. Ich bin dankbar für auch noch so *unbedeutende* Kleinigkeiten und verweise an dieser Stelle an den Ordinarius für Semiotik Umberto Eco und seine humorvolle Kurzgeschichte *Wie man ein Vorwort schreibt* in dem Buch mit dem Titel *Wie man mit einem Lachs verreist und andere nützliche Ratschläge.*

Mein besonderer Dank gilt meinen Eltern.

Erklärung

Hiermit versichere ich, daß ich die vorliegende Arbeit selbständig verfaßt und ausschließlich die angegebenen Hilfsmittel und Quellen verwendet habe.

Kiel, den