KfK 4340
Dezember 1987

Codierung von stochastischen Binärbildern mit Hilfe von Bitschichtrechnern

A. Nasraoui
Institut für Material und Festkörperforschung

Kernforschungszentrum Karlsruhe

KERNFORSCHUNGSZENTRUM KARLSRUHE
Institut für Material- und Festkörperforschung

KfK 4340

Codierung von stochastischen Binärbildern mit Hilfe von Bitschichtrechnern

Abdelmalek Nasraoui

von der Naturwissenschaftlichen Fakultät der Technischen Universität
Carolo-Wilhelmina zu Braunschweig genehmigte Dissertation

Kernforschungszentrum Karlsruhe GmbH., Karlsruhe

Kernforschungszentrum Karlsruhe GmbH
Postfach 3640, 7500 Karlsruhe 1

ISSN 0303-4003

Codierung von stochastischen Binärbildern mit Hilfe von Bitschichtrechnern

Zusammenfassung

In der Bildverarbeitung sind aufgrund der zu manipulierenden großen Datenmengen effiziente Codierungsmethoden zur Datenkomprimierung sowohl bei der Ausführung von Alorithmen als auch bei der Übertragung und Speicherung von Bilddaten erforderlich. Während für sequentielle Rechner zahlreiche Codierungsverfahren entwickelt wurden, werden Codierungsmethoden für parallele Rechner weniger untersucht.
In der vorliegenden Arbeit werden vorwiegend auf Heuristik basierende parallele Algorithmen zur Codierung von Binärbildern entwickelt. Dabei stehen im Vordergrund die Datenkomprimierung von Binärbildern zur Archivierung oder Übertragung von Bilddaten einerseits und die Beschreibung deren zugrundeliegender stochastischer Szenen (aus der Materialkunde, Biologie...) im Sinne einer Klassifikation nach Größe und Form der Bildkomponenten andererseits.
Die Parallelisierung des bekannten sequentiellen Kettencodes wird anhand eines Codierungsalgorithmus mit einem hohen Parallelitätsgrad behandelt.

Coding of Stochastic Pictorial Data by Bit Hybrid Computer

Abstract

Due to the large amount of data to be manipulated in image processing, efficient coding methods for data compression are necessary both for the execution of algorithms and for transmission and storage of pictorial data.
While for sequential computers many coding methods have been developed, they have been less investigated so far for parallel computers. In the present work parallel coding algorithms, mainly based upon heuristics, are developed. Great importance is attributed to data compression of binary images for filing or transmission, on the one hand, and to description of the underlying stochastic scenes (from material science, biology ...) in terms of a classification by size and shape of the components of the image, on the other hand. The parallelization of the well-known sequential chain algorithm is treated by use of a coding algorithm with a high degree of parallelism.

Inhaltsverzeichnis:

0. Einleitung

Es liegt in der Natur der Sache, daß die Kreativität sich unter anderem von einem dem Menschen inhärenten Nachahmungstrieb führen läßt, dessen Sinn uns wahrscheinlich für immer verborgen bleiben wird, dessen Ziele wir jedoch zunehmend mit unterschiedlichen Erwartungen aufnehmen und denen wir gleichermaßen kritische Bedeutungen beimessen. Aus der Fülle der einem hier einfallenden Beispiele wollen wir das der künstlichen Intelligenz herausgreifen, insbesondere deren Anwendung in industrieller Handhabungstechnik und Robotik. Das Vorbild ist hier das Gehirn. Der nachzuahmende Prozeß ist die natürliche Informationsverarbeitung der über Sinnesorgane zum Gehirn geleiteten und dort wahrgenommenen Signale. Die verhältnismäßig um astronomische Größenordnungen primitivere künstliche Verarbeitung der durch vielfältige Sensoren (optische, akustische, taktile ...) aufgefangenen Signale findet in der Maschine statt, die seit der Entdeckung des Transistors zur nächsten unsere Gesellschaft verändernden - und die Tendenzen sind verstärkt spürbar - Entwicklung des zwanzigsten Jahrhunderts avanciert wird.

Die Menge der bei derartiger Informationsverarbeitung zu bewältigenden Daten kann unter Umständen sehr umfangreich werden. Im Falle von optischen Sensoren beispielsweise kann ein Bild aus bis zu 4096x4096 Bildpunkten bestehen. Allein wenn jeder Bildpunkt nur ein Bit (Bit=kleinste Informationseinheit) darstellt, sind über 10 Mio Bit pro Bild zu verarbeiten. Die Manipulation solcher Daten ist auf drei verschiedenen Ebenen zeit- und raumaufwendig: Bei der Ausführung von Bildverarbeitungsalgorithmen, bei der Übertragung und bei der Speicherung von Bildern.

Um die Ausführungszeiten der Algorithmen zu reduzieren sind grundsätzlich zwei Wege möglich: 1) Einsatz von geeigneten Rechnerarchitekturen (parallele Rechner, Pipeline Rechner, spezielle Hardware); 2) Einsatz von sequentiellen Rechnern und Anwendung der Algorithmen auf codierten (komprimierten) Daten. Um den Speicheraufwand und die Übertragungszeit zu reduzieren, ist eine Datenkompression notwendig.

Codierungsmethoden zur Reduktion von Speicheraufwand und Übertragungszeiten unter der Verwendung von sequentiellen Rechnern wurden ausführlich in der Literatur behandelt. Bei den diese Codierungsmethoden implementierenden Algorithmen wird während einer Bildtransformation der Wert eines Bildpunktes zu einem gegebenen Zeitpunkt t stets als Funktion der Werte seiner Nachbarn (in einer wohldefinierten Nachbarschaft) zu früheren Zeitpunkten t-1, t-2, ... ermittelt. Solche Algorithmen werden als sequentiell bezeichnet.

Werden dagegen parallele Rechner zur Bildcodierung eingesetzt, so ist eine unmittelbare Übersetzung von sequentiellen Algorithmen in parallele Algorithmen prinzipiell möglich, jedoch in der Regel ineffizient. Der meist irreführende Begriff "parallel" bezieht sich hier in Anlehnung zur Theorie der zellularen Automaten darauf, daß in einer Bildtransformation alle Bildpunkte gleichzeitig derselben globalen Transformation unterliegen, gemäß einer wohldefinierten, diese globale Transformation festlegenden lokalen Überführungsfunktion. Die Entwicklung von Algorithmen für diese Rechner - deren Einsatz sich angesichts der bedeutenden Fortschritte der VLSI-Technologie (Very Large Scale Integration) in verschiedenen Anwendungen zunehmend vollzieht - erfordert eine neue, zur Umformulierung der Probleme und Erarbeitung von Lösungen notwendige Denkweise. Dabei kann die Theorie der zellularen Automaten, die als Modelle von parallelen Rechnern gelten, von großem Nutzen sein.

In der vorliegenden Arbeit werden ausschließlich parallele Codierungsmethoden im obigen Sinne behandelt, die als primäres Ziel eine Datenkompression verfolgen. Dabei werden zusätzlich noch weitere, eine effiziente Codierung auszeichnende Kriterien, beispielsweise eine strukturierte Bilddarstellung als Vorstufe einer späteren Bildanalyse, eine progressive Bildcodierung und -rekonstruktion zur Vernachlässigung nicht relevanter Details oder zur Früherkennung der Bilder während interaktiver Suchvorgänge in Bilddatenbanken... gezielt berücksichtigt.

Die zu codierenden Bilder sind hier in erster Linie solche, die aus der Analyse von stochastischen Szenen (z.B. Proben aus der Materialkunde, Biologie etc...) stammen. Dabei werden nur Bilder behandelt, deren Bildpunkte jeweils durch ein Bit dargestellt werden. Solche Bilder werden Binärbilder genannt, und sie werden aus sogenannten Grauwertbildern, bei denen ein Bildpunkt durch i. allg. 8 Bit dargestellt wird, infolge einer Segmentierung (Schwellwertbildung) erzeugt.

Der Aufbau der Arbeit ist in der Feststellung begründet, daß vom informationstheoretischen Gesichtspunkt die vollständige Information eines Binärbildes offensichtlich in bestimmten Teilen des Bildes enthalten ist. Aus der Literatur ist bekannt, daß sowohl die Kontur als auch die Menge der lokalen Maxima (oft auch Skelett genannt) eine komprimierte Form des zugehörigen Binärbildes darstellen und daß aus diesen komprimierten Darstellungen das Originalbild exakt rekonstruiert werden kann. Basierend auf dieser bekannten Erkenntnis wird in dieser Arbeit den Fragen nachgegangen, zu welchen weiteren Formen ein Binärbild sich komprimieren läßt und inwieweit diese komprimierten Formen mit Hilfe von parallelen Transformationen bestimmt werden können. Den für diese Transformationen notwendigen theoretischen Hintergrund entnehmen wir der Theorie der zellularen

Automaten und stellen ihn im ersten Kapitel der Arbeit in Form von grundlegenden Definitionen dar. Begriffe aus der Terminologie der Bildverarbeitung und der mathematischen Morphologie, die in dieser Arbeit benutzt werden, werden mit Hilfe dieser Definitionen formuliert.

Im zweiten Kapitel werden zunächst die Konzepte der sequentiellen Codierung erläutert. Anschließend werden im Hinblick auf die Verwendung eines etwas speziellen parallelen Rechners, eines Bitschichtrechners, parallele (genauer globale) Transformationen bezüglich beliebiger Raster zur Codierung von Binärbildern definiert. Als Beispiel solcher Transformationen wird die klassische Mittelachsentransformation (MAT) als eine globale Transformation bezüglich des von Neumann- (bzw. Moore-) Rasters angegeben und eine Verbesserung ihrer Datenkompressionen vorgeschlagen.

Das dritte Kapitel befaßt sich mit der naheliegenden Erweiterung der Codierung mit einem Raster (z.B. die MAT) zur Codierung mit mehreren Rastern. Dabei wird sich herausstellen, daß diese Codierung als ein allgemeines Suchproblem formuliert werden kann, und es werden zwei Ansätze zur Bestimmung einer hierfür optimalen Lösung untersucht. Zur praktischen Anwendung dieser Codierung werden heuristische Suchstrategien vorgeschlagen.

Im vierten Kapitel wird das Heranziehen der heuristischen Codierungsmethoden des dritten Kapitels zur Beschreibung stochastischer Szenen, insbesondere zur Merkmalextraktion im Hinblick auf Klassifikation und Größenverteilung der Bildobjekte untersucht. Ein weiteres, bei der Analyse solcher Szenen wichtiges Merkmal bildet die Formbeschreibung. Dazu werden Codierungsmethoden vorgeschlagen, die

auf der Partitionierung des Bildes in Muster basieren und neben der Datenkompression in einem begrenzten Umfang auch zur Formbeschreibung herangezogen werden können.

Während im dritten und vierten Kapitel die komprimierte Form eines Binärbildes durch dessen Darstellung als Vereinigung bzw. Partition von geometrischen Mustern im Mittelpunkt stand, wird in diesem fünften Kapitel auf die Frage einer parallelen Konturcodierung eingegangen. Dazu wird ein Algorithmus mit einem relativ hohen Parallelitätsgrad angegeben, der im Gegensatz zur herkömmlichen Konturcodierung den Konturcode für alle Konturpunkte gleichzeitig bestimmt. Die eigentliche Konturverfolgung findet sinnvollerweise im sequentiellen Rechner statt, wobei allerdings die wesentlichen Schritte dazu vom parallelen Rechner bereits durchgeführt wurden.

In der Schlußbetrachtung (Kapitel 6) werden die im Rahmen dieser Arbeit erzielten Ergebnisse diskutiert und auf weitere offene Fragen der parallelen Codierung eingegangen.

1. Grundbegriffe und Definitionen

In der mittlerweile recht umfangreichen Literatur der Bildverarbeitung werden oft unterschiedliche Terminologien benutzt, die mangels einer einheitlichen zugrundeliegenden Theorie zu nicht immmer eindeutigen Interpretationen führen. Aus mindestens zwei Gründen kann (muß) diese Theorie die der zellularen Automaten sein: 1) Mehr und mehr verdeutlicht sich aufgrund der in der Bildverarbeitung vorkommenden Datenstrukturen (Arrays, Vektoren, ...) und der darauf anzuwendenden Transformationen (Filterung, Faltung, Transformationen der mathematischen Morphologie etc...) die Inhärenz der parallelen Algorithmen und der Bildverarbeitung. 2) Diese Tendenz ist nicht zuletzt darauf zurückzuführen, daß angesichts der drastischen Preissenkung der VLSI-Bauelemente parallele Bildverarbeitungssysteme inzwischen kommerziell geworden sind /Preston und Uhr (1982), Duff (1982), Preston und Duff (1984)/. Parallele Rechner bis zu 10^6 Prozessoren werden in den verschiedenen Bereichen der Bildverarbeitung eingesetzt /Reeves (1984)/. Mit dieser Entwicklung finden die anfangs als Modelle entworfenen zellularen Automaten /Unger (1958), Burks (1970), Vollmar (1979), Preston (1984)/ ihre Realisierung.

Grundlegend für die in diesem Kapitel eingeführten Definitionen ist /Vollmar (1979)/. Die dort benutzte Notation wurde hier weitgehend beibehalten insbesondere in den Definitionen 1.1, 1.3, 1.5 und 1.6. In den darauf folgenden Definitionen 1.7 bis 1.9 werden die zentralen Transformationen der mathematischen Morphologie (Erosion, Dilatation, Ouverture, Fermeture) in der Terminologie der zellularen Automaten formuliert.

Definition 1.1: Es bezeichnen Z^d die Menge aller d-Tupel (d=1,2,...) von ganzen Zahlen und A eine endliche nichtleere Menge von Symbolen, (Zustands-) Alphabet genannt. Sei $z_0 \in A$ ein ausge-

wähltes Symbol von A. Eine Abbildung $c : Z^d \rightarrow A$ heißt Konfiguration. Die Menge aller Konfigurationen wird mit C bezeichnet. Ein Element $u \in Z^d$ wird Zelle und c(u) Zustand der Zelle genannt. z_0 wird auch als Ruhezustand bezeichnet . Die Menge aller Zellen u mit $c(u) \neq z_0$ wird Träger der Konfiguration c genannt und mit tr(c) bezeichnet. Die Kardinalität von tr(c) wird mit #tr(c) (oder einfach #c) bezeichnet. Eine Konfiguration c heißt endlich, wenn tr(c) endlich ist.

Bereits an dieser Stelle wollen wir, bevor wir die klassischen Definitionen aus der Theorie der zellularen Automaten fortsetzen, die Begriffe "Binärbild" und "Grauwertbild" in Anlehnung zur Definition 1.1 als besondere Konfigurationen präzisieren.

Definition 1.2: Sei $G = \{1,\ldots,n\}^2 \subset Z^2$ und $A = \{0,1,\ldots,g-1\}$, $(g > 1)$. Eine Konfiguration $c : G \rightarrow A$ heißt ein g-Grauwertbild. c heißt Binärbild, wenn gilt: $A = \{0,1\}$. Die Menge aller auf G definierten Bilder wird mit C_n bezeichnet. Ein Element $u \in G$ wird Bildpunkt (auch Pixel oder Pel) mit dem Wert c(u) genannt.

Bemerkungen:

a) In dieser Arbeit werden ausschließlich auf G definierte Binärbilder behandelt.
b) Die Begriffe Binärbild und (endliche) Konfiguration werden gleichermaßen verwendet.
c) Der Träger eines Binärbildes wird gelegentlich "Informationsanteil" /Pecht (1981), Nasraoui (1982)/ und die Menge der Bildpunkte mit dem Wert 0 Hintergrund genannt.

Eine in der Bildverarbeitung sehr verbreitete Art von Operationen bilden die sogenannten Nachbarschaftsoperationen oder lokalen Transformationen. Sie lassen sich am elegantesten mit Hilfe der hier

ausgewählten Notation der Zellularautomaten formulieren.

Definition 1.3: Ein m-Tupel $N = (a_1,\ldots,a_m)$ von paarweise verschiedenen d-Tupeln $a_i \in Z^d$ heißt Nachbarschaftsindex. Eine Abbildung $\nu:(Z^d)^m \times Z^d \to (Z^d)^m$ definiert durch $\nu(N,u) = (u+a_1,\ldots,u+a_m)$ wird Nachbarschaft der Zelle u bezüglich N genannt. Jede Komponente von ν wird Nachbar von u genannt. Sei $\nu_N : Z^d \to (Z^d)^m$ eine Abbildung definiert durch : $\nu_N(u) = \nu(N,u)$. Sei $c^m : (Z^d)^m \to A^m$ eine Abbildung definiert durch $c^m(u_1,\ldots,u_m) = (c(u_1),\ldots,c(u_m))$. $c^m(\nu_N(u))$ wird Konfiguration der Nachbarschaft von u genannt.

Bemerkungen:

a) Die Wahl eines Ursprungs in N, etwa $a_1 = (0,\ldots,0)$ ist zweckmäßig.
b) Im Falle von auf G definierten Konfigurationen wird der Zustand der außerhalb von G liegenden Nachbarn der "Grenzzellen" stets als Ruhezustand z_0 angenommen.
c) Im allgemeinen ist die Reihenfolge der Komponenten von N und $\nu_N(u)$ irrelevant, so daß diese m-Tupel auch als Mengen aufgefaßt werden können. Wir werden dieselbe Notation N und $\nu_N(u)$ beibehalten und lassen die Differenzierung aus dem Kontext hervorgehen. Die Menge N wird dann Raster genannt.

Definition 1.4: Seien c eine Konfiguration, u und v zwei Elemente aus tr(c) (bzw. aus dem Hintergrund von c). u und v heißen zusammenhängend bezüglich eines Rasters N, wenn es eine Sequenz $(u_1,\ldots,u_k)$ von "Punkten" in tr(c) (bzw. im Hintergrund von c) gibt, so daß gilt:

1) $u_1 = u$ und $u_k = v$
2) u_i und u_{i+1} sind Nachbarn (bez. N) $(1 \leq i \leq k-1)$

Bemerkung: "Ist zusammenhängend mit" bildet eine Äquivalenzrelation (vgl. z.B. /Rosenfeld (1979)/), die Zellen von tr(c) (bzw. vom Hintergrund von c) in Äquivalenzklassen unterteilt; jede Äquivalenzklasse wird Komponente der Konfiguration genannt.

Zwei in der Verarbeitung von digitalen Bildern auf quadratischem Gitter oft verwendete Raster, bekannt unter den Namen von Neumann- und Moore-Raster, wollen wir hier explizit angeben. Sie werden insbesondere zur Festlegung des "Abstandes" (4- und 8-Metrik) in digitalen Bildern sowie der Konnektivität (connectedness) von Pixels (zur Bildung von Komponenten des Bildes) herangezogen /Rosenfeld (1982)/. Zur Definition dieser Raster benötigen wir die Einführung von zwei Normen.

Definition 1.5: Es bezeichnen $|u|$ und $||u||$ zwei Normen von $u \in Z^d$ definiert durch

$$|u| = \sum_{i=1}^{d} |u_i| \qquad \text{und} \qquad ||u|| = \max_{1 \leq i \leq d} \{ |u_i| \}$$

respektiv. Die Raster $H = \{u \,/\, |u| \leq 1\}$ und $M = \{u \,/\, ||u|| \leq 1\}$ werden von Neumann- und Moore-Raster respektive genannt.

Wie bereits erwähnt werden neben der Unterteilung eines Binärbildes in Hintergrund und Träger wiederum benachbarte Punkte (bez. H bzw. M) gleichen Wertes (0 oder 1) sowohl im Hintergrund als auch im Träger mit Hilfe der Äquivalenzrelation "ist zusammenhängend mit" zu Komponenten zusammengefaßt. Die Komponenten des Trägers werden Objekte genannt. Die in Objekten eingeschlossenen Komponenten des Hintergrundes werden Löcher genannt. Eine wichtige "Größe" in der Bildanalyse ist die sogenannte Eulerzahl, definiert als die Anzahl der Objekte minus die Anzahl der Löcher /Rosenfeld (1979), Gray (1971), Minsky und Papert (1969)/. Wir werden darauf in Kapitel 4 ausführlich eingehen, insbesondere auf deren Berechnung mit Hilfe

von lokalen Transformationen, die nun auch definiert werden können.

Definition 1.6: Eine Abbildung σ: $A^m \rightarrow A$ heißt lokale Transformation. Eine Abbildung τ: $C \rightarrow C$ heißt parallele Transformation. τ heißt globale Transformation, wenn sie mit Hilfe einer lokalen Transformation σ folgendermaßen definiert werden kann :

$$\forall\, u \in Z^d : \tau(c(u)) = \sigma(c^m(\nu_N(u)))$$

Die Menge aller globalen Transformationen wird mit T bezeichnet.

Bemerkung: Die globale Transformation τ läßt sich anhand des folgenden Diagramms als Komposition von Abbildungen leichter veranschaulichen:

$$\tau : Z^d \overset{\nu_N}{\rightarrow} (Z^d)^m \overset{c^m}{\rightarrow} A^m \overset{\sigma}{\rightarrow} A$$

Die in dieser Arbeit vorkommenden parallelen Transformationen sind ausschließlich globale Transformationen. Parallele Transformationen, die sich aus unterschiedlichen lokalen Transformationen zusammensetzen, führen zu inhomogenen zellularen Netzen und werden hier nicht betrachtet. Ein Beispiel solcher Transformationen und deren Anwendung im Bereich der Verarbeitung von 3-dimensionalen Szenen kann in /Nasraoui und Vollath (1985)/ gefunden werden.

Wir werden in den folgenden Definitionen die zentralen Transformationen der mathematischen Morphologie mit den Termini der zellularen Automaten formulieren. Ursprünglich von Minkowski als Mengenoperationen definiert /Hadwiger (1950)/ und später in der mathematischen Morphologie wiederaufgegriffen /Matheron (1967), Serra (1969)/ spielen diese Transformationen in der Bildverarbeitung eine wesentliche Rolle /Serra (1982), Sternberg (1980), Vollath et al. (1985)/.

Definition 1.7: Seien c eine Konfiguration, $A = \{0,1\}$, $N = (a_1,\ldots,a_m)$ ein Nachbarschaftsindex und $\underline{N}$ der um den Ursprung

(z.B. die erste Komponente von N) gespiegelte Nachbarschaftsindex N.

1) Eine globale Transformation $D_N : C \rightarrow C$ definiert durch folgende lokale Transformation $\delta : A^m \rightarrow A$ gemäß :

$$\delta(c^m(\nu(N,u))) = \begin{cases} 1 & \text{falls } c^m(\nu(\underline{N},u)) \neq (0,\ldots,0) \\ 0 & \text{sonst} \end{cases}$$

heißt Dilatation (der Konfiguration c mit dem Raster N).

2) Eine globale Transformation $E_N : C \rightarrow C$ definiert durch folgende lokale Transformation $\varepsilon : A^m \rightarrow A$ gemäß :

$$\varepsilon(c^m(\nu(N,u))) = \begin{cases} 1 & \text{falls } c^m(\nu(N,u)) = (1,\ldots,1) \\ 0 & \text{sonst} \end{cases}$$

heißt Erosion (der Konfiguration c mit dem Raster N).
Das Raster N wird strukturierendes Element genannt.

Die lokalen Transformationen δ und ε in der obigen Definition wiederspiegeln das in /Serra (1969)/ eingeführte Gesetz des "alles oder nichts" (la loi du tout ou rien) zur Definition der Erosion und Dilatation: Bei der geometrischen Veranschaulichung der beiden Operationen im Falle beispielsweise von Binärbildern wird dieses Gesetz folgendermaßen interpretiert: Jeder Bildpunkt wird mit dem Ursprung des verwendeten strukturierenden Elementes abgetastet. Ist das strukturierende Element in einem Objekt des Bildes ganz enthalten, so gehört der Punkt zur erodierten Konfiguration. Berührt das um seinen Ursprung gespiegelte strukturierende Element ein Objekt im Bild, so gehört der Punkt zum dilatierten Bild.

Die Ausführung der Erosion und Dilatation auf einem zellularen Automaten erfolgt mit Hilfe von einfachen Überführungsfunktionen: Bei der Erosion einer gegebenen Konfiguration mit einem strukturierenden Element N bleibt ein Automat im Zustand 1 genau dann, wenn alle seine Nachbarn bezüglich N im Zustand 1 sind. Bei der Dilatation nimmt jeder Automat den Zustand 1 genau dann ein, wenn er

selbst oder einer seiner Nachbarn bez. des um den Ursprung gespiegelten strukturierenden Elementes $\underline{N}$ im Zustand 1 ist. Diese Überführungsfunktionen können unmittelbar aus der folgenden zur Definition 1.7 äquivalenten Definition entnommen werden.

Definition 1.8: Es bezeichne $\pi_i(u)$ die Projektion der i-ten Komponente von $u \in Z^d$. Die lokalen Transformationen δ und ε von Definition 1.7 können mit Hilfe der Projektion π_i folgendermaßen definiert werden:

$$\delta(c^m(\nu(N,u))) = \begin{cases} 1 & \text{falls } \exists i: \pi_i(c^m(\nu(\underline{N},u))) = 1 \\ 0 & \text{sonst} \end{cases}$$

$$\varepsilon(c^m(\nu(N,u))) = \begin{cases} 1 & \text{falls } \forall i: \pi_i(c^m(\nu(N,u))) = 1 \\ 0 & \text{sonst} \end{cases}$$

Die Kombination der beiden Operationen Erosion und Dilatation führt zu nützlichen Transformationen, die breite Anwendung in der Bildverarbeitung finden. Eine ausführliche Behandlung dieser und weiterer Transformationen im Rahmen eines unter mathematischer Morphologie bekannt gewordenen Zweiges der Bildverarbeitung geben /Matheron (1969),(1976), Serra (1969),(1982)/. Wir begnügen uns hier lediglich mit zwei solchen Transformationen, der Ouverture und der Fermeture, die wir im folgenden definieren wollen.

Definition 1.9: Sei c eine Konfiguration und N ein strukturierendes Element. Ferner seien D_N und E_N eine Dilatation und Erosion gemäß Def.1.7 . Die globalen Transformationen O_N, $F_N : C \to C$ definiert durch :

$$O_N(c) = D_N(E_N(c))$$

$$F_N(c) = E_N(D_N(c))$$

werden Ouverture bzw. Fermeture genannt.

Oft wird in der Literatur ein Binärbild B als eine Menge von Punkten

in Z^2 aufgefaßt, d.h. im Sinne von unseren Definitionen mit dessen Träger tr(B) (vgl. Definition 1.1) identifiziert und globale Transformationen von B, insbesondere die Erosion und Dilatation und deren Kombinationen werden folglich als Mengenoperationen definiert. Dabei werden für die Dilatation und Erosion die Symbole $\oplus$ und $\ominus$ verwendet. Die Dilatation und die Erosion von B mit dem strukturierenden Element N (als Menge von Punkten aufgefaßt) werden dann definiert durch:

$$B \oplus N = \bigcup_{x \in N} B_x \qquad \text{und} \qquad B \ominus N = \bigcap_{x \in \underline{N}} B_x$$

wobei B_x das um x verschobene B bedeutet. Diese Schreibweise entspricht der Definition von der Minkowski'schen Summe und Differenz zweier Punktmengen nach /Hadwiger (1950)/.

Eine weitere verbreitete Notation für digitale Bilder im allgemeinen ist die Matrizenschreibweise (vgl. z.B. /Pratt (1978), Pavlidis (1982)/): Das Bild wird als Matrix von ganzen Zahlen aufgefaßt. Von dieser Notation werden wir in dieser Arbeit keinen Gebrauch machen.

2. Datenkompression von Binärbildern

Die in der Bildverarbeitung auftretenden großen Mengen von Bilddaten (bis zu 4096x4096 Pixels) führen bekanntlich zu hohem Speicheraufwand und langen Übertragungs- und Ausführungszeiten, insbesondere wenn die Verarbeitung auf herkömmlichen sequentiellen Rechnern erfolgt. Angesichts der infolgedessen notwendigen Datenkompression wurden in den letzten Jahrzehnten zahlreiche mathematische und heuristische Codierungsverfahren entwickelt. In diesem Kapitel werden die Grundkonzepte dieser Verfahren angeschnitten und weitere zur Implementierung auf parallele Rechner geeignete Verfahren beschrieben.

2.1 Datenkompression in der klassischen Codierung

Bereits anfangs der fünfziger Jahre wurden Konzepte zur Datenkompression (wir benutzen auch die Bezeichnung Datenreduktion in einer äquivalenten Weise) von Binärbildern ausgearbeitet /Laemmel (1951), Elias (1955)/, die in den meisten heutigen Codierungsverfahren wiedererkannt werden können. Grund dieser früheren Entwicklung war nicht weniger die zu dieser Zeit als Novum geltende mathematische Formulierung der Informationstheorie /Shannon und Weaver (1949)/.

Vom Gesichtspunkt der Informationstheorie ist ein Binärbild als eine Folge von Zeichen aus dem Zeichenvorrat {0,1} aufzufassen, die von einer Nachrichtenquelle mit gewissen Wahrscheinlichkeiten erzeugt werden. Eine derartige Entstehung von Zeichenfolgen wird als stochastischer Prozeß bezeichnet. Die einzelnen Zeichen wirken dabei als Informationsträger, deren Anordnung als Nachricht interpretiert wird. Das Maß an Information, das einem Zeichen zukommt, d.h. der Informationsgehalt pro Zeichen wird Entropie der Nachrichtenquelle

genannt. Aufgrund der statistischen Abhängigkeit der Zeichen in einer Folge läßt sich eine Codierung der von der Nachrichtenquelle erzeugten Zeichen immer finden, die folgenden, als Hauptergebnisse der Shannon'schen Informationstheorie zu betrachtenden Bedingungen genügen:

a) Wie effizient der Code auch sein mag, der Informationsgehalt eines codierten Zeichens kann nicht kleiner als die Entropie der Nachrichtenquelle sein;
b) Effiziente Codierungen existieren, die die Entropie der Nachrichtenquelle annähern.

Zur Präzisierung der Begriffe Information, Entropie, Nachricht, Codierung..., und zur mathematischen Abhandlung der als Lehrsätze der Informationstheorie bekannten Ergebnisse a) und b) sei auf die Literatur der Informationstheorie, insbesondere auf /Shannon und Weaver (1949), Wolfowitz (1978)/ verwiesen.

Zur Annäherung der Entropie der Nachrichtenquelle werden bei der Codierung von Binärbildern statistische Modelle mit Hilfe von Verbundwahrscheinlichkeiten und bedingten Wahrscheinlichkeiten der Zeichen herangezogen /Huang (1977), Arps (1979)/. Verbundwahrscheinlichkeiten führen zu den bekannten Codierungsverfahren wie der Block-Codierung /z.B. Laemmel (1951), Kunt und Johnsen (1980)/, der Lauflänge-Codierung /Capon (1959), Huang (1977)/ etc... Auf bedingten Wahrscheinlichkeiten basieren die sogenannten Prädiktiv-Codierungen /z.B. Elias (1955), Preuss (1975)/. Vergleiche der bei diesen Codierungsverfahren erzielten Entropien bezüglich ausgewählter Binärbilder können unter anderem aus /Musmann und Preuss (1977)/ entnommen werden.

Von den Größen der Informationstheorie wollen wir an dieser Stelle lediglich den als Maß zum Effizienzvergleich (im Sinne von b)) von

Codierungsverfahren anerkannten Kompressionsfaktor benutzen. Mit Hilfe des Kompressionsfaktors lassen sich in der Tat heuristische Codierungsverfahren leichter auf Effizienz beurteilen und untereinander vergleichen, obgleich mathematische, auf Theoremen der Informationstheorie basierende Modelle für solche Codierungsverfahren aufgestellt werden können.

Definition 2.1: Sei $c : G \rightarrow A$ ein Binärbild. Werden n^2 Bit benötigt, um c zu speichern und wird der Speicheraufwand infolge eines Codierungsverfahrens auf m Bit reduziert, so wird $CF = n^2/m$ Kompressionsfaktor genannt.

Von besonderer Bedeutung sind Codierungsverfahren, die neben einer exakten Rekonstruktion des ursprünglichen Bildes und einem Kompressionsfaktor $CF > 1$ zusätzlich noch weitere Kriterien (vgl. Abschnitt 2.1) erfüllen. Codierungsalgorithmen werden i. allg. als effizient bezeichnet, wenn sie:

1) bildanalytische Elemente enthalten: Darunter fallen etwa eine Beschreibung des Bildes mit Hilfe von Verteilungen über Größe und Form von Komponenten, topologische Beziehungen zwischen den Komponenten sowie deren hierarchischer Aufbau etc...
2) eine weitere Verarbeitung der codierten Daten erlauben,
3) eine geringe Komplexität in der Implementierung aufweisen,
4) eine einfache, möglichst billige Hardware erfordern.

Zu diesen allgemein anerkannten Effizienzkriterien kommen in der Regel noch zusätzliche anwendungsorientierte Kriterien hinzu. Die Güte der Codierungsalgorithmen resultiert demnach aus einer geeigneten Gewichtung der an eine Codierung gestellten Anforderungen. Zahlreiche, meistens auf Heuristik basierende Codierungsverfahren wur-

den in den letzten Jahrzehnten mit dem Ziel entwickelt, die hier aufgezählten Auswahlkriterien optimal zu erfüllen. Eine Bibliographie hierfür gibt /Arps (1980)/.

Unseres Wissens wurden alle diese Codierungsverfahren implementierende Algorithmen für herkömmliche sequentielle Rechner konzipiert. Die Frage, inwieweit parallele Rechner für die Codierung von Bildern (Binär- sowie Grauwertbildern) eingesetzt werden können, wurde bisher nicht gestellt. Vom Einsatz von parallelen Rechnern in den verschiedenen Disziplinen aus Wissenschaft und Technik (z.B. zur Lösung von partiellen Differentialgleichungen aus verschiedenen Gebieten der Physik, zur Lösung von Problemen der numerischen Mathematik, Bildverarbeitung, Handhabungstechnik etc...) werden hauptsächlich höhere Geschwindigkeiten bei der Ausführung von Algorithmen erwartet. Sie werden auch dann erzielt, wenn die Probleme sich so formulieren lassen, daß parallele Algorithmen (Stone (1973), Feilmeier (1977), Sakharov (1984), Yamashita (1985)) zu deren Lösungen entwickelt werden können. In der Bildverarbeitung beispielsweise können aufgrund der Datenstrukturen der zu manipulierenden Objekte (Vektoren, Matrizen) zahlreiche Aufgaben mit Hilfe von parallelen Algorithmen effizient gelöst werden. Aus diesem Grund und angesichts der enormen Fortschritte der VLSI-Technologie, die zu drastischen Preissenkungen der Bauelemente geführt haben (und weiterhin führen werden), werden parallele Rechner auch auf diesem Gebiet zunehmend eingesetzt.

Wir werden in diesem und in den nächsten Kapiteln Algorithmen beschreiben, die Antworten, wenn auch manchmal nur partiell, auf die Frage der Codierung von Binärbildern mit Hilfe von parallelen Rechnern geben.

2.2 Codierung mit parallelen Rechnern

Nach der in /Flynn (1966)/ vorgeschlagenen Taxonomie der Rechnersysteme werden parallele Rechner in sogenannten SIMD- (Single-Instruction Multiple-Data), MIMD-Maschinen (Multiple-Instructions Multiple-Data) und MISD (Multiple-Instructions Single-Data) eingeteilt. SIMD-Maschinen werden auch als Feldrechner (oder Vektorrechner) bezeichnet, da die Befehle sich auf ganze Felder (Vektoren) von Daten beziehen. Feldrechner umfassen in der Regel mehrere gleichartige Prozessoren (Processing Elements (PE)), die zum gleichen Zeitpunkt den gleichen Befehl ausführen. Dabei operiert jeder Prozessor auf einem ihm zugeteilten Datenfeld. Zu den bekanntesten Feldrechnern zählen der DAP (Distributed Array Processor) mit 64x64 1-Bit Prozessoren /Flanders et al.(1977)/, der MPP (Massively Parallel Processor) mit 128x128 1-Bit Prozessoren /Batcher (1982)/ und der CLIP4 (Cellular Logic Image Processor) mit 96x96 1-Bit Prozessoren /Preston und Duff (1984)/.

Mit MIMD-Maschine wird ein in der Regel aus unterschiedlichen Prozessoren bestehendes Rechnersystem bezeichnet, bei dem alle Prozessoren an derselben Aufgabe arbeiten. MISD-Maschinen sind von keiner praktischen Bedeutung.

Die im Rahmen der vorliegenden Arbeit entwickelten parallelen Algorithmen beziehen sich in erster Linie auf Feldrechner, obgleich zu deren Implementierung ein sogenannter Bitschichtrechner /Pecht et al. (1982)/ herangezogen wurde. Aufgrund ihrer Manipulation von ganzen Binärbildern als Operanden in logischen Bildoperationen (Verschiebungen der Operanden zueinander inbegriffen) eignen sich Bitschichtrechner (z.B. der TAS (Texture Analysis System) /Nawrath und Serra (1979)/, der PICAP (Picture Array Processor) /Kruse (1973)/, der Cytocomputer /Sternberg (1978)/...) in besonderem Maße zur Im-

plementierung paralleler Algorithmen, auch wenn sie meistens nur über einen einzigen als Pipeline benutzten 1-Bit Prozessor verfügen, und finden daher einen breiten Einsatz in der Bildverarbeitung sowohl in der Forschung als auch in der Industrie /Preston und Duff (1984)/. Wir werden im nächsten Abschnitt eine kurze Beschreibung des bereits oben erwähnten Bitschichtrechners /Pecht et al. (1982)/ geben.

2.2.1 Der Bitschichtrechner

Der Bitschichtrechner ist eine wesentliche Komponente eines in Fig. 2.1 dargestellten Bildverarbeitungssystems, genannt PACOS (Pattern Considering System) /Vollath et al. (1985)/. Der gesamte Programmablauf und die Steuerung der einzelnen Systemkomponenten wird von einem Wirtsrechner (z.B. PDP-11/23, 8086-Prozessor...) kontrolliert, insbesondere die vom Bitschichtrechner auszuführenden Funktionen: Segmentierung eines von einer Fernsehkamera erfaßten Grauwertbildes zu einem Binärbild, Ausführung von logischen Operationen auf Binärbildern, Übertragung von Binärbildern zwischen dem Bitschichtrechner und dem Wirtsrechner. Fig. 2.2 zeigt die Hardware-Struktur des Bitschichtrechners: Bis zu 32 Bitschichten (256x256 Binärmatrizen) werden in je 64 Kbit Speicher abgelegt. Dabei entspricht dem (i,j)-ten Element der Matrix die Adresse a = 256*i + j. Bei einer logischen Operation zwischen zwei Bitschichten werden mit Hilfe des Adreßgenerators die entsprechenden beiden 64K Speicher in 50 ns Takt ausgelesen und bitweise in einer arithmetisch-logischen Einheit (ALU: arithmetical logical unit) verknüpft. Beinhaltet eine logische Operation eine Verschiebung einer Bitschicht um einen Vektor (dx,dy), so werden mit Hilfe eines Verschiebungsaddierers die Adressen a' der verschobenen Matrixelemente gemäß a' = a + 256*dx + dy berechnet. Die in diesem Fall errechneten Adressen von Elementen

außerhalb der Matrix werden durch einen Parameter (Randbit) spezifiziert, der angibt, ob sie den Wert 0 oder 1 einnehmen. Auf dem Weg zur Speicherung des Ergebnisses einer logischen Operation zwischen zwei Bitschichten in eine ausgewählte Ergebnis-Bitschicht passiert der Bitstrom einen Zähler, der die Anzahl der '1'-Bits zählt, sowie eine (in Fig. 2.2 nicht gezeichnete) Hardware-Einrichtung, die die ersten 10K Adressen a der '1'-Bits in einem RAM (random access memory) speichert. Eine logische Operation zwischen zwei Bitschichten dauert etwa 10 ms.

Im Rahmen der für das gesamte Bildverarbeitungssystem entwickelten Software stehen dem Benutzer mit den zahlreichen Basisroutinen, dem virtuellen System /Pecht (1984)/ sowie dem Interpreter /Friehmelt et al. (1985)/ hilfreiche Softwarepakete zur Entwicklung von Algorithmen auf verschiedenen Ebenen zur Verfügung.

2.2.2 Globale Transformationen zur Datenkompression

Vom Gesichtspunkt eines Zellularautomaten wird der Codierungsprozeß eines Binärbildes im wesentlichen als eine Folge von globalen Transformationen einer das Binärbild darstellenden Anfangskonfiguration aufgefaßt, die zur Erzeugung eines Codes des Bildes führen. Der Code kann aus besonderen, während der Transformationen gewonnenen '1'-Zellen (Konfigurationspunkte mit dem Wert 1) zusammen mit den zu diesen Zellen geführten Schritten bestehen. Dabei werden die '1'-Zellen mit Hilfe deren Koordinaten und die assoziierten Transformationen in geeigneter Weise dargestellt. Solche '1'-Zellen wollen wir Wurzelzellen nennen und im nächsten Abschnitt genauer definieren. Auf Wurzelzellen basierende Codes werden wir ausführlich in den Kapiteln 3 und 4 behandeln.

Der Code kann aber auch nur aus den besonderen '1'-Zellen bestehen, wenn sie zu einer eindeutigen Decodierung des Bildes führen können. Solche Zellen sind beispielsweise die Konturzellen der Bildobjekte (vgl. Kapitel 5), oder die Punkte einer der Konfigurationen, die bei der bekannten sukzessiven Anwendung der 'XOR'-Operation zur Reproduzierbarkeit von Mustern entstehen /Amoroso und Cooper (1970)/. In diesem Fall gehören noch zum Code der Schritt, bei dem diese Konfiguration entstanden ist, sowie das in der 'XOR'-Operation benutzte Raster. Wir vermuten, daß diese auf der Reproduzierbarkeit der Muster basierende Codierungsart zu interessanten Ergebnissen führen kann. Wir werden sie jedoch im Rahmen dieser Arbeit nicht weiterverfolgen.

Bei der Implementierung der zur Erzeugung solcher Codes entwickelten Algorithmen werden die globalen Transformationen von Konfigurationen auf Bitschichtoperationen zurückgeführt, bei denen die lokalen Nachbarschaftsoperationen durch einfache Verschiebungen (shifts) realisiert werden. Analog zum Zellularautomaten werden auch hier alle Zellen einer Bitschicht derselben lokalen Nachbarschaftsoperation unterworfen. Wir werden im Rahmen dieser Arbeit nur auf globalen Transformationen basierende parallele Codierungsalgorithmen behandeln.

Sollten jedoch in einer Bitschicht unterschiedliche (z.B. ortsabhängige) lokale Nachbarschaftsoperationen ausgeführt werden, so kann eine geeignete Wahl von Masken zur Simulation dieser Operationen herangezogen werden. Die Masken sind so zu wählen, daß nur der gerade zu bearbeitende Teil der Bitschicht gemäß der lokalen ortsabhängigen Nachbarschaftsoperation transformiert wird. Das Ergebnis der gesamten parallelen (nicht globalen) Transformation setzt sich dann aus den einzelnen ortsabhängigen Teilergebnissen globaler Transformationen zusammen. Eine Anwendung solcher Transformationen

zu Verzerrungsproblemen 3-dimensionaler Szenen kann in /Nasraoui und Vollath (1986)/ gefunden werden. Ähnlich können auch Codierungsalgorithmen mit ortsabhängigen globalen Transformationen untersucht werden. Dies würde jedoch aus dem Rahmen dieser Arbeit fallen und kann daher nicht berücksichtigt werden. Wir werden uns hier lediglich auf parallele Codierungsalgorithmen beschränken, die auf globalen Transformationen basieren und folglich mit Zellularautomaten im Sinne von Definition 1.7 realisiert werden können.

2.3 Wurzelzellen - Verallgemeinerte lokale Maxima

Eine im folgenden näher spezifizierte Obermenge der Wurzelzellen bilden die sogenannten lokalen Maxima /Arcelli et al. (1981), Rosenfeld und Kak (1982)/. Wir werden in diesem Abschnitt globale Transformationen definieren, mit deren Hilfe die bisherige Definition der lokalen Maxima verallgemeinert wird.

Definition 2.2: Seien $c \in C_n$ ein Binärbild und $t_N \in T$ eine globale Transformation (bezüglich eines Rasters N). Eine k-fache (k=0,1,...) Komposition von t_N wird rekursiv definiert durch

$$t_N^0(c) = c$$

$$t_N^k(c) = t(t_N^{k-1}(c)) \qquad (k > 0).$$

Bemerkungen:

a) k-fache Kompositionen von globalen Transformationen werden häufig in Iterationsprozessen zur Isolierung bestimmter Punktmengen (wie es in den folgenden Ausführungen der Fall sein wird) oder i. allg. zur Extraktion bestimmter Merkmale aus dem Bild benutzt.

b) Wir werden gelegentlich in der Schreibweise $t_N^k(c)$ die Klammern weglassen.

Definition 2.3: Seien $c \in C_n$ ein (auf $G = \{1,\ldots,n\}^2$ definiertes) Binärbild, E_N die Erosion (vgl. Def.1.7) und O_N die Ouverture (vgl. Def.1.9) bezüglich eines Rasters N. Eine globale Transformation γ_N wird folgendermaßen definiert :

$$\gamma_N^k(c) = O_N(E_N^{k-1}c) \qquad (k > 0)$$

Für alle $u \in G$ seien (mit Hilfe von E_N und γ_N) folgende globale Transformationen ρ_N und $\sigma_N^{(k)}$ $(k > 0)$ definiert :

$$\rho_N c(u) = \begin{cases} k & \text{falls } E_N^k(c(u)) = 0 \text{ und } E_N^{k-1}c(u) = 1 \\ 0 & \text{sonst} \end{cases}$$

$$\sigma_N^{(k)}c(u) = \begin{cases} k & \text{falls } \gamma_N^k(c(u)) = 0 \text{ und } E_N^{k-1}c(u) = 1 \\ 0 & \text{sonst} \end{cases}$$

ρ_N wird Wachstumstransformation genannt. $\sigma_N^{(k)}$ wird Wurzeltransformation der k-ten Generation genannt.

Schließlich sei $W_N^{(k)}$ eine globale Transformation definiert durch

$$W_N^{(k)}c(u) = \begin{cases} 1 & \text{falls } \sigma_N^{(k)}c(u) = k \\ 0 & \text{sonst} \end{cases}$$

$W_N^{(k)}c$ wird Menge der Wurzelzellen (von c) der k-ten Generation genannt.

Bemerkungen:

a) ρ_N und $\sigma_N^{(k)}$ sind globale Transformationen bez. des Alphabets $A = \{1,\ldots,n\}$. Dabei nimmt k die Werte $1,2,\ldots,m$ ein, wobei m den Schritt angibt, bei dem die k-fache Erosion von c mit N zur

Null-Konfiguration ($\underline{0}$) führt (d.h. $E_N^m(c)=\underline{0}$).

b) In der Schreibweise $t^{(k)}$ ist k ein Index und soll nicht mit der k-fachen Komposition t^k verwechselt werden .

c) Sei kN die rekursiv definierte Menge:

$$0N = \{(0,0)\}$$

$$(k+1)N = kN \oplus N$$

wobei $\oplus$ die Minkowski'sche Mengenaddition (Dilatation) bezeichnet. ρ_N ordnet dann jedem Bildpunkt die maximale Zahl k zu, so daß das -aus diesem Punkt als Ursprung wachsende- Muster kN im Träger von c enthalten ist. $\sigma_N^{(k)}(c)$ ist die Konfiguration der k-Zellen (d.h. mit dem Wert k) von $\rho(c)$, die in ihrer Nachbarschaft N maximal sind.

d) Für das besondere von Neumann-Raster H liefert ρ_H den minimalen Abstandswert eines jeden Bildpunktes zur Kontur seiner zugehörigen Bildkomponente. σ_H definiert durch

$$\sigma_H c(u) = \max \{ \sigma_H^{(k)} c(u); (k=1,\ldots,m) \}$$

ist die diskrete Version einer ursprünglich in /Blum (1967)/ als Mittelachsenfunktion (medial axis function (MAF)) bzw. Skelettfunktion (skeletal function) eingeführten kontinuierlichen Transformation bezüglich des Einheitskreises anstelle von H. Die Menge der durch die Mittelachsenfunktion erhaltenen Punkte werden in /Blum (1967)/ Skelett genannt. σ_H ist in der Literatur /vgl. z.B. Rosenfeld und Kak (1982)/ jedoch mehr unter Mittelachsentransformation (MAT) bekannt. Das dabei erhaltene diskrete Skelett ist nicht immer zusammenhängend und wird deswegen auch Menge der lokalen Maxima genannt /vgl. z.B. Arcelli et al.

(1981)/.

Unter Skelett versteht man im allgemeinen die kleinste zusammenhängende Obermenge der lokalen Maxima /vgl. z.B. Stefanelli und Rosenfeld (1971), Pavlidis (1980), Montanari (1969)/. In Anlehnung zu dieser Terminologie wollen wir gelegentlich mit verallgemeinerten lokalen Maxima die Menge der Wurzelzellen bezeichnen.

Die globalen Transformationen γ_N, ρ_N, $\sigma_N^{(k)}$ und $W_N^{(k)}$ wurden so definiert, daß das ursprüngliche Binärbild c aus $\sigma_N^{(k)}(c)$ bzw. aus $W_N^{(k)}(c)$ rekonstruiert werden kann, wobei k die Anzahl der Anwendungen von $\sigma_N^{(k)}c$ bis zur Entstehung der k-ten Generation von Wurzelzellen darstellt. Die Menge der Wurzelzellen ist dabei nicht minimal. Solche Mengen werden in der Signalverarbeitung als redundant bezeichnet. Die Synthese des Originalbildes aus der Menge der Wurzelzellen sowie die Redundanz dieser Menge wollen wir als Propositionen formulieren.

Proposition 2.1: Seien $c \in C_n$ eine Konfiguration und $W_N^{(i)}c$ $(i=1,\ldots,m)$ die Konfigurationen aller Wurzelzellen von c. Ferner seien $\psi_i : C_n \to C_n$ und $\psi : (C_n)^m \to C_n$ Transformationen definiert durch :

$$\psi_i = D_N^{i-1}(W_N^{(i)}) \quad \text{und} \quad \psi(\psi_1,\ldots,\psi_m) = \bigvee_{i=1}^{m} \psi_i$$

wobei 'V' die logische OR-Funktion bedeutet.
Dann gilt:

$$\psi(\psi_1 c,\ldots,\psi_m c) = c \qquad (1)$$

Beweis: Wir werden zeigen, daß $tr(c) = tr(\psi(\psi_1,\ldots,\psi_m)c)$.

1) Sei $u \in tr(\psi(\psi_1 c,\ldots,\psi_m c))$. Dann :

$\exists\, i\colon 1 \le i \le m :\ u \in tr(\psi_i c)$ (per Def. von ψ)

$\Rightarrow$ $u \in tr(D_N^{i-1}(W_N^{(i)}c))$ (per Def. von ψ_i)

$\Rightarrow$ $u \in tr(D_N^{i-1}(E_N^{i-1}c))$, da $tr(W_N^{(i)}c) \subset tr(E_N^{i-1}c)$

$\Rightarrow$ $u \in tr(c)$,

da $tr(D_N^{i-1}(E_N^{i-1}c)) = tr((D_N^{i-1}E_N^{i-1})c) \subset tr(c)$

2) Sei $u \in tr(c)$. Dann :

a) Entweder $\exists i\colon\ 1 \le i \le m\colon\ u \in tr(W_N^{(i)}c)$

$\Rightarrow$ $u \in tr(D_N^{i-1}(W_N^{(i)}c))$, da $tr(W_N^{(i)}c) \subset tr(D_N^{i-1}(W_N^{(i)}c))$

$\Rightarrow$ $u \in tr(\psi_i c)$ $\Rightarrow$ $u \in tr(\psi(\psi_1,\ldots,\psi_m)c)$

b) Oder $\forall i\colon 1 \le i \le m : u \notin tr(W_N^{(i)}c)$

$\Rightarrow \exists i\colon 1 \le i \le m :\ u \in tr(D_N^{i-1}(W_N^{(i)}c))$,

da sonst $u \in tr(W_N^{(i)}c)$ im Widerspruch zur Annahme b).

Wählt man für das Raster N in Proposition 2.1 das (von Neumann-) Raster H bzw. das (Moore-) Raster M, so führt (1) zu der Rekonstruktion eines Binärbildes c aus dessen sogenannten Mittelachsentransformation, die als Codierungsverfahren bereits in der Literatur ausführlich behandelt wurde. Wir werden diese Transformation im nächsten Abschnitt als Beispiel einer globalen Transformation angeben.

Proposition 2.2: Seien c, $W_N^{(i)}c$ (i=1,...,m) und ψ aus Proposition 2.1. Dann existieren Konfigurationen c und $w_N^{(i)}c$, sowie Raster N,

so daß gilt :

$$tr(w_N^{(i)}c) \subset tr(W_N^{(i)}c)$$

$$\psi_i = D_N^{i-1}(w_N^{(i)}c)$$

und $$\phi(\psi_1 c,\ldots,\psi_m c) = c$$

Beweisskizze: Sei N das (Moore-) Raster M. $w_M^{(i)}c$ wird aus $W_M^{(i)}c$ folgendermaßen konstruiert: Aus jeder horizontalen (bzw. vertikalen) Folge von $j > 2$ benachbarten Wurzelzellen von $W_M^{(i)}c$ $(i>1)$ werden alle Zellen entfernt, die zwischen der $(1+(2i-1)k)$-ten und der l-ten liegen, wobei $l = \min\{ 1+(2i-1)(k+1), j \}$ und $k = 0,1,\ldots,\lfloor\frac{j-1}{2i-1} - 1\rfloor$ mit $\lfloor x\rfloor$= größte ganze Zahl kleiner oder gleich x.

Proposition 2.2 verdeutlicht die Redundanz in der Codierung einer Konfiguration c mit Hilfe der Wurzelzellen $W_N^{(i)}c$ $(i=1,\ldots,m)$, die eliminiert werden können, wenn kleinere Mengen von Wurzelzellen $w_N^{(i)}c$ zur Darstellung von c herangezogen werden. Im nächsten Abschnitt werden wir für den Fall der Mittelachsentransformation eine Methode in Form eines Satzes zur vollständigen Eliminierung dieser Redundanz beschreiben.

2.4 Die Mittelachsentransformation als Beispiel einer globalen Transformation

Wie bereits in der Bemerkung c) zur Definition 2.3 erwähnt wurde, stellt die Mittelachsentransformation (MAT) einen Sonderfall der in dieser Definition angegebenen globalen Transformation σ_N dar. Ursprünglich wurde die MAT von Blum /Blum (1967)/ als Modell eines sich propagierenden Prozesses, auch Grasfeuer (grassfire) genannt, zur Formbeschreibung (shape description) eingeführt und in /Blum und

Nagel (1977)/ weiterentwickelt. Sie kann auch zur Datenkompression herangezogen werden, wenn die zu codierenden Bilder große homogene Flächen aufweisen /Mott-Smith und Baer (1972)/.

Wir werden im nächsten Abschnitt die Datenkompression mit Hilfe der MAT erläutern und im darauf folgenden Abschnitt auf die Verbesserung dieser Datenkompression, insbesondere auf die vollständige Eliminierung der Redundanz eingehen.

2.4.1 Datenkompression mit der MAT

Die lokalen Maxima $W_H^{(k)}c$ und die entsprechenden Distanzwerte k stellen eine codierte Form des ursprünglichen Binärbildes c dar. Dilatiert man jedes $W_H^{(k)}c$ mit dem Raster H k-mal und verknüpft die jeweiligen Ergebnisse mit der logischen OR-Operation, so erhält man das Originalbild vollständig wieder. Die morphologischen Aspekte dieser Darstellung wollen wir zunächst zurückstellen und lediglich die durch diese Repräsentation mögliche Datenkompression als Proposition formulieren.

Proposition 2.3: Sei $c \in C_n$ ein Binärbild und $W_H^{(k)}c$ die lokalen Maxima der k-ten Generation $(k=1,2,\ldots,m)$. Die MAT führt zu einer Datenkompression genau dann wenn gilt:

$$n^2/(2\log_2 n.\sum_{k=1}^{m} \#W_H^{(k)}c+\log_2 m) > 1$$

Beweis: Die Speicherung der $\#W_H^{(k)}c$ lokalen Maxima einer k-ten Generation benötigt $2\log_2 n$ Bit für die Koordinaten eines jeden lokalen Maximums. Bei m Generationen werden zusätzlich $\log_2 m$ Bit zur Spezifikation der Generation benötigt.

Auf ihre Benutzung zur Datenkompression wurde die MAT in zahlreichen Arbeiten untersucht: In /Pfaltz und Rosenfeld (1967)/ lieferte die MAT vergleichbare Ergebnisse zur Konturcodierung. Die Untersuchungen von /Mott-Smith und Baer (1972)/ führten zu der Aussage, daß die mit Hilfe der MAT erzielte Datenkompression im allgemeinen nicht effizient ist. Nur in Verbindung mit der Bildanalyse kann die MAT auch zur Datenkompression herangezogen werden. Die in /Märgner und Zamperoni (1977)/ durchgeführten Experimente bestätigten trotz einiger Versuche zur Verbesserung der Datenreduktion diese Aussage.

2.4.2 Verbesserung der Datenkompression mit der MAT

Die Nachteile der MAT im Hinblick auf die Datenkompression können zum Teil behoben werden, wenn zusätzliche Algorithmenkomplexität in Kauf genommen werden kann. Ansätze zur Verbesserung der Datenkompression von der MAT bilden unter anderem: -i- die Eliminierung der Redundanz, -ii- die Unterdrückung der lokalen Maxima mit dem Distanzwert unter einer wählbaren Schwelle, und -iii- die weitere Codierung der lokalen Maxima.

-i- Eliminierung der Redundanz

Zur Eliminierung der Redundanz wurde in /Märgner und Zamperoni (1977)/ festgestellt, daß zwischen je 2k benachbarten lokalen Maxima mit dem Distanzwert k bezüglich des Moore-Rasters genau 2(k-1) lokale Maxima redundant sind. Diese Aussage wollen wir auf beliebige Folgen von lokalen Maxima bezüglich des Moore- und des von Neumann-Rasters im folgenden Satz erweitern. Dabei bezeichne $\lfloor x \rfloor$ die größte ganze Zahl kleiner oder gleich x.

Satz 2.1: Sei $c \in C_n$ ein Binärbild, $W_M^{(k)}c$ und $W_H^{(k)}c$ die lokalen Maxima der k-ten ($k > 1$) Generation bezüglich des Moore- und von

Neumann-Rasters respektive. Dann sind in $W_M^{(k)}c$ in jeder horizontalen (bzw. vertikalen) Folge von $m > 2$ benachbarten lokalen Maxima genau $m-(\lfloor(m-2)/(2k-1)\rfloor+2)$ redundant. Analog sind in $W_H^{(k)}c$ in jeder 45^o- (bzw. 135^o-) diagonalen Folge von $m > 2$ benachbarten lokalen Maxima genau $m-(\lfloor(m-2)/(k-1)\rfloor+2)$ redundant.

Beweis: Die redundanten lokalen Maxima werden analog zu der in der Beweisskizze von Prop. 2.2 beschriebenen Vorgehensweise ermittelt.

Bemerkungen:

a) Diese vollständige Eliminierung der Redundanz ist leicht zu implementieren und relativ schnell durchzuführen im Gegensatz zur Vermutung von /Rosenfeld und Kak (1982)/ (Seite 195).

b) Für den allgemeineren Fall von Wurzelzellen $W_N^{(k)}c$ muß eine entsprechende Aussage noch formuliert werden, zumindest für konvexe Raster N.

-ii- Unterdrückung der lokalen Maxima

Die Unterdrückung von lokalen Maxima bedeutet eine Vernachlässigung unwesentlicher Bildteile und führt somit zu einer nicht exakten Rekonstruktion des Originalbildes. Ergebnisse diesbezüglich wurden in /Märgner und Zamperoni (1977) und Zamperoni (1980)/ dargestellt.

-iii- Weitere Codierung der lokalen Maxima

Über eine weitere Codierung der lokalen Maxima mit dem gleichen Distanzwert mit Hilfe der Konturcodierung wurde in /Zamperoni (1980)/ berichtet. Eine ähnliche Vorgehensweise wird in /Frank et al. (1980)/ vorgeschlagen, wobei die lokalen Maxima mit dem gleichen Distanzwert zu sogenannten Saatpunkten (seed points) reduziert werden. Saatpunkte selbst sind lokale Maxima bezüglich eines weiteren Rasters.

3. Codierung mit mehreren Rastern

Im vorigen Kapitel wurde die Mittelachsentransformation (MAT) als Beispiel einer globalen Transformation im Sinne von Definition 2.3 angegeben. Aus der Literatur ist bekannt, daß die MAT abgesehen von ihren morphologischen Eigenschaften und deren Anwendung in der Form- und Gestaltbeschreibung /Blum (1964)/, für die Datenkompression im allgemeinen nicht besonders effizient ist. Wir wollen in diesem Kapitel die Frage untersuchen, ob mit Hilfe von weiteren Rastern globale Transformationen gefunden werden können, die eine effizientere Datenkompression erzielen.

Vom geometrischen Gesichtspunkt her bedeutet die Anwendung der globalen Transformation $W_N^{(k)}$ von Definition 2.3 auf ein Binärbild c die Darstellung von c als Vereinigung von Mustern kN. Jedes Muster wird aus dem entsprechenden Element des Trägers von $W_N^{(k)}$ c , der Wurzelzelle des Musters, erzeugt. Die Wahl eines Rasters N, das die Form der Muster kN festlegt und zu einer effizienteren Datenkompression führt, ist nicht trivial. Da man im allgemeinen davon ausgeht, daß keine a priori Information über das zu codierende Bild vorliegt, wird es vom Gesichtspunkt der Datenkompression her keinen Grund geben, aus einer vorgegebenen endlichen Menge von Rastern eines zu bevorzugen. Daß es in dieser Menge von Rastern mindestens eines gibt, mit dem der höchste Kompressionsfaktor erzielt werden kann, ist offensichtlich. Da bereits in einer kleinen Nachbarschaft etwa von rxs-Bildpunkten ($r,s \in \mathbb{N}$), die Anzahl der unterschiedlichen Muster 2^{rs} beträgt, ist das systematische Suchen eines optimalen Rasters im allgemeinen nicht durchführbar. Vielmehr werden a priori Informationen über das Bildmaterial vorausgesetzt, und infogedessen ist eine Eingrenzung der infrage kommenden Raster möglich und auch

sinnvoll. Eine Verbesserung des mit einem optimalen Raster erzielten Kompressionsfaktors kann mit einer weiteren Codierung von Wurzelzellen derselben Generation bezüglich dieses Rasters erfolgen. Eine heuristische Methode, diese Verbesserung zu erzielen, kann darin bestehen, für jede Konfiguration von Wurzelzellen wiederum ein optimales Raster zu bestimmen.

Die Codierung eines Binärbildes mit einem optimalen Raster und die anschließende Verbesserung des Kompressionsfaktors werden im ersten Abschnitt dieses Kapitels behandelt. Der zweite Abschnitt stellt die Komplexität des allgemeineren Codierungsproblems dar und unterstreicht die Bedeutung der heuristischen Strategien bei solchen Suchproblemen.

Ein weiterer, auch auf mehreren Rastern beruhender Ansatz besteht darin, eine endliche Menge von Rastern, die wir atomare Raster nennen, festzulegen und Muster im Binärbild zu suchen, die als lineare Kombination im Sinne der Minkowski'schen Summe /Hadwiger 1950/ von diesen atomaren Rastern dargestellt werden können. Man kann sich leicht klarmachen, daß die Anzahl der möglichen Muster mit der Anzahl der atomaren Raster exponentiell steigt und somit das Berücksichtigen aller Kombinationen ausschließt. Vielmehr werden sich auch hier Suchstrategien empfehlen, die zwischen dem erzielten Kompressionsfaktor und dem dafür aufgebrachten Suchaufwand ihr Optimum finden. Jede lineare Kombination der atomaren Raster besitzt eine Wurzelzelle, aus der das zugehörige Muster erzeugt werden kann. Auch hier kann die Datenkompression verbessert werden, wenn die Wurzelzellen derselben Generation mit einem weiteren Raster codiert werden. Mit diesem Ansatz beschäftigt sich der zweite Abschnitt dieses Kapitels. Im letzten Abschnitt dieses Kapitels wollen wir die beiden Ansätze im Hinblick auf die erzielten Kompressionsfaktoren miteinander vergleichen.

3.1 Codierung mit Hilfe eines optimalen Rasters: Eine heuristische Methode

Nach Proposition 2.1 kann ein Binärbild c mit Hilfe seiner Wurzelzellen bezüglich eines beliebigen Rasters codiert werden. Aus diesen Wurzelzellen und dem dazugehörigen Raster kann c vollständig rekonstruiert werden. Als Beispiel einer Transformation zur Berechnung der Wurzelzellen gemäß Definition 2.3 wurde im Abschnitt 2.2 die für die Kompression von c bekanntlich nicht besonders effiziente MAT angegeben. Es ist naheliegend, die Frage zu stellen, ob mit einer endlichen Menge von paarweise verschiedenen Rastern eine bessere Effizienz der Kompression von c erzielt werden kann. Nach welchen Kriterien diese Menge zu wählen ist, obliegt einer Heuristik, die auf a priori Informationen über das zu codierende Bild basiert. Eine heuristische Methode zur Untersuchung dieser Frage wird im folgenden Abschnitt dargestellt. Anschließend wird ein Modell zur Verallgemeinerung der Methode beschrieben.

Die triviale Aussage, daß es in einer gegebenen endlichen Menge von Rastern ein Raster gibt, mit dem der bestmögliche Kompressionsfaktor erzielt werden kann, wollen wir als Satz formulieren.

Satz 3.1: Sei $c \in C_n$ ein Binärbild, ζ eine endliche Menge von Rastern und $W_N^{(k)}c$ eine Konfiguration von Wurzelzellen der k-ten Generation (k=1,2,...) bezüglich eines Rasters $N \in \zeta$. Dann existiert ein Raster $N^* \in \zeta$, so daß für alle $N \in \zeta$ gilt:

$$\sum_k \#W_{N^*}^{(k)}c \leq \sum_k \#W_N^{(k)}c$$

Ein Raster $N^* \in \zeta$, das die obige Bedingung erfüllt, wird optimales Raster (bezogen auf die Menge ζ) genannt.

In Tabelle 3.1 sind eine Menge ζ von 16 Rastern und die dazugehörigen Kompressionsfaktoren für das Binärbild von Fig. I mit 256x256 Pixels aufgezeichnet. Daß das zweite Raster in der ersten Spalte der Tabelle den besten Kompressionsfaktor erzielt, ist bei diesem Bild nicht unbedingt im voraus zu erkennen. Es sei dabei zu bemerken, daß die MAT bezüglich des Moore-Rasters unter den hier erprobten Rastern am schlechtesten abgeschnitten hat (vgl. 2-tes Raster in der 7-ten Zeile der Tabelle 3.1).

Raster	Kompressions-faktor	Raster	Kompressions-faktor
**	4.98	**	5.5
* *	5.6	** **	4.64
* *	3.65	*** ***	4.0
* *	3.89	** *** **	3.68
* **	4.67	** *** **	3.68
** *	4.8	* *** *	4.24
* **	5.17	*** *** ***	3.4
** *	4.88	** **** **** **	3.7

Tabelle 3.1: Einige Raster und die zugehörigen Kompressionsfaktoren für das Bild von Fig.I

3.1.1 Verbesserung des Kompressionsfaktors eines optimalen Rasters

Der mit dem optimalen Raster $N^* \in \zeta$ erzielte Kompressionsfaktor kann im allgemeinen verbessert werden, wenn die Wurzelzellen $W^{(k)}_{N^*}c$ jeder k-ten Generation im Binärbild c nach dem selben Schema (d.h. zu jeder Generation von Wurzelzellen wird ein optimales Raster gesucht) mit Hilfe weiterer Raster aus ζ codiert werden. Zur weiteren Codierung der Wurzelzellen können im Prinzip alle Raster von ζ herangezogen werden, die N^* nicht enthalten; denn solche können keine weitere Datenkompression mehr ermöglichen. Als Modell für diese Codierungsschritte dient die Baumstruktur von Fig.3.1. Dabei haben die Komponenten des Baumes folgende Bedeutung:

- Die Wurzel des Baumes stellt die Konfiguration c dar;
- Die Kanten $W^{(1)}_{N^*}, \ldots, W^{(k)}_{N^*}$ verbinden die Wurzel c mit ihren Nachfolgern. Jeder i-te dieser Nachfolger wird wiederum mit seinen Nachfolgern durch $W^{(1)}_{N_i^*}, \ldots, W^{(l_i)}_{N_i^*}$ verbunden. Das bedeutet: Jede i-te Generation von Wurzelzellen wird mit Hilfe eines für sie optimalen Rasters N_i^* codiert.
- Jede Kante $W^{(i)}_{N^*}$ bzw. $W^{(l_i)}_{N_i^*}$ ist auf einen Knoten $W^{(i)}_{N^*}c$ bzw. $W^{(l_i)}_{N_i^*}c$ gerichtet.

Diese weitere Codierung wollen wir auch in Form eines Satzes angeben.

Satz 3.2: Sei c ein Binärbild, ζ eine endliche Menge von Rastern N_j, $j \in I \subset \mathbb{N}$, und $W_N{}^{(i)}c$ die Wurzelzellen der i-ten Generation bezüglich N. Ferner seien N^* und $N_i{}^*$ optimale Raster (im Sinne von

Satz 3.1) zur Codierung von c bzw. $W^{(i)}_{N^*}c$. Dann gilt

$$\sum_i \sum_{k_i} \#W^{(k_i)}_{N_i^*}(W^{(i)}_{N^*}c) \leq \sum_i \#W^{(i)}_{N^*}c \qquad (k_i \in \mathbb{N})$$

Beweis: trivial

Aus Tabelle 3.2 kann für $\zeta = \{ **, \overset{*}{*}, *\,{}^{*}, {}^{*}\,*, *\overset{*}{*}*, \overset{*}{*}\,\overset{*}{*} \}$ und $N^* = \{ \overset{*}{*} \}$ das Ergebnis einer solchen weiteren Codierung angewandt auf das Bild von Fig.I aus der ersten Spalte entnommen werden. Dabei wurden zunächst alle Generationen von Wurzelzellen bezüglich des für das Bild optimalen Rasters $\overset{*}{*}$ (vgl. Tabelle 3.1) bestimmt, und anschließend wurde jede Generation von Wurzelzellen mit dem aus ζ für sie optimalen Raster codiert. Die Kompressionsfaktoren der Spalten 1 bis 6 wurden ebenfalls nach diesem Verfahren errechnet. Dabei wurde jeweils das Raster der 1. Zeile und i-ten Spalte als optimales Raster angenommen. Man beachte, daß der Kompressionsfaktor der vorletzten Spalte das Ergebnis einer weiteren Codierung der lokalen Maxima (Wurzelzellen bez. des von Neumann-Rasters H) darstellt.

Optimales Raster	* *	* *	* *	*	* * * *	* * * *
Kompressions-faktor	7.12	6.90	6.11	5.79	6.47	7.70

Tabelle 3.2: Bestimmung der Wurzelzellen für jedes der Raster aus der Menge ζ (erste Zeile der Tabelle) und weitere Codierung der Wurzelzellen mit einem für sie optimalen Raster aus ζ.

Der Rechenaufwand zur Verbesserung des Kompressionsfaktors eines optimalen Rasters wird im folgenden Satz als die Anzahl der dazu notwendigen Vergleiche der zusätzlich zu berechnenden Konfigurationen von Wurzelzellen formuliert.

Satz 3.3: Sei c ein Binärbild und ζ eine endliche Menge von m Rastern. Zur Codierung von c und dessen k_j Nachfolger in jeder j-ten Knotengeneration des Baumes von Fig. 3.1a mit dem jeweiligen optimalen Raster aus ζ sind

$$a_j = \sum_{i=0}^{j} (m-i)k_i \qquad (1)$$

Vergleiche von Wurzelzellen notwendig, wobei $0 \leq j \leq m$ und $k_0 = 1$

Beweis: (per Induktion über j)

1) Für den trivialen Fall $j = 0$ ist $a_0 = mk_0 = m$. Denn es müssen m Vergleiche durchgeführt werden, um aus ζ das optimale Raster zur Codierung von c zu bestimmen.

2) Angenommen (1) gilt für j-1. Dann entstehen k_j neue Konfigurationen, die als Knoten der j-ten Generation im Baum von Fig. 3.1a dargestellt werden. Für jede dieser Konfigurationen ist ein optimales Raster aus (m-j) unterschiedlichen, bis zur Entstehung dieser Konfigurationen noch nicht benutzten Rastern aus ζ zu bestimmen. Dazu sind dann $k_j(m-j)$ zusätzliche Vergleiche notwendig. Daraus folgt :

$$a_j = a_{j-1} + (m-j)k_j = \sum_{i=0}^{j-1} (m-i)k_i + (m-j)k_j = \sum_{i=0}^{j} (m-i)k_i$$

und somit die Behauptung.

Die Komplexität des Suchalgorithmus zur Durchführung der a_j Vergleiche von Satz 3.3 wird durch die 'Dichte' des Suchbaumes von Fig. 3.1a bestimmt. Entscheidend dabei ist die Anzahl der Knoten im Baum, die wiederum vom Verhältnis der Größe der Objekte im Bild zu der Größe der Raster von ζ abhängig ist. Die Anzahl der Knoten je Knotengeneration im Suchbaum von Fig. 3.1a wird im folgenden Korollar explizit angegeben.

Korollar 3.1: Seien c, ζ , m und k_j (j=1,...,m-1) wie in Satz 3.3. Ferner sei k_{ji} die Anzahl der nichtleeren Wurzelzellen aller Generationen der k_j Konfigurationen (der j-ten Knotengeneration) bezüglich der optimalen Raster N^*_{j1}, ... , $N^*_{j,k_{j-1}}$ respektiv (vgl. Fig. 3.1a). Dann gilt :

$$k_j = \sum_{i=1}^{n} k_{ji}$$

wobei $n = k_{j-1}$.

Beweis: Der Beweis folgt unmittelbar aus Satz 3.3 und Fig. 3.1a.

Die Codierung eines Binärbildes mit einem optimalen Raster und die mit einer weiteren Codierung der Wurzelzellen mögliche Verbesserung

des Kompressionsfaktors führen nicht immer zu einer optimalen Kompression des Bildes. Dies sei am folgenden Beispiel erläutert:

Beispiel 3.1: Seien $N_1 = \begin{smallmatrix} & * \\ * & * \end{smallmatrix}$ und $N_2 = \begin{smallmatrix} * & \\ * & * \\ * & \end{smallmatrix}$ zwei Raster und c die in der ersten Spalte der Tabelle 3.3 aufgezeichneten Konfiguration. Dabei bedeuten '*' und '.' Punkte des Trägers der Konfiguration und des Hintergrundes respektive. Das optimale Raster ist hier N_2, da $\#W_{N_1}^{(1)}c + \#W_{N_1}^{(2)}c = 8$ und $\#W_{N_2}^{(1)}c + \#W_{N_2}^{(1)}c + \#W_{N_2}^{(2)}c = 4$. Bei einer weiteren Codierung der Wurzelzellen mit den Rastern N_1 und N_2 erweist sich jedoch die Wahl von N_1 als optimales Raster günstiger, denn $\#W_{N_2}^{(1)}(W_{N_1}^{(2)}c) = 3$ und $\#W_{N_1}^{(1)}(W_{N_2}^{(2)}c) = 4$.

c	$W_{N_1}^{(1)}c$	$W_{N_1}^{(2)}c$	$W_{N_2}^{(1)}(W_{N_1}^{(2)}c)$
**	..	..	..
***	...	*..	...
****	....	**..	*...
****	....	**..	*...
***	...	**.	*..
**	..	*.	..

	$W_{N_2}^{(1)}c$	$W_{N_2}^{(2)}c$	$W_{N_2}^{(3)}c = W_{N_1}^{(1)}(W_{N_2}^{(3)}c)$
	..	..	..
	...	...	...
	....	....	**..
	....	....	**..
	...	...	...
	..	..	..

Tabelle 3.3: Codierung von c mit N_1 (bzw. N_2) als optimales Raster und weitere Codierung der Wurzelzellen mit N_2 (bzw. N_1).

Ein weiteres Beispiel dafür, daß ein optimales Raster nicht immer zur optimalen Gesamtkompression führt, kann für das Bild von Fig.I aus der 1. und 6. Spalte der Tabelle 3.2 entnommen werden. Obwohl das Raster $\begin{smallmatrix}*\\*\end{smallmatrix}$ sich in einem unmittelbaren Vergleich der Raster (ohne weitere Codierung der Wurzelzellen) zunächst als optimal erwiesen hat (vgl. Tabelle 3.1), ist bei einer weiteren Codierung das Raster $\begin{smallmatrix}* & *\\ * & *\end{smallmatrix}$ noch effizienter.

Eine Codierung nach Satz 3.3 stellt lediglich eine heuristische Strategie dar, die keineswegs als eine optimale Lösung des allgemeineren, im nächsten Abschnitt formulierten Codierungsproblems betrachtet werden kann.

3.1.2 Verallgemeinerte optimale Raster

Das allgemeinere Codierungsproblem eines Binärbildes mit mehreren Rastern kann folgendermaßen formuliert werden:
Sei c ein Binärbild und ζ eine endliche Menge von Rastern N_i, wobei $i \in I \subset \mathbb{N}$. Zunächst werden alle nichtleeren Wurzelzellen

$$W^{(1)}_{N_{i_1}} c, \quad W^{(2)}_{N_{i_1}} c, \quad \ldots$$

bezüglich eines Rasters N_{i_1}, $i_1 \in I$ bestimmt. Dann werden wiederum alle nichtleeren Wurzelzellen der Konfiguration $W^{(1)}_{N_{i_1}} c$ bezüglich eines Rasters N_{i_2}, $i_2 \in I \setminus \{i_1\}$, die von $W^{(1)}_{N_{i_2}}$ $(W^{(1)}_{N_{i_1}} c)$ bezüglich eines Rasters N_{i_3}, $i_3 \in I \setminus \{i_1, i_2\}$, usf ... bestimmt. Es stellt sich dann die Frage, wie diese Raster N_{i_j} zu wählen sind, so daß ein optimaler Kompressionsfaktor von c mit Hilfe von ζ erzielt werden kann.

Als Modell zur Darstellung dieses Codierungsproblems eignet sich eine Baumstruktur mit folgender Bedeutung:
- Wurzel des Baumes : die Konfiguration c
- Kanten und Knoten des Baumes : c wird mit seinen Nachfolgern

$W^{(1)}_{N_{i_1}}c$, $W^{(2)}_{N_{i_1}}c$, ... durch die Kanten $W^{(1)}_{N_{i_1}}$, $W^{(2)}_{N_{i_1}}$, ... verbunden. Jeder weitere Knoten q, der mit c durch einen Pfad $W^{(k_1)}_{N_{i_1}}$, $W^{(k_2)}_{N_{i_2}}$,..., $W^{(k_m)}_{N_{i_m}}$ verbunden ist, wird mit seinem Nachfolger $W^{(k_{m+1})}_{N_{i_{m+1}}}q$ durch $W^{(k_{m+1})}_{N_{i_{m+1}}}$ verbunden.

Ein Ausschnitt der ersten beiden Nachfolgergenerationen eines solchen Baumes wird in Fig. 3.2 dargestellt.

Bei der Suche nach einer optimalen Sequenz von Rastern für das hier gestellte Codierungsproblem muß zusätzlich noch die Reihenfolge der Anwendung der Raster berücksichtigt werden; denn es gilt:

Satz 3.4: Sei c ein Binärbild, ς eine endliche Menge von Rastern und $W^{(k)}_N c$ die Wurzelzellen der Generationen k=1,2,... Dann existieren Raster N_1, N_2 so daß gilt:

$$\sum_l \sum_k \#W^{(l)}_{N_1}(W^{(k)}_{N_2}c) \quad \neq \quad \sum_j \sum_i \#W^{(j)}_{N_2}(W^{(i)}_{N_1}c)$$

Beweis: Anstelle eines formalen Beweises wollen wir ein Beispiel einer Konfiguration c und Raster N_1 , N_2 angeben, die der Aussage dieses Satzes genügen. Man betrachte die Raster $N_1 = \begin{matrix} * \\ ** \\ * \end{matrix}$ und $N_2 = \begin{matrix} ** \\ ** \end{matrix}$ und die in der ersten Spalte von Tabelle 3.4 angegebene Konfiguration c. In den Spalten 2 bis 6 sind die Wurzelzellen aller Generationen bezüglich beider Raster aufgezeichnet. In diesem Beispiel ist

$\sum_{l=1}^{2}\sum_{k=1}^{3} \#W_{N_1}^{(l)}(W_{N_2}^{(k)}c) = 1$, während $\sum_{j=1}^{1}\sum_{i=1}^{3} \#W_{N_2}^{(j)}(W_{N_1}^{(i)}c) = 4$ ist.

c	$W_{N_2}^{(1)}c$	...	$W_{N_2}^{(3)}c$	$W_{N_1}^{(1)}(W_{N_2}^{(3)}c)$	$W_{N_1}^{(2)}(W_{N_2}^{(3)}c)$
***	...	...	...	...	...
****	....	....	....	....	....
****	....	....	...*	....	....
****	....	....	..**	....	...*
***	...	...	..*	...	...

	$W_{N_1}^{(1)}c$	...	$W_{N_1}^{(3)}c$	$W_{N_2}^{(1)}(W_{N_1}^{(3)}c)$	$W_{N_2}^{(2)}(W_{N_1}^{(3)}c)$
	...	...	...	...	...
	....	.*..	....	....	....
	....	....	..**	..**	..**
	....	...*	....	....	....
	...	...	...	...	...

Tabelle 3.4: Abhängigkeit der Wurzelzellen von der Reihenfolge der verwendeten Raster.

Der zur Lösung des hier beschriebenen Codierungsproblems resultierende Rechenaufwand wird im folgenden Satz formuliert.

Satz 3.5: Sei c ein Binärbild und ζ eine endliche Menge von $m > 0$ Rastern. Zur Codierung con c und dessen k_j Nachfolgern in jeder j-ten Knotengeneration des Baumes von Fig. 3.2a mit allen Rastern von ζ sind

$$a_j = \sum_{i=0}^{j} (m-i)k_i$$

Vergleiche von Wurzelzellen notwendig, wobei $0 \leq j \leq m$ und $k_0 = 1$.

Beweis: Die Beweiskonstruktion ist analog zu der von Satz 3.3.

Die Komplexität des Suchalgorithmus zu obigem Satz wird auch hier in Form eines Korollars ausgedrückt.

Korollar 3.2: Seien c, ς, m und k_j (j=1,...,m-1) wie in Satz 3.5. Ferner sei k_{ji} die Anzahl der nichtleeren Wurzelzellen aller Generationen der k_j Konfigurationen (der j-ten Knotengeneration) bezüglich der Mengen von Rastern $\{N_j, \ldots, N_m\}$, $\{N_{j-1}, N_{j+1}, \ldots, N_m\}$, ..., $\{N_1, \ldots, N_{m-j+1}\}$ respektiv (vgl. Fig. 3.2a). Dann gilt:

$$k_j = \sum_{i=1}^{n} k_{ji}$$

wobei $n = (m-(j-1))k_{j-1}$.

Beweis: Der Beweis folgt unmittelbar aus Satz 3.5 und Fig. 3.2a.

Bemerkung: Um sich eine Vorstellung über den Unterschied in der Komplexität der Suchalgorithmen von Satz 3.3 und Satz 3.5 zu verschaffen, sind die 'Dichten' der Suchbäume von Fig. 3.1a und Fig. 3.2a näher zu betrachten und miteinander zu vergleichen.

Angesichts des im allgemeinen großen Aufwandes, bei einer gegebenen Menge von Rastern alle möglichen Wurzelzellen von Satz 3.5 zu berechnen, ist eine systematische Betrachtung aller Kombinationen impraktikabel. Infogedessen sind heuristische Strategien wie etwa die im vorigen Abschnitt eingeführte Strategie, ein Binärbild durch seine codierten Wurzelzellen bezüglich eines optimalen Rasters darzustellen, für derartige Suchprobleme notwendig.

Wir werden im nächsten Abschnitt das Codierungsproblem eines Binärbildes mit Hilfe mehrerer Raster von einem weiteren Aspekt her betrachten und auch dafür heuristische Strategien vorschlagen.

3.2 Codierung mit Hilfe einer optimalen Sequenz von Rastern: Eine heuristische Methode

Ein weiterer Ansatz, das Codierungsproblem eines Binärbildes mit einer endlichen Menge von Rastern heuristisch zu behandeln, besteht darin, im Gegensatz zur bisherigen Beibehaltung desselben optimalen Rasters für alle Generationen von Wurzelzellen (vgl. Abschnitt 3.1) die Möglichkeit eines Wechsels des Rasters nach jeder ermittelten Generation einzubeziehen. Die dabei resultierenden Wurzelzellen werden dann bezüglich Sequenzen von Rastern definiert. Nur im Spezialfall einer aus demselben Raster bestehenden Sequenz stimmen die Wurzelzellen hier mit denen des optimalen Rasters von Abschnitt 3.1 überein. Die Wahl der Raster erfolgt auch hier heuristisch. Von jeder Wurzelzelle einer k-ten Generation wächst ein Muster als (Minkowski' sche) Summe der zur Entstehung dieser Zelle geführten Sequenz von k Rastern. Aufgrund dieses Wachstumsprozesses wollen wir für die Menge der benutzten Raster die Bezeichnung atomare Raster einführen. Wir werden in diesem Abschnitt eine allgemeine Beschreibung dieses Ansatzes mit Hilfe einer Baumstruktur als Modell angeben. Anschließend werden wir eine heuristische Suchstrategie und einige mit ihr erzielten Ergebnisse sowie deren Verbesserung vorstellen.

3.2.1 Verallgemeinerte optimale Sequenz von Rastern

Zunächst wollen wir die in Def. 2.2 und 2.3 eingeführten Transformationen bezüglich eines Rasters auf eine Sequenz von Rastern erweitern.

Definition 3.1: Seien $c \in C_n$ ein Binärbild, $\zeta = \{ N_i : i \in I \subset \mathbb{N} \}$ eine endliche Menge von atomaren Rastern und S_k eine Sequenz von k Rastern N_{i_j}, $i_j \in I$. Eine globale Transformation t_{S_k} (bezüglich der Sequenz S_k) wird folgendermaßen definiert :

$$t_{S_0}(c) = c$$
$$t_{S_k}(c) = t_{N_{i_k}}(t_{S_{k-1}}(c)) \qquad (k > 0)$$

Bemerkung: In der Schreibweise $t_{S_k}(c)$ werden wir gelegentlich die Klammern weglassen.

Definition 3.2: Seien $c \in C_n$ ein Binärbild, E_{S_k} eine Erosion (bez. einer Sequenz S_k) und O_N eine Ouverture (bez. eines Rasters N) von c. Globale Transformationen γ_{S_k} $(k > 0)$ und W_{S_k} werden folgendermaßen definiert :

$$\gamma_{S_k}(c) = O_{N_{i_k}}(E_{S_{k-1}}(c))$$

$$W_{S_k}\, c(u) = \begin{cases} 1 & \text{falls } \gamma_{S_k} c(u)=0 \text{ und } E_{S_{k-1}} c(u)=1 \\ 0 & \text{sonst} \end{cases}$$

Bemerkung: Anstelle von t_{S_k} können wir auch t_{P_k} schreiben, wobei

$$P_k = \sum_{j=1}^{k} N_{i_j} \qquad (i_j \in I)$$

die (Minkowski' sche) Summe der k Raster der Sequenz darstellt. Wir bevorzugen jedoch die Schreibweise t_{S_k}, um den Charakter der Sequenz hervorzuheben, denn bei der Implementierung der globalen Transformation t_{S_k} werden die k Raster und nicht etwa deren Minkowski' sche Summe verwendet. Vorteile solcher Zerlegungen von Rastern kann man sich leicht klarmachen, wenn man bedenkt, daß die Anzahl der benötigten Operationen bei der Implementierung von t_P immer größer oder gleich denen - im allgemeinen sogar wesentlich größer als die -, die benötigt werden, wenn t_{S_k} verwendet wird. Diese Eigenschaft wird in /Serra (1969), Pecht (1985)/ benutzt, um Minkowski' sche Operationen zu beschleunigen.

3.2.1.1 Ein Suchbaum

Mit Hilfe dieser Notation können wir nun den Ansatz zur Codierung eines Binärbildes mit einer Sequenz von Rastern näher erläutern. Als Modell wollen wir auch hier eine Baumstruktur wählen. Die Komponenten des Baumes erhalten dabei folgende Bedeutung :

- Wurzel des Baumes : die Konfiguration c
- Kanten des Baumes : Jeder Knoten q ist mit seinem i-ten Nachfolger durch die Transformation $E_{N_i} q$ als Kante verbunden. Das bedeutet: Die Konfiguration c wird mit den Rastern N_1, N_2,..., N_m erodiert. Die Ergebnisse dieser Erosionen werden als Nachfolger von c aufgefaßt, die wiederum mit N_1,...,N_m erodiert werden, und so fort...
- Knoten der ersten Generation : die erodierten Konfigurationen $E_{N_1} c$, $E_{N_2} c$, ... von c bezüglich N_1, N_2,...
- Knoten der k-ten Generation : Die mit einer Sequenz $S_k=(N_{i_1},\ldots, N_{i_k})$ von Rastern sukzessiv erodierte Konfiguration c. Diese Konfiguration wird gemäß Def. 3.1 $E_{S_k} c$ bezeichnet. Dabei wird von einer Sequenz S_k ein Pfad von k erodierten Konfigurationen
$$E_{N_{i_1}} c \;,\; E_{N_{i_2}} c \;,\; \ldots \;,\; E_{N_{i_k}} c$$
festgelegt, der einen Knoten der k-ten Generation mit der Wurzel des Baumes verbindet.

In Fig.3.3 wird ein solcher Baum dargestellt. Dabei werden zur Erläuterung einige Sequenzen S_k explizit ausgeschrieben.

Jedem Knoten $E_{S_k} c$ im Baum von Fig. 3.3 wird gemäß Definition 3.2 eine Konfiguration von Wurzelzellen $W_{S_k} c$ zugeordnet. In der Tat läßt

sich $W_{S_k}\, c$ aus dem Vorgänger von $E_{S_k}\, c$ und dem - in der auf E_{S_k} gerichteten Kante $E_{N_{i_k}}$ benutzten - Raster N_{i_k} per Definition ermitteln. Somit kann jeder Pfad, der die Wurzel c des Baumes mit einem beliebigen Knoten verbindet, als ein Code von c betrachtet werden. Der Pfad der Länge 0 ist trivialerweise das Bild c selbst.

Mit der Baumstruktur von Fig. 3.3 als Modell besteht nun die Codierung eines Binärbildes c mit Hilfe einer endlichen Menge von Rastern darin, einen optimalen Pfad von Erosionsschritten, oder äquivalent dazu eine Sequenz von Rastern zu finden, die zum bestmöglichen Kompressionsfaktor von c führen. Bei einer Menge von m Rastern und einer Baumtiefe von k Generationen ist ein solcher Pfad unter den maximal m^k möglichen Pfaden zu ermitteln. Bei derartigen Suchproblemen definiert man im allgemeinen eine sogenannte Nutzfunktion (auch Kostenfunktion genannt) /Nilsson 1982/, die eine Gewichtung der Knoten des Suchbaumes darstellt, und man sucht dann entlang eines Pfades ein Optimum dieser Funktion. In unserem Fall werden wir die Knoten mit der jeweiligen Anzahl der Wurzelzellen gemäß folgender Definition gewichten.

Definition 3.3: Sei $S_k = (N_{i_1},\ldots,N_{i_k})$ die einem Pfad $E_{S_k}\,c$ des Baumes von Fig.3.4 zugeordnete Sequenz von Rastern aus ζ . Eine Nutzfunktion eines Pfades $E_{S_k}\,c$ wird mit $\varkappa(E_{S_k}\,c)$ bezeichnet und folgendermaßen definiert :

$$\varkappa(E_{S_k}\,c) = \sum_{i=1}^{k} \#W_{S_i}\,c$$

Ein Pfad $E_{S_k}^{*}\,c$ heißt optimal, wenn gilt :

$$\varkappa(E_{S_k}^{*}\,c) = \min\{\,\varkappa(E_{S_k}\,c) \quad \text{für alle } S_k\,\}\ .$$

Die Berechnung der Nutzfunktion für alle Pfade des Suchbaumes ist nicht praktikabel. Vielmehr werden heuristische Suchstrategien herangezogen, die, auch wenn sie das Finden eines optimalen Pfades nicht garantieren, zumindest eine zufriedenstellende Kombination des Nutzens des Pfades, in unserem Fall des erzielten Kompressionsfaktors, und des dafür erforderlichen Suchaufwandes beschaffen. Wir werden im nächsten Abschnitt eine solche Strategie beschreiben.

3.2.1.2 Eine gleichmäßige Suchstrategie

In der Heuristik der Suchstrategien unterscheidet man in der Regel zwischen einem Suchen mit (a priori) Information (informed search) und einem Suchen ohne Information (uninformed search) /Nilsson 1982/. Während bei Algorithmen des informierten Suchens die Entscheidungen aufgrund der vorhandenen Information und der Zwischenergebnisse getroffen werden, sind nicht informierte Algorithmen durch festgelegte Entscheidungsregeln charakterisiert , von denen sie im gesamten Ablauf begleitet werden. Wir werden hier ausschließlich nicht informierte Algorithmen betrachten und gehen davon aus, daß alle Pfade mit der gleichen Wahrscheinlichkeit optimal sein können.

Eine heuristische Methode, sich einen optimalen Pfad im Suchbaum anzunähern, besteht darin, die Nutzfunktion in Unterbäumen des gesamten Suchbaumes zu berechnen und die Konkatenation von den Anfangsteilen der resultierenden optimalen Pfade in den jeweiligen Unterbäumen zum optimalen Pfad der Strategie, d.h. des gesamten Suchbaumes, zu deklarieren. Wenn die Unterbäume gleich sind, bezeichnet man ein solches Suchen als gleichmäßig /Nilsson (1982)/. Eine gleichmäßige Suchstrategie wollen wir im folgenden näher betrachten.

Es bezeichnen T den Suchbaum von Fig. 3.3, d dessen Tiefe, T_1 einen Unterbaum von T mit derselben Wurzel c und einer Tiefe $d_1 < d$. d_1 ist so zu wählen, daß der Aufwand für die Berechnung aller Nutzfunktionen der Pfade von T_1 durchführbar ist. Es sei $E^*_{S_d}$ c ein Pfad in T_1, so daß die Nutzfunktion $\varkappa(E^*_{S_d} c)$ minimal ist. Nach Def. 3.3 ist $E^*_{S_d}$ c ein optimaler Pfad in T_1. Die ersten $d' \leq d_1$ Knoten

(von der Wurzel aus gesehen) von $E^{*}_{S_d}$ c werden als d' erste Knoten des optimalen Pfades des gesamten Baumes selektiert. Der d'-te Knoten des Pfades $E^{*}_{S_d}$ c bildet nun die Wurzel des nächsten auszuwertenden Unterbaumes T_2. Die Tiefe von T_2 ist wieder d_1. Diese Vorgehensweise wird in Fig. 3.5 schematisch dargestellt. Das Suchverfahren wird so lange fortgesetzt, bis der letzte aufgenommene Knoten ein Blatt des Baumes T ist.

In Tabelle 3.5 sind einige Ergebnisse der Anwendung einer gleichmäßigen Suchstrategie mit einer Tiefe des Unterbaumes von $d_1=1$ auf das Bild von Fig.I. zusammengestellt. Die Wahl der atomaren Raster obliegt keiner weiteren Einschränkung, außer der in einer 3x3-Nachbarschaft enthalten zu sein. Ein Vergleich der hier erzielten Kompressionsfaktoren mit denen von Tabelle 3.1 zeigt, daß auch bei einem minimalen Suchaufwand das Kombinieren mehrerer Raster zu Kompressionsfaktoren führt, die i. allg. höher als die eines einzelnen Rasters sind. Daß dies nicht immer der Fall sein kann, belegt der Kompressionsfaktor in der letzten Zeile von Tabelle 3.5 verglichen mit den Kompressionsfaktoren in der ersten Zeile von Tabelle 3.1. Der Unterschied ist jedoch so geringfügig, daß angesichts der in den meisten Fällen vorteilhaften Datenkompression sich die Codierung mit mehreren Rastern bei einer effizienten Suchstrategie empfiehlt.

Atomare Raster	Kompressions-faktoren	Atomare Raster	Kompressions-faktoren
`  *` `**,*`	6.8	`    *` `**,*`	5.3
`  *` `**, *`	5.2	`* *` `*,*`	6.1
`* *` `*, *`	6.0	`* *` `* , *`	5.0
`  * *` `**,*,*`	7.3	`  * * *` `**,*,* , *`	7.9
`** * **  *` `*,**, *, **`	6.8	`**  * **` `* ,**, *`	6.2

`*  **`
`***, **`
`*`
	6.6	`***  **`
		`***, **`
		`***`
			5.5

`***  *`
`***, ***`
`***  *`
	4.4	`**  **`
		`***, ***`
		`**  **`
			5.1

`**` `**, **`	4.96		

Tabelle 3.5: Kompressionsfaktoren des Bildes von Fig.I erzielt mit der gleichmäßigen Suchstrategie der Tiefe d_1=1.

Durch die Parameter d_1 und d' läßt sich der für gleichmäßige Suchstrategien erforderliche Suchaufwand den vorhandenen Speicherkapazitäten und Rechenzeiten anpassen. Die beiden Extremsituationen treten dann auf, wenn d_1 = d für den maximalen Suchaufwand und d_1 = d' = 1 für den minimalen Suchaufwand gewählt werden. In Tabelle 3.6 werden die Suchstrategien mit (d_1=1; d'=1) und (d_1=4; d'=1) anhand

der auf den Bildern von Fig. I bis IV mit Hilfe der atomaren Raster
{ $\begin{matrix} * \\ * * \end{matrix}$, $\begin{matrix} * \\ * \end{matrix}$, $\begin{matrix} * \\ * \end{matrix}$, $\begin{matrix} * \\ * \end{matrix}$ } erzielten Kompressionsfaktoren gegenübergestellt. Bei den hier durchgeführten Experimenten betrug der Rechenaufwand der Strategie mit $d_1=4$ im Mittel das 10^3-fache des Rechenaufwandes der Strategie mit $d_1=1$. Trotz dieses erheblichen Anstieges des Rechenaufwandes konnten keine nennenswerten Verbesserungen des Kompressionsfaktors erzielt werden.

Bild	Fig. I	Fig. II	Fig. III	Fig. IV
Kompressionsfaktor ($d_1=d'=1$)	7.88	8.68	8.18	1.83
Kompressionsfaktor ($d_1=4;d'=1$)	7.94	8.74	8.30	1.85

Tabelle 3.6: Vergleich der Suchstrategien ($d_1=d'=1$) und ($d_1=4$; $d'=1$).

Bereits bei einem minimalen Suchaufwand können gleichmäßige Suchstrategien zu bedeutenden Verbesserungen des Kompressionsfaktors führen, insbesondere wenn eine geeignete Menge von atomaren Rastern verwendet wird. Selektionskriterien für solche Raster können wir hier nicht angeben. In unseren Versuchsreihen haben die aus zwei benachbarten Punkten bestehenden Raster bei den hier benutzten Bildern am besten abgeschnitten. Es bleibt jedoch festzuhalten, daß diese Bilder nur im begrenzten Rahmen repräsentativ sein können. Sie sind aus einem zur Charakterisierung von nuklearen Brennelementen zusammengestellten Bildmaterial entnommen. Im allgemeinen läßt sich

für gleichmäßige Suchalgorithmen feststellen, daß relativ hohe Kompressionsfaktoren erzielt werden können, die auch vom Rechenaufwand her vertretbar sind, wenn in den einzelnen Unterbäumen der Suchaufwand gering gehalten wird. Nachteilig bleibt nach wie vor ihre Myopie.

Bemerkung : (Reduktion des Suchbaumes)

Eine Möglichkeit, den Suchaufwand nach einem optimalen Pfad im Baum von Fig. 3.3 zu reduzieren, besteht darin, auf die Berücksichtigung der Reihenfolge der Knoten in einem Pfad zu verzichten. Das bedeutet: Alle Pfade von T, von denen bis auf die Reihenfolge der Knoten einer bereits existiert, werden aus T entfernt. Der resultierende Baum wird T^* genannt. Bei m atomaren Rastern und einer Baumtiefe d reduziert sich die Anzahl der Pfade in T^* von m^d auf d^{m-1}. Für m=4 wird T^* in Fig. 3.4 dargestellt. Diese Baumreduktion hat dann keine nachteilige Wirkung auf die Datenkompression, solange es keine Bildteile im sukzessiv erodierten Bild c gibt, in denen eines der atomaren Raster nicht enthalten ist. Andernfalls kann die Baumreduktion zum Verlust eines optimalen Pfades führen.

3.2.2 Verbesserung des Kompressionsfaktors einer Sequenz von Rastern

Die bei der Anwendung von gleichmäßigen Suchstrategien berechneten Wurzelzellen bezüglich der Menge ς von atomaren Rastern können nach dem selben Verfahren oder nach dem Verfahren des optimalen Rasters weiter codiert werden. Im letzteren Fall wird für jede Generation von Wurzelzellen aus einer Menge ς^* (nicht unbedingt gleich ς) von atomaren Rastern ein für diese Konfiguration von Wurzelzellen optimales Raster ermittelt. Diese Verbesserung des Kompressionsfaktors wurde bereits im Zusammenhang mit den Wurzelzellen bez.

eines optimalen Rasters im Abschnitt 3.1.1 besprochen. Tabelle 3.7 enthält verbesserte Kompressionsfaktoren (in Klammern die nicht verbesserten), die mit den aufgezeichneten Rastern auf dem Bild von Fig. I erzielt wurden. Die im allgemeinen signifikante Verbesserung ist dabei zu unterstreichen.

Atomare Raster	Kompressions-faktor	Atomare Raster	Kompressions faktor
* **,*	7.6 (6.8)	* **,*	7.3 (5.3)
* **, *	7.1 (5.2)	* * *,*	7.7 (6.1)
* * *, *	7.8 (6.0)	* * * , *	5.7 (5.0)
* * * **,*,* , *	9.2 (7.9)	** **	7.7 (5.5)

Tabelle 3.7: Kompressionsfaktoren des Bildes von Fig. I erzielt mit einer verbesserten gleichmäßigen Suchstrategie der Tiefe $d_1=1$. Die Verbesserung der Kompressionsfaktoren erfolgte durch eine weitere Codierung der mit der Suchstrategie erhaltenen Wurzelzellen mit Hilfe von $\zeta^* = \{$ * * , * , * , * $\}$.
Die Werte in Klammern sind die Kompressionsfaktoren ohne diese Verbesserung.

3.3 Vergleich der beiden heuristischen Methoden

Die in den beiden Abschnitten 3.1 un d 3.2 beschriebenen Strategien wurden im Hinblick auf ihre Datenkompression anhand der Bilder von Fig. I,II und VII (jeweils 256 x 256 Pixels) miteinander verglichen. Die erzielten Kompressionsfaktoren und die dafür benötigten logischen Bildoperationen sind in den Spalten 1 bis 8 der Tabelle 3.8 aufgelistet. Für die Codierungsmethode mit einer optimalen Sequenz von Rastern (vgl. Abschnitt 3.2) wurde die Menge der atomaren Raster

$$\zeta_a = \{ \begin{matrix}**\end{matrix}, \begin{matrix}*\\*\end{matrix}, \begin{matrix}*\\*\\*\end{matrix}, \begin{matrix}*&*\end{matrix} \}$$

gewählt und für die Tiefe der zu suchenden Unterbäume wurde $d_1 = 1$ gewählt. Für die Codierungsmethode mit einem optimalen Raster (vgl. Abschnitt 3.1) wurden jeweils die Raster

$$\begin{matrix} & * & \\ *** \\ & * & \end{matrix}, \quad \begin{matrix} *** \\ *** \\ *** \end{matrix}, \quad \begin{matrix} ** \\ ** \end{matrix}$$

als optimal angenommen. Die weitere Codierung der Wurzelzellen erfolgte in allen Fällen mit Hilfe von ζ_a. In den letzten 4 Spalten sind die Ergebnisse der Mittelachsentransformation (vgl. Abschnitt 2.4) bezüglich der von Neumann- und Moore-Raster jeweils aufgezeichnet.

Aus diesen Ergebnissen geht hervor, daß vom Gesichtspunkt der Datenkompression die Codierung mit einer Sequenz von optimalen Rastern zu deutlich besseren Kompressionsfaktoren führt. Der dafür höhere Rechenaufwand bleibt weiterhin vertretbar.

Methoden	MOS				MOR				MAT			
Raster	** * *	* *	* *** *		*** *** ***		** **		* *** *		*** *** ***	
CF und Zeit Bilder	CF	T	CF	T	CF	T	CF	T	CF	T	CF	T
Fig. I	9.2	3	6.6	1	6.5	1	7.7	2	4.2	&	3.4	&
Fig. II	8.9	5	4.8	2	6.3	1	6.7	2	4.1	&	3.8	&
Fig. VII	4.9	2	3.2	1	4.5	1	4.6	2	2.2	&	2.1	&

Tabelle 3.8: Kompressionsfaktoren CF und Rechenzeit T (in 10^3 logischen Bildoperationen (LBO); & bedeutet unter 500 LBO) der Codierungsmethoden:

MOS = Methode der optimalen Sequenz (vgl. Abschnitt 3.2)
MOR = Methode des optimalen Rasters (vgl. Abschnitt 3.1)
MAT = Mittelachsentransformation (vgl. Abschnitt 2.4)

4. Datenkompression und Szenenanalyse

Effiziente Codierungsalgorithmen zeichnen sich dadurch aus, daß sie neben einer hohen Datenkompression zusätzliche Anforderungen erfüllen (vgl. Abschnitt 2.1), insbesondere wenn sie eine gewisse Beschreibung des zu codierenden Bildes beinhalten. Diese Beschreibung hängt natürlich von der Art der Bilder ab: Wenn beispielsweise Dokumente (Faksimile, Zeichnungen ...) mit statistischen Methoden (vgl. dazu unter anderem /Huang (1977)/), oder durch Zerlegung in geometrische Muster /z.B. Pavlidis (1968), Zamperoni (1980)/ etc... codiert werden, dann wird keine Information über den semantischen Inhalt des Bildes vermittelt. Dagegen werden bei den auf Mustererkennung basierenden Codierungsmethoden, beispielsweise bei der Codierung von Zeichen (characters) von /Ascher und Nagy (1974), Pratt et al. (1980)/ die Objekte im Bild erkannt und zur Speicherung oder Übermittlung codiert. Ebenso aufschlußreich können geometrische und topologische Merkmale der aus Materialkunde, Biologie und vergleichbaren Gebieten stammenden Bilder sein, wenn sie durch geeignete Codierungsmethoden zusätzlich extrahiert werden können. Eigentliche Methoden zur Merkmalextraktion wurden in vielen Büchern beschrieben (vgl. dazu z.B. /Duda und Hardt (1972), Kazmierczak (1980) Niemann (1983)/,...), insbesondere in Zusammenhang mit der Computer-Vision (computer vision) und Robotik (z.B. /Marr (1982), Ballard (1982), Pugh (1983)/,...). Zahlreiche Arbeiten zu diesem Thema, die in den letzten 10 Jahren publiziert wurden, können in den meisten Fachzeitschriften der Bildverarbeitung oder künstlichen Intelligenz konsultiert werden.

Wir werden in diesem Kapitel (in den Abschnitten 4.1 und 4.2) die Merkmalextraktion ausschließlich in Zusammenhang mit den bisher eingeführten Codierungsmethoden und mit weiteren in den Abschnitten 4.3 und 4.4 noch zu beschreibenden Codierungsmethoden behandeln . Dabei

werden wir uns auf die geometrischen und topologischen Eigenschaften sowie auf die hierarchische Strukturierung des Bildes beschränken.

4.1 Merkmalextraktion mit der Codierungsmethode des optimalen Rasters

Die heuristische Codierungsmethode zur Ermittlung eines optimalen Rasters (vgl. Abschnitt 3.1) führt zu einer Darstellung eines Binärbildes c, die mit Hilfe der Baumstruktur von Fig. 3.1 aus einem (k+1)-Tupel von Vektoren $(V_0, V_1, \ldots, V_k)$ hergeleitet werden kann. Dabei sind $V_0 \in T^{k_1}$ und $V_i \in T^{k_{i+1,k_i}}$ $(i=1,\ldots,k;\ k_i \in \mathbb{N}$ und T ist die Menge aller globalen Transformationen) folgendermaßen definiert:

$$V_0 = (W_{N^*}^{(1)}, \ldots, W_{N^*}^{(k_1)})c = (v_{01}, \ldots, v_{0k_1})$$

$$V_1 = ((W_{N_{11}^*}^{(1)}, \ldots, W_{N_{11}^*}^{(k_{21})})\, v_{01}, \ldots, (W_{N_{1k_1}^*}^{(1)}, \ldots, W_{N_{1k_1}^*}^{(k_{2k_1})})\, v_{0k_1})$$

$$= (v_{11}, \ldots, v_{1k_2})$$

$$\vdots$$

$$V_i = ((W_{N_{i1}^*}^{(1)}, \ldots, W_{N_{i1}^*}^{(k_{i+1,1})})v_{i-1,1}, \ldots, (W_{N_{ik_i}^*}^{(1)}, \ldots, W_{N_{ik_i}^*}^{(k_{i+1,k_i})})v_{i-1,k_i})$$

$$= (v_{i1}, \ldots, v_{i,k_{i+1}})$$

$$\vdots$$

wobei für $V = (v_1,\ldots,v_n) \in T^n$ gilt : $Vc = (v_1c,\ldots,v_nc)$ $(n \in \mathbb{N})$. V_0c und $(V_1,\ldots,V_k)c$ werden grobe bzw. feine Darstellung von c genannt. Bei einer groben Darstellung von c impliziert eine i-te nichtleere Komponente $W^{(i)}_{*}c$ von V_0c die Existenz von in c enthaltenen Mustern iN^*, N^*_Ndie aus den Wurzelzellen $W^{(i)}_{*}c$ bis zum Rand des Trägers von c wachsen können.

Bei der feinen Darstellung von c haben diese Muster in der j-ten nichtleeren Komponente $v_{i,j}$ von V_ic $(j=1,\ldots,k_{i+1})$ die Form

$$a_1P_1 + a_2P_2 + \ldots + a_iP_i$$

wobei $P_1 = N^*$

$$P_u \in \{N^*_{u,1},\ldots,N^*_{u,k}\} \qquad (1 < u < i)$$

und $a_1 \in \{ 1,\ldots,k_1 \}$

$$a_u \in \{1,\ldots,k_{u+1,1}\} \cup \ldots \cup \{1,\ldots,k_{u+1,k_u}\}$$

Eine in der Bildanalyse von Proben aus der Materialkunde, Biologie und analogen Disziplinen oft verwendete Methode zur Klassifizierung von Objekten nach der Größe ist das sogenannte Delfiner Konzept /Delfiner (1972)/, bei dem die Größe eines Objektes durch die eines maximalen in diesem Objekt einschreibbaren Musters repräsentiert wird.

Dabei wird das Muster als ein Vielfaches $\lambda \in \mathbb{N}$ eines frei wählbaren Rasters (auch strukturierendes Element genannt) ausgedrückt. Im allgemeinen werden Raster aus wenigen benachbarten Punkten benutzt, die zu Aussagen über Sehnenverteilungen, Flächenverteilungen etc... führen. In Bildverarbeitungssystemen mit quadratischem Gitter benutzt man in der Regel als strukturierende Elemente zwei benachbarte Punkte in den acht Richtungen für die Sehnenmessungen und das Moore-Raster, das von Neumann-Raster oder deren Summe für die Flächenmes-

sung. In Bildverarbeitungssystemen mit einem hexagonalen Gitter /Serra (1982)/ werden oft als strukturierende Elemente zwei benachbarte Punkte in den drei (0^o-, 60^o- und 120^o-) Richtungen für Sehnenmessungen und das kleinste Hexagon für Flächenmessungen verwendet. Der Parameter λ wird dann als Maß für die Größe der Partikel in der Probe herangezogen, und Objekte der Größe λ bilden eine λ-Größenklasse (oder auch einfach λ-Klasse) bezüglich des verwendeten Rasters.

Es erhebt sich natürlich die Frage, ob eine derartige Einteilung der Objekte aus dem Code V_0c eines Binärbildes c hergeleitet werden kann. Eine unmittelbare Verwendung des Codes, d.h. ohne weitere Transformation kann lediglich zu oberen Schranken für die Objektklassen sowie für die Anzahl von Objekten in jeder Klasse führen; denn wird ein Binärbild c in der Form einer groben Darstellung V_0c codiert, so bedeutet die Anzahl der Objekte in der i-ten nichtleeren Komponente $W^{(i)}_{N^*}c$ von V_0c eine obere Schranke für die Objekte in der von N^* implizierten i-ten Objektklasse. Diese obere Schranke ist auf die Zerlegung von Objekten zurückzuführen: Während der Berechnung einer Komponente von V_0c kann sich ein Objekt von c infolge von Konkavitäten seiner Kontur in mehrere Objekte zerlegen. Dies kann sowohl die Anzahl der Objekte einer nichtleeren Objektklasse erhöhen als auch eine fiktive Objektklasse erzeugen.

Bei einer Codierung des Bildes c durch seine feine Darstellung $(V_1,\ldots,V_k)c$ wird die Anzahl der Objekte in der j-ten nichtleeren Komponente v_{ij} eines Vektors V_ic als eine maximale Anzahl der Objekte in der vom Muster $\sum_{u=1}^{i} a_uP_u$ implizierten i-ten Klasse interpretiert.

Wir werden im folgenden die Anzahl der Objekte einer Klasse für den Fall einer groben Darstellung (die feine Darstellung ist analog) etwas näher betrachten. Zunächst wollen wir die Verteilung der Objekte einer Klasse bezüglich der gesamten Anzahl der Objekte sowie bezüglich der gesamten Fläche der Objekte von c in einer Definition präzisieren.

Definition 4.1: Sei c ein Binärbild, e(c) die Anzahl der Objekte in c und #c die Fläche der Objekte von c (d.h. die Anzahl der '1'-en in c). Ferner seien $W_N^{(n_i)}$ (i=1,2,...) die nichtleeren Wurzelzellen von c bezüglich N. Die reellen Funktionen F,A : $\mathbb{N} \to \mathbb{R}$ definiert durch

$$F(i) = \frac{e(D_N^{(n_j-1)}(W_N^{(n_j)}c))}{\sum_{j=1}^{i} e(D_N^{(n_j-1)}(W_N^{(n_j)}c))} \qquad A(i) = \frac{\#(D_N^{(n_j-1)}(W_N^{(n_j)}c))}{\sum_{j=1}^{i} \#(D_N^{(n_j-1)}(W_N^{(n_j)}c))}$$

werden Verteilungsfunktion und Flächenverteilung der i-ten Objektklasse genannt.

Bemerkungen:

a) Falls die Objekte von c einfach zusammenhängend sind, dann ist e(c) mit der sogenannten Eulerzahl identisch (siehe Abschnitt 4.1.1.1).

b) Aus solchen Verteilungsfunktionen F und A können bekanntlich weitere statistische Parameter (Flächenanteile von Klassen, mittlere Fläche eines Objektes, Varianz, Standardbweichung, etc...) einer als Binärbild zu analysierenden Probe beispielsweise aus der Materialkunde, Biologie, etc... hergeleitet werden. Darauf wollen wir in dieser Arbeit nicht eingehen.

Aus Definition 4.1 folgt, daß die Erstellung einer Verteilungs-

funktion F für die tatsächlichen Objekte im Bild sowohl der Anzahl der Objektklassen als auch der Anzahl der Objekte je Klasse bedarf. Für die Flächenverteilung wird neben der Fläche der Objektklassen die Fläche je Objektklasse benötigt. Die Anzahl der tatsächlichen Objektklassen sowie die Anzahl der Objekte in einer Objektklasse kann genau bestimmt werden, bzw. (mit weniger Rechenaufwand) approximiert werden, wenn die aufgrund der Objektzerlegung resultierende Erhöhung der Anzahl von Objekten und Klassen eliminiert bzw. eingeschränkt wird. Wir werden im folgenden zeigen, wie dies sowohl während der Erstellung des Codes V_0c als auch aus dem Code V_0c erfolgen kann.

4.1.1 Klassierung während der Codierung

In diesem Abschnitt wird zunächst ein Algorithmus beschrieben, der zu einer Approximation der Anzahl von Objekten und Objektklassen führt. Eine leichte, jedoch unter Umständen rechenaufwendige Modifikation des Algorithmus ermöglicht eine genaue Berechnung der Anzahl von Objekten und Objektklassen. Der Algorithmus dazu wird auch angegeben.

4.1.1.1 Approximation der Anzahl von Objekten je Objektklasse

Bei dieser Approximation wird während der Erstellung des Codes V_0c die Anzahl der gefundenen Objekte und Objektklassen durch partielle Rekonstruktion der Objekte aus deren erodierten Teilen korrigiert. Das Zählen von Objekten wird zunächst auf das Zählen von einzelnen Punkten zurückgeführt, wobei jeder Punkt genau einem Objekt entspricht. Der folgende Algorithmus beschreibt diese Schritte. Weitere Möglichkeiten zum Zählen von Komponenten eines Binärbildes bilden das Markieren der Komponente (vgl. /Rosenfeld und Kak (1982)/) und die Verwendung der Eulerzahl (vgl. Bemerkung c) zu Algorithmus 4.1.1).

Algorithmus 4.1.1: Approximation der Anzahl von Objekten je Objektklasse

Sei c ein Binärbild und N^* das optimale Raster im Code $V_0 c$

0) Setze i = 1

1) Bilde $W^{(n_i)}_{N^*} c$

2) Rekonstruiere aus $E^{n_i}_{N^*} c$ die zugehörigen Objekte in $E^{n_i-1}_{N^*} c$. Nenne das Ergebnis X.

3) Bilde $Y := W^{(n_i)}_{N^*} c$ and not X.

4) Reduziere jede Komponente von Y zu einem Punkt. Die Anzahl dieser Punkte ist die gesuchte obere Schranke der Anzahl von Objekten in der i-ten Klasse.

5) Setze i := i+1 und wiederhole die Schritte 1 bis 5 für alle Klassen i, d.h. bis $E^{n_i}_{N^*} c = \emptyset$ ist.

Bemerkungen zu Algorithmus 4.1.1 :

a) Die Rekonstruktion von Schritt 2 erfolgt folgendermaßen:
Seien $A_0 := D_{N^*}(E^{n_i}_{N^*} c)$ und $B := E^{n_i-1}_{N^*} c$.
1) Für t=0,1,... berechne $D_M(A_t)$, die Dilatation von A_t mit dem Moore-Raster M und bilde $Z := D_M(A_t)$ and B.
2) Bilde $A_{t+1} := A_t$ or Z und wiederhole Schritte 1 und 2, solange $A_{t+1} \neq A_t$ ist.

b) Die Reduktion jeder Komponente eines Binärbildes zu einem einzigen Punkt ist in der Literatur unter 'shrinking' bekannt. Parallele 'shrinking'-Algorithmen wurden in /Levialdi (1972),

Kameswara Rao et al. (1976), Rosenfeld (1979)/ beschrieben. Während der Algorithmus von /Rosenfeld (1979)/ nur auf einfach zusammenhängende Komponenten angewandt werden kann, können die Algorithmen von /Levialdi (1972)/ und /Kameswara Rao et al. (1976)/ beliebige Komponenten zu einem Punkt reduzieren, wobei in /Levialdi (1972)/ dieser Punkt zum Rand der Komponente gehört und in /Kameswara Rao et al. (1976)/ der Punkt im Zentrum der Komponente liegt.

c) Eine Alternative zur Reduktion von Schritt 4 des Algorithmus 4.1.1 ist das Zählen der Komponenten von Y. Das Zählen von Komponenten eines Binärbildes erfolgt bekanntermaßen mit Hilfe der sogenannten Eulerzahl (auch Konnektivitätszahl oder Genus (genus) genannt). Die Eulerzahl liefert bei einfach zusammenhängenden Objekten genau die Anzahl dieser Objekte /Minsky und Papert (1969)/. Sind jedoch die Objekte mehrfach zusammenhängend, so liefert die Eulerzahl die Anzahl der Objekte minus der Anzahl der in diesen Objekten eingeschlossenen Löcher. In diesem Fall kann die Eulerzahl zu nicht eindeutig interpretierbaren Ergebnissen führen. Eine Möglichkeit, Komponenten (mit oder ohne Löcher) eines Bildes zu zählen, besteht darin, zunächst die Löcher von nicht einfach zusammenhängenden Objekten zu schließen und danach die Eulerzahl des resultierenden Bildes zu berechnen. Das Löcherschließen erfolgt bekanntlich durch Propagieren vom Bildrand aus von '1'-en (d.h. Ersetzen von '0'-en des Hintergrundes durch '1'-en) bis zu den äußeren Rändern der Objekte. Anschließend wird die Negation des Ergebnisses mit dem ursprünglichen Bild durch logisches or verknüpft. Die Prozedur wird so oft wiederholt, wie mehrfach die Objekte zusammenhängend sind (d.h. Objekte in Löchern, die selbst wiederum Löcher einschließen etc...). Ein effizienterer Algorithmus zum Schließen von Löchern, der auf einem Ansatz der logischen xor-Operation basiert, wird in Abschnitt 5.3 beschrieben.

Algorithmen zur Berechnung der Eulerzahl von Binärbildern (in einem quadratischen Gitter) mit Hilfe von lokalen Nachbarschaftsoperationen werden in /Rosenfeld (1979)/ und /Gray (1971)/ beschrieben, wobei in letzterer Arbeit auch der Fall vom hexagonalen Gitter berücksichtigt wird. Die dort benutzten lokalen Operationen ergeben für die Berechnung der Eulerzahl bezüglich der von Neumann-Nachbarschaft den gleichen Rechenaufwand wie bezüglich der Moore-Nachbarschaft, während die von /Rosenfeld (1979)/ betrachteten Nachbarschaften ein 1:2-Verhältnis zugunsten der von Neumann-Nachbarschaft aufweisen. Im einzelnen gilt: Seien d,e,f,t,v sowie e_i, t_i, d_j (i=1,...,4; j=0,1) und

$$e' = e_1+e_2+e_3+e_4$$
$$t' = t_1+t_2+t_3+t_4$$
$$d' = d_0+d_1$$

die Anzahl der aus Tabelle 4.1 jeweils in einem Binärbild c vorkommenden Muster.

Muster	$\begin{matrix}1\end{matrix}$	$\begin{matrix}11\end{matrix}$ oder $\begin{matrix}1\\1\end{matrix}$	$\begin{matrix}1\\1\end{matrix}$ oder $\begin{matrix}1\\1\end{matrix}$	$\begin{matrix}1&1\\11,&11,\end{matrix}$ $\begin{matrix}11\\1\end{matrix}$ oder $\begin{matrix}11\\1\end{matrix}$	$\begin{matrix}11\\11\end{matrix}$
Anzahl	v	e	d	t	f

Muster	$\begin{matrix}00\\01,\end{matrix}$	$\begin{matrix}00\\10,\end{matrix}$ $\begin{matrix}10\\00,\end{matrix}$ oder $\begin{matrix}01\\00\end{matrix}$	$\begin{matrix}10\\01\end{matrix}$ oder $\begin{matrix}01\\10\end{matrix}$	$\begin{matrix}01&10&11\\11,&11,&10\end{matrix}$ oder $\begin{matrix}11\\01\end{matrix}$
Anzahl	e_1	e_2 e_3 e_4	d_0 d_1	t_1 t_2 t_3 t_4
		e'	d'	t'

Tabelle 4.1: Betrachtete Muster und die Anzahl ihrer Vorkommen zur Berechnung der Eulerzahl.

Satz 4.1: (nach /Rosenfeld (1979)/)
Seien c ein Binärbild, $e_H(c)$ und $e_M(c)$ die Eulerzahlen von c bezüglich der von Neumann- und Moore-Nachbarschaften respektiv. Ferner seien v, e, d, t und f die Anzahl der in c vorkommenden Muster gemäß Tabelle 4.1. Dann gilt:

$$e_H(c) = v-e+f \quad \text{und} \quad e_M(c) = v-e-d+t-f$$

Beweis: (siehe /Rosenfeld (1979)/)

Satz 4.2: (nach /Gray (1971)/)
Seien c, $e_H(c)$ und $e_M(c)$ wie in Satz 4.1 und e', d', t' die Anzahl der in c vorkommenden Muster gemäß Tabelle 4.1. Dann gilt:

$$4e_H(c) = v' - t' + 2d'$$

und

$$4e_M(c) = v' - t' - 2d'$$

Beweis: (siehe /Gray (1971)/)

Man kann leicht nachweisen, daß die Eulerzahl gleich der Anzahl der Konvexitäten eines Binärbildes in einer Richtung minus der Anzahl der Konkavitäten in derselben Richtung ist. Werden aufgrund dieser Eigenschaft der Eulerzahl die e_i, t_i $(i=1,\ldots,4)$ und d_j Muster $(j=0,1)$ nach den zusammengehörenden Konvexitäten und Konkavitäten gruppiert, so ergibt sich unmittelbar:

Satz 4.3 : Seien c, $e_H(c)$ und $e_M(c)$ wie in Satz 4.1 und e_i, t_i, d_j $(i=1,\ldots,4;\ j=0,1)$ die Anzahl der in c vorkommenden Muster gemäß Tabelle 4.1. Dann gilt:

$$e_H(c) = e_i - t_i - d_{mod(i,2)}$$
$$e_M(c) = e_i - t_i + d_{mod(i+1,2)}$$

Beweis : Sei zunächst der Fall der von Neumann-Nachbarschaft be-

trachtet. Für alle Muster einer 2x2-Nachbarschaft ("quad") markiere man die beiden Grenzlinien zwischen einer '1' und ihren beiden '0'-Nachbarn (im quad). Solche Linien kommen nur in quads mit einer einzigen '1' sowie in quads mit zwei diagonalen '1'-en vor. Diese quads werden als Konvexitäten betrachtet und positiv gezählt. Analog werden die Grenzlinien, die eine 0 von ihren beiden '1'-Nachbarn (im quad) trennen markiert (dies ist nur bei quads mit drei '1'-en der Fall), und die entsprechenden Konkavitäten negativ gezählt. Mit dieser Markierung haben dann eine Konvexität und eine Konkavität dieselbe Richtung (d.h. sie werden zusammengruppiert), wenn deren Grenzlinien bis auf eine Translation ein Quadrat bilden. Somit ergibt sich genau die Einteilung von Satz 4.3. Im Falle der Moore-Nachbarschaft wird analog verfahren, wobei zur Konvexität nur die quads mit einer einzigen '1' und zur Konkavität zusätzlich zu den quads mit drei '1'-en auch die "quads" mit zwei diagonalen '1'-en beitragen.

Mit dem Satz 4.3 reduziert sich die Berechnung der Eulerzahl auf die Betrachtung einer der vier äquivalenten Klassen von 2x2-Nachbarschaften. Die Berechnung kann weiterhin vereinfacht werden, wenn in jeder Klasse das "don't care" Bit berücksichtigt wird; denn es gilt:

Satz 4.4: Seien e_1, t_1, v_1 und d_1 die Anzahl der Muster

X0 01 00 01
01, 11, 01 und 1X

im Binärbild c, wobei X ein "don't care" Bit bedeutet. Ferner seien e_i, t_i, v_i und d_i (i=2,3,4) die Anzahl ihrer jeweiligen Rotationen um jeweils 90-Grad. Dann gilt:

$$e_H(c) = e_i - t_i$$

$$e_M(c) = v_i - d_i$$

Beweis: Dies ergibt sich unmittelbar daraus, daß die quads mit den beiden diagonalen '1'-en im Falle der von Neumann-Nachbarschaft den Konvexitäten und im Falle der Moore-Nachbarschaft den Konkavitäten angerechnet werden (vgl. Beweis von Satz 4.3).

4.1.1.2 Genaue Anzahl der Objekte je Objektklasse

Die genaue Anzahl der Objekte einer Klasse kann durch eine leichte, jedoch rechenaufwendige Modifikation von Algorithmus 4.1.1 ermittelt werden. Dieser zusätzliche Rechenaufwand erfolgt aufgrund der Rekonstruktion im Schritt 4 des folgenden Algorithmus.

Algorithmus 4.1.2: Anzahl der Objekte je Objektklasse

0) Setze $i = 1$

1) Bilde $W_{N^*}^{(n_i)} c$

2) Rekonstruiere aus $E_{N^*}^{n_i} c$ die zugehörigen Objekte in $E_{N^*}^{n_i-1} c$ gemäß Bemerkung a) zu Algorithmus 4.1.1. Nenne das Ergebnis X.

3) Bilde $Y := W_{N^*}^{(n_i)} c$ and not X.

4) Rekonstruiere aus dem ursprünglichen Bild c die zugehörigen Objekte von Y. Dabei werden

$$A_0 := D_{N^*}^{n_i}(Y) \quad \text{und} \quad B := c$$

in die Bemerkung a) zu Algorithmus 4.1.1 eingesetzt. Die Anzahl der Objekte kann nun mittels der Eulerzahl nach einer möglichen Löcherschließung errechnet werden.

5) Setze i := i+1 und wiederhole die Schritte 1) bis 5) für alle Klassen i, d.h. bis $E_{N^*}^{n_i} c = \emptyset$ ist.

Der Rechenaufwand von Schritt 4 kann der Bemerkung a) zu Algorithmus 4.1.1 leicht entnommen werden. Die vollständige Rekonstruktion aller Objekte einer Klasse kann zwar rechenaufwendig sein, liefert jedoch dafür sowohl Anzahl als auch Fläche dieser Objekte und ermöglicht somit die Aufstellung einer genauen Größenverteilung der Objekte im Bild.

Klassierung während der Codeerstellung ist im allgemeinen rechenintensiv und kann infolgedessen nur dann praktiziert werden, wenn neben der Codierung zusätzlich noch eine Bildanalyse angestrebt wird. Wir werden in den Abschnitten 4.3 und 4.4 Codierungsmethoden beschreiben, die mehr auf die Merkmalextraktion dieser Art zielen.

Erstrebenswert in der Codierung eines Bildes bleibt nach wie vor die Verwendung des Codes zur weiteren Bildanalyse. Wir werden im folgenden Abschnitt einen Algorithmus angeben, der aus dem Code $V_0 c$ eines Binärbildes c eine Verteilungsfunktion sowie eine Flächenverteilung extrahiert. Für den Code $(V_1,\ldots,V_k)c$ kann ein analoger Algorithmus konstruiert werden.

4.1.2 Klassierung nach der Codierung

Der folgende Algorithmus berechnet aus der Menge der nichtleeren Wurzelzellen $(W_N^{(n_1)} c,\ldots,W_N^{(n_k)} c)$ eines Binärbildes c die genaue Anzahl der Objektklassen (bezüglich eines gewählten Rasters N) sowie die Anzahl der Objekte je Objektklasse.

Algorithmus 4.1.3 : (zur Berechnung der Verteilungsfunktion F)

1) Bilde aus V_0c die k Musterklassen Z_i:

$$Z_i := D_N^{n_i-1}(W_N^{(n_i)}c)$$

2) Decodiere c :

$$c := \bigvee_{i=1}^{k} Z_i$$

und setze i := k

3) Rekonstruiere aus Z_i und c die Objekte der i-ten Objektklasse gemäß Bemerkung a) zu Algorithmus 4.1.1. Nenne das Ergebnis dieser Rekonstruktion X_i.

4) Bilde $Z_{i-1} := Z_i$ and not X_i

5) Setze i := i-1 und wiederhole 3) bis 5) so lange, bis i = 1 ist.

Offensichtlich bilden die nichtleeren Objektklassen X_i von Schritt 3 des obigen Algorithmus genau die vom Raster N implizierten Objektklassen. Die Anzahl der Objekte je Klasse kann nun mit einem der bereits erläuterten Verfahren (siehe Abschnitt 4.1.1.1) erfolgen.

4.2 Merkmalextraktion mit der Codierungsmethode der optimalen Sequenz von Rastern

Die heuristische Codierungsmethode von Abschnitt 3.2 zur Ermittlung einer optimalen Sequenz von Rastern, mit der ein Binärbild c anhand der Baumstruktur von Fig. 3.3 als Modell codiert wird, führt zu einem (k+1)-Tupel von Vektoren $(V_0, V_1, \ldots, V_k)$ aus denen sowohl der Codes V_0c (grobe Darstellung) als auch der Code $(V_1,\ldots,V_k)c$

(feine Darstellung) abgeleitet werden können. Dabei werden V_i (i=0,...,k) folgendermaßen definiert:

$$V_0 = (W_{N_{i_1}}, \ldots, W_{N_{i_k}})c$$

$$V_1 = (W_{N_{i_{k+1}} W_{N_{i_1}}}, \ldots, W_{N_{i_{k+l_1}} W_{N_{i_1}}})c$$

$$\vdots$$

$$V_k = (W_{N_{i_{k+l+1}} W_{N_{i_k}}}, \ldots, W_{N_{i_{k+l+l_k}} W_{N_{i_k}}})c$$

wobei

$$l = \sum_{i=1}^{k-1} l_i$$

Die bei der groben Darstellung V_0c implizierten, in c einschreibbaren Muster haben hier die Form:

$$s_1 = N_{i_1}$$

$$s_2 = s_1 + N_{i_2}$$

$$\vdots \qquad\qquad \vdots$$

$$s_k = s_{k-1} + N_{i_k}$$

Werden die Komponenten von V_0c zu einer feinen Darstellung $(V_1,\ldots,V_k)c$ weitercodiert, so werden die dabei verwendeten Raster die Existenz von Mustern in c impliziern, deren Form sich jeweils aus der (Minkowski' schen) Addition dieser Raster zu den entsprechenden Teilsequenzen $s_1, \ldots, s_k$ ergibt.

Im übrigen gilt für die beiden Codes V_0c und $(V_1,\ldots,V_k)c$ die im letzten Abschnitt besprochene Merkmalextraktion aus diesen Codes, insbesondere kann ein zu Algorithmus 4.1.1 analoger Algorithmus zur Berechnung der Verteilungsfunktion (bez. einer Sequenz von Rastern) konstruiert werden.

4.3 Codierung und Formbeschreibung

Neben der Größenverteilung der Objekte in einer Probe aus der Materialforschung beispielsweise, von der man physikalisch auf Größenverteilung von Partikeln auf einer Fläche oder in einem Volumen des zu untersuchenden Materials schließen kann, spielen oft Informationen über die Form der Objekte zur weiteren Charakterisierung der Probe eine wichtige Rolle. Formanalysen werden insbesondere bei Porenstrukturen von verschiedenen Materialien in der Werkstoffkunde, Metallurgie... sowie bei Zellen- und Gewebeuntersuchungen in der Biologie durchgeführt.

Eine formale Beschreibung einer Form (shape) ist bekanntlich ein komplexes Problem, das mit der dafür zu starren euklidischen Geometrie und zu vagen Topologie nicht zufriedenstellend (oft nicht einmal annährend zufriedenstellend) behandelt werden kann. In den sich mit diesem Gebiet beschäftigenden Wissenschaftskreisen unterschiedlicher Disziplinen (Mathematik, Physik, Informatik, Psychologie...) wird die Meinung vertreten, es müssen vielmehr neue, zur Formbeschreibung (shape description) geeignete mathematische Formalismen entwickelt werden. Interessante Beiträge zu diesem Thema sowie weitere Literaturhinweise können in /vgl. z.B. Zusne (1970), Marr (1982), Artificial Intelligence (1981)/ gefunden werden.

Die im nächsten Abschnitt beschriebene Methode, die neben der Kom-

pression des Bildes Aussagen über die Form seiner Objekte liefert, ist daher lediglich in einem begrenzten Anwendungsgebiet zu betrachten. Ebenso sind vergleichbare Methoden /z.B. Vollath (1982)/, die mit verschiedenen strukturierenden Elementen eine Formbeschreibung der Komponenten eines Binärbildes zu erzielen versuchen, in einem reduzierten Umfang einsetzbar.

4.3.1 Partitionierung eines Binärbildes

Die Partitionierung eines Binärbildes zielt darauf hin, das Bild als Vereinigung von disjunkten Mustern (gleicher oder unterschiedlicher) Form darzustellen. Muster gleicher Form bezeichnen wir auch als homothetisch.

Definition 4.2: Seien $c \in C_n$ ein Binärbild und ς eine endliche Menge von Rastern. Ein m-Tupel von Konfigurationen $(d_1,\ldots,d_m) \in (C_n)^m$ heißt Partition von c, wenn es ein m-Tupel von Konfigurationen $(b_1,\ldots,b_m) \in (C_n)^m$ und ein m-Tupel von Rastern $(N_1,\ldots,N_m) \in \varsigma^m$ gibt, so daß gilt :

1) $d_i = D_{N_i}^{n_i}(b_i) \quad \forall i=1,\ldots,m; \quad n_i \in \mathbb{N}$

2) $\forall\, p,q \in tr(b_i)\colon p+n_iN_i \cap q+n_iN_i = \emptyset \qquad (p \neq q)$

3) $c = \bigcup_{i=1}^{m} d_i$

4) $d_i \text{ and } d_j = \emptyset \qquad \forall\, i \neq j$

$(c_1,\ldots,c_m)$ und $(N_1,\ldots,N_m)$ legen die Partition $(d_1,\ldots,d_m)$ fest. Das m-Tupel von Rastern wird auch das m-Tupel von Partitionsrastern genannt. Das Binärbild c wird dann in Mustern $u+n_iN_i$ $(u \in b_i)$ partitioniert.

Bemerkungen:

a) Offensichtlich bilden $(b_1,\ldots,b_m)$, $(N_1,\ldots,N_m)$ und $(n_1,\ldots,n_m)$ einen Code von c (Bedingung 1) und 3).
b) $(b_1,\ldots,b_m)$ ist im allgemeinen von der Menge der Wurzelzellen von c gemäß Def. 2.2 verschieden.
c) Die Disjunktheit der Muster (Bedingung 2)) wird im Hinblick auf Aussagen über die Form der Objekte gefordert.

In den nächsten Abschnitten werden Algorithmen zur Bestimmung einer Partition von c beschrieben, oder genauer zur Berechnung der m-Tupel $(b_1,\ldots,b_m)$ und $(n_1,\ldots,n_m)$ mit Hilfe der Raster $N_1,\ldots,N_m$, die diese Partition festlegen und als Code von c verwendet werden. Zunächst sei der Fall betrachtet, in dem lediglich ein einziges Raster aus ζ verwendet wird. Das bedeutet: c wird in homothetische Muster partitioniert. Der Algorithmus hierzu wurde bereits in /Nasraoui (1984)/ veröffentlicht.

4.3.1.1 Ein Algorithmus zur Partitionierung in homothetische Muster

Die Disjunktheit der homothetischen Muster wird durch das Entfernen der "Überlappungsstellen" der Muster erreicht. Mit Hilfe des nächsten Lemmas 4.1 und des nächsten Satzes 4.5 werden Algorithmen beschrieben, die disjunkte Muster konstruieren.

Lemma 4.1: Es seien $c \in C_n$ ein Binärbild und $N \in \zeta$ ein konvexes Raster. Ferner sei b eine Konfiguration, für die gilt:

$$b = E_N^k(c) \neq \emptyset \qquad (k \in \mathbb{N})$$

$$E_N(b) = \emptyset$$

Dann existiert eine parallele Transformation $R : C_n \to C_n$, die b auf $\underline{b}$ abbildet, so daß gilt:

1) $tr(\underline{b}) \subseteq tr(b)$

2) $p+kN \cap q+kN = \emptyset \qquad \forall\ p,q \in tr(\underline{b}): p \neq q$

3) $E_N^k(D_N^k(b)$ <u>and not</u> $D_N^k(\underline{b})) = \emptyset$

<u>Beweis</u>: Der Beweis wird so konstruiert, daß eine Konfiguration b, die die Bedingung 2) nicht erfüllt, in $\underline{b}$ transformiert wird, so daß die Bedingungen 1) bis 3) erfüllt sind.
Seien $h_1 = \{(0,0),(1,0)\}$, $h_2 = -h_1$, $v_1 = \{(0,0),(0,1)\}$, $v_2 = -v_1$, $dl_1 = \{(0,0),(-1,1)\}$, $dl_2 = -dl_1$, $dr_1 = \{(0,0),(1,1)\}$, $dr_2 = -dr_1$ Raster.
Für ein Raster x sei $t_x(b) := E_x^{k'+1}(D_N^k(b))$, so daß gilt:
$E_x^{k'}(N) \neq \emptyset$ und $E_x^{k'+1}(N) = \emptyset$. Dabei ist $t_{h_i}(b)$ (i=1,2) die Konfiguration der '1'-Punkte von $D_N^k(b)$, die eine horizontale Reihe von k '1'-Nachbarn (d.h. einen horizontalen Pfad von '1'-Punkten) besitzen. $t_{v_i}(b)$, $t_{dl_i}(b)$ und $t_{dr_i}(b)$ stellen analoge '1'-Punkte in den anderen Richtungen respektiv dar. Die so definierten Transformationen $t_{h_i}(b)$, $t_{v_i}(b)$, $t_{dl_i}(b)$ und $t_{dr_i}(b)$ grenzen ein Gebiet ein, in das höchstens ein Muster kN eingeschrieben werden kann (vgl. Fig.4.1). $\underline{b}$ wird dann folgendermaßen konstruiert:

<u>Algorithmus 4.3.1</u>: Eliminieren von Überlappungen der Muster kN

0) Setze c' := c

1) Bestimme $t_{h_1}(b)$ und bilde c' := c' <u>and not</u> $t_{h_1}(b)$

2) Wiederhole Schritt 1 für t_{v_1}, t_{dl_1} und t_{dr_1}

3) Bilde $\underline{b} := E_N^k(c')$ <u>or</u> $\underline{b}$

4) Bilde c := c <u>and not</u> $D_N^k(\underline{b})$ und wiederhole Schritte 0 bis 3 solange $E_N^k(c') \neq \emptyset$ ist

5) Setze R(b) := $\underline{b}$

Satz 4.5: Seien $c \in C_n$ ein Binärbild, $N \in \zeta$ ein Raster und R die parallele Transformation von Lemma 4.1. Ferner seien $(c_1,\ldots,c_m)$, $(\underline{b}_1,\ldots,\underline{b}_m)$ und $(b_1,\ldots,b_m)$ die folgendermaßen iterativ definierten m-Tupel von Konfigurationen:

1) $c_0 := c$

2) $\underline{b}_i := E_N^{n_i}(c_{i-1}) \neq \emptyset$ und $E_N(\underline{b}_i) = \emptyset$ $\quad (n_i \in \mathbb{N};\ i=1,\ldots,m)$

3) $b_i := R(\underline{b}_i)$

4) $c_i := c_{i-1}$ and not $D_N^{n_i}(b_i)$

Dann legen $(b_1,\ldots,b_m)$ und N eine Partition von c gemäß Def.4.2 fest.

Beweis: Der Beweis folgt unmittelbar aus Lemma 4.1, wenn jede Komponente b_i als b im Algorithmus 4.1 betrachtet wird.

Die Iterationsschritte 1 bis 4 von Satz 4.5 enthalten nahezu alle Schritte eines Algorithmus zur Partitionierung von c in homothetische Muster.

Algorithmus 4.3.2 : Partitionierung in homothetische Muster

Sei c ein Binärbild, $\mathbb{N}$ ein konvexes Raster und i ein Indexzähler.

0) Setze i=1

1) Bestimme $n_i \in \mathbb{N}$, so daß gilt:

$\underline{b}_i := E_N^{n_i}(c) \neq \emptyset$ und $E_N(\underline{b}_i) = \emptyset$

Falls $n_i = 1$ setze $b_i := c$ und STOP

2) Reduziere $\underline{b}_i$ zu b_i gemäß Algorithmus 4.3.1

$b_i := R(\underline{b}_i)$

3) Bilde $c := c \text{ and not } D_N^{n_i}(b_i)$

4) Setze i := i+1 und wiederhole Schritte 1 bis 4

Algorithmus 4.3.2 liefert ein m-Tupel $(b_1,\ldots,b_m)$ von Konfigurationen, die zusammen mit dem Raster N eine Partition von einem Binärbild c gemäß Def.4.2 festlegen.

Die Erosion $E_N^{n_i}(c)$ von Schritt 1) wird in der Literatur oft die "allerletzte" (ultimate) Erosion genannt und zur Trennung von sich überlappenden Objekten benutzt /Meyer (1979), Beucher und Lantuejoul (1979)/.

4.3.1.1.1 Datenkompression

Eine codierte Form des Binärbildes c stellen das m-Tupel von Konfigurationen $(b_1,\ldots,b_m)$, das m-Tupel $(n_1,\ldots,n_m)$ der Anzahl der Erosionsschritte bis zur allerletzten Erosion und das Raster N dar. Es gilt per Konstruktion:

Satz 4.6: Sei c ein Binärbild und $(b_1,\ldots,b_m)$ das gemäß Algorithmus 4.3.2 konstruierte m-Tupel von Konfigurationen. Dann gilt :

$$c = \bigvee_{i=1}^{m} D_N^{n_i}(b_i)$$

Die Decodierung von c nach Satz 4.6 benötigt $O(m^2)$ Dilatationsschritte. Dies kann im Sinne des Hornerschen Schemas auf $O(m)$ Schritte beschleunigt werden. Es gilt trivialerweise:

Satz 4.7: Seien c ein Binärbild, $(b_1,\ldots b_m)$ ein m-Tupel von Konfigurationen und N ein Raster die eine Partition von c gemäß Algorithmus 4.3.2 festlegen. Dann gilt:

$$c = \bigvee_{i=1}^{m} \Delta_i$$

wobei Δ_i iterativ definiert wird durch :

$$\Delta_0 = \emptyset$$

$$\Delta_i = D_N^{n_i - n_{i+1}}(\Delta_{i-1} \underline{\text{or}}\ b_i) \qquad (i > 0)$$

$$n_{m+k} = n_m \qquad (k > 0)$$

Eine Datenkompression wird dann erzielt, wenn der Speicheraufwand für die einzeln '1'-Punkte der Konfigurationen b_i nicht größer als der für c ist. Erwartungsgemäß nimmt der Kompressionsfaktor mit steigender Anzahl der Klassen in der Partition von c ab. Insbesondere kann der Kompressionsfaktor kleiner als 1 werden, wenn die '1'-Punkte von b_m, dem Rest des Bildes nach Entfernung aller Klassen von Mustern, einzelnen codiert werden. Möglichkeiten zur Reduktion der Nachteile einer Codierung mit der Partitionsmethode bilden a) eine geeignete Wahl des Rasters; b) eine weitere Codierung des Restes b_m.

Wahl des Rasters

Unterschiedliche Raster führen zu unterschiedlichen Kompressionsfaktoren, die von dem zu codierenden Binärbild c abhängig sind. Vom Gesichtspunkt der Datenkompression wird aus einer endlichen Menge

von konvexen Rastern ein Raster selektiert, das den bestmöglichen Kompressionsfaktor erzielt. In der Tabelle 4.2 sind einige Kompressionsfaktoren zusammengestellt, die mit verschiedenen Rastern auf das Bild von Fig.V erzielt wurden. Dabei bedeuten die Werte in Klammern die erzielten Kompressionsfaktoren bei jeweiliger Vernachlässigung des Restes b_m. Die sich dann ergebenden partiellen Rekonstruktionen des Bildes von Fig.V sind in Fig.Va-Vc dargestellt.

Raster	** **** **** **	*** *** ***	* *** *****
Bild	Fig.V	Fig.V	Fig.V
Kompressions-faktor	0.49 (10.4)	0.6 (4.9)	0.3 (4.8)

Tabelle 4.2: Erzielte Kompressionsfaktoren bei der Partitionierung des Binärbildes von Fig.V in oktagonale, quadratische und dreieckige Muster.

Die Kompressionsfaktoren von Tabelle 4.2 unterstreichen die Einschränkung der Partitionsmethode als Codierungsmethode, wenn eine exakte Rekonstruktion des ursprünglichen Bildes gefordert wird und keine weitere Codierung des Restes durchgeführt wird. Im Hinblick auf eine partielle Rekonstruktion oder eine weitere Codierung des Restes, wird als Partitionsraster das Raster mit dem bestmöglichen,

bei Vernachlässigung des Restes entstehenden Kompressionsfaktors selektiert. Im Falle des Bildes von Fig.V und der Raster von Tabelle 4.2 wird als Partitionsraster das Oktagon gewählt.

Weitere Codierung des Restes

Um eine mögliche Datenexpansion zu vermeiden (vgl. Tabelle 4.2), ist eine weitere Codierung des Restes b_m erforderlich. Dazu können mit Hilfe weiterer Raster (im allgemeinen Teilmengen des Partitionsrasters) sowohl Codierungsmethoden von Kapitel 3 als auch weitere Partitionierungen verwendet werden. In Tabelle 4.3 sind Kompressionsfaktoren einer solchen weiteren Codierung im Falle der Bilder von Fig. II, Fig. V und Fig. VI in den 2. und 3. Spalten zusammengestellt. Eine Fortsetzung der hier mit dem selektierten Oktagon durchgeführten Partitionierung auf den jeweiligen Rest b_m mit dem Raster (0,0), (1,0), (0,1), (1,1) bei Vernachlässigung des dabei erneut entstehenden Restes führte zu den Werten der 2. Spalte von Tabelle 4.3. Dabei entstanden bei der partiellen Rekonstruktion die Bilder von Fig. II.A, Fig. V.A und Fig. VI.A. Die 1. Spalte enthält die Kompressionsfaktoren bei der Vernachlässigung des jeweiligen Restes b_m der Bilder von Fig. II, Fig. V und Fig. VI. Die partiell rekonstruierten Bilder sind in Fig. II.B, Fig. V.a und Fig. VI.B respektiv dargestellt.

Die verbesserte Codierungsmethode des optimalen Rasters (vgl. Abschnitt 3.1.1) ergab bei der Codierung des jeweiligen Restes b_m mit Hilfe der Raster {(0,0), (1,0)}, {(0,0), (0,1)} , {(0,0), (1,1)} und {(0,0), (1,-1)} die Kompressionsfaktoren der dritten Spalte von Tabelle 4.3.

Raster zur Partition des gesamten Bildes	** **** **** **	Raster zur Partition des Restes	** **
Raster zur Codierung des Restes mit der Methode von Abs. 3.1.1	**,	*, *,	* * *

Kompressions-faktoren bei :	Vernachlässigung des Restes	Approximation des Restes	exakter Rekonstruktion
Fig.II	31.0	6.2	4.0
Fig.V	10.4	3.4	1.2
Fig.VI	41.0	9.8	5.0

Tabelle 4.3: Kompressionsfaktoren der Bilder von Fig. II, Fig. V und Fig. VI mit der Vernachlässigung (1. Spalte), Approximation (2.Spalte) und exakten Codierung (3. Spalte) des jeweiligen Restes b_m. Die jeweiligen Partitionen mit dem Oktagon wurde mit dem Quadrat (s. Tabelle oben rechts) fortgesetzt. Die exakte Codierung des Restes wurde mit der verbesserten Methode des optimalen Rasters von Abschnitt 3.1.1 durchgeführt.

4.3.1.1.2 Größenverteilung

Eine Partition nach Algorithmus 4.3.2 eines Binärbildes c impliziert eine Unterteilung der Objekte von c in Klassen, die in Abhängigkeit des verwendeten Rasters nach Größe sortiert werden. Diese Klassierung wollen wir zunächst in Form einer Definition angeben:

Definition 4.3: Seien P die Menge der '1'-Komponenten (Objekte) von c und Q_i die Menge der '1'-Komponenten (Muster) von d_i (i=1,...,m). Ein Muster $q \in Q_i$ identifiziert ein Objekt $p \in P$, wenn $q \subset p$ und es

kein Muster q' gibt, so daß q' $\in Q_j$ mit $j < i$ und $q' \subseteq p$.
"q identifiziert p" wollen wir als Relation R(p,q) bezeichnen. Die Relation R unterteilt P in Klassen P_{k_i} definiert durch:

$$P_{k_i} := \{p \in P: \exists q \in Q_i: R(p,q) \}$$

Berühren oder überlappen sich die Objekte eines Bildes (in diesem Fall ist ein Objekt in der bisherigen Definition von einer '1'-Komponente verschieden), so stimmen die nach der Relation R von der obigen Definition implizierten Objektklassen nicht mit denen einer "intuitiven" Klassenbildung überein. Das Bild von Fig. V stellt ein Beispiel eines solchen Falles dar: Durch ihre gegenseitigen Berührungen bilden die Partikel in dieser Szene nahezu eine einzige '1'-Komponente (im Sinne von Def. 1.4 und der Bemerkung dazu). Die Unterteilung der Partikel in Klassen nach Def. 4.3 würde zu einer nicht sinnvollen Klassierung führen. Wir werden hier nicht versuchen, eine Definition für "intuitive" Objektklassen anzugeben und wollen daher die "entdeckten" Musterklassen als eine Annäherung der tatsächlichen Objektklassen annehmen. Da unterschiedliche Raster zu unterschiedlichen Musterklassen führen (vgl. z.B. die Bilder von Fig. V.1, V.2, und V.3), wollen wir das Raster zur Größenverteilung heranziehen, das die im nächsten Abschnitt beschriebene Formähnlichkeitsbedingung erfüllt. Zunächst sollen die in diesem Abschnitt mit einem beliebigen Raster entdeckten Musterklassen als obere Schranken für die tatsächlichen gelten.

Der Klassierung gemäß Definition 4.3 wird bei stochastischen Szenen (aus der Materialkunde, Biologie ...) viel mehr Bedeutung im Vergleich zu einer Betrachtung einzelner Objekte beigemessen, denn solche globalen Analysen sind im allgemeinen aufschlußreicher /Serra (1982), Delfiner (1972)/. In der hier eingeführten Partitionierung können folgende Abschätzungen der Anzahl von Objektklassen,

der Anzahl von Objekten in einer Klasse sowie der mittleren Fläche eines Objektes in einer Klasse unmittelbar aus dem der Partition $(d_1,\ldots,d_m)$ zugrundeliegenden Code $(b_1,\ldots,b_m)$, $(n_1,\ldots,n_m)$ und N abgeleitet werden:

i) Die Anzahl der Klassen P_{k_i} ist kleiner oder gleich m, der Anzahl der entdeckten Klassen von Mustern Q_i im Bild c.

ii) Die Anzahl der Objekte in jeder Klasse P_{k_i} ist kleiner oder gleich der Anzahl der Muster in Q_i.

iii) Die Fläche eines Objektes A(p) einer Klasse G_{k_i} kann (asymptotisch) angegeben werden durch:

$$A(p) \geq n_i^2 . A(N)$$

wobei n_i die benötigte Anzahl von Erosionsschritten zur Bestimmung von b_i (vgl. Algorithmus 4.3.2, Schritt 1) ist.

Mit Hilfe dieser Merkmale können Verteilungen über die implizierten Klassen von Objekten, über die Anzahl der Objekte je Klasse, über die Größe der Objekte (z.B. Fläche) in jeder Klasse etc... zumindest approximativ berechnet werden.

4.3.1.1.3 Formbeschreibung

In diesem Abschnitt wird ein heuristischer Ansatz beschrieben, der aus einer Partition eines Binärbildes c Aussagen über die Form (shape) der Objekte von c in bezug auf das Partitionsraster N ableitet. Es wird insbesondere versucht, die Ähnlichkeit zwischen der Form eines Objektes und der eines Partitionsrasters zu quantisieren. Dabei wollen wir folgendermaßen vorgehen: Wir bestimmen für jede

Klasse i von Mustern aus d_i alle Muster aus den Klassen $j \geq i$ (d.h. kleinere oder gleiche Muster), die in einem 'von Neumann-Abstand 1' von den Mustern der Klasse i entfernt liegen. Wir sagen dann, daß die Muster der Klasse i zu den Mustern der Klasse j benachbart sind. Da jedes Objekt von c von mindestens einem Muster einer Klasse i identifiziert wird, ist diese Identifikation (im Sinne der Formähnlichkeit) um so "unzutreffender" je mehr (kleinere oder gleiche) Muster anderer Klassen $j \geq i$ mit den Mustern der Klasse i benachbart sind.

Im folgenden werden wir eine heuristische Vorgehensweise vorschlagen, wie diese Nachbarschaften bestimmt und zur Spezifizierung (mit Hilfe einer Quantisierung) der formbezogenen Bezeichnung "zutreffend" benutzt werden können.

Definition 4.4: Sei $(d_1,\ldots,d_m)$ eine Partition von einem Binärbild c bez. eines Rasters N. Ein Muster p aus d_i und ein Muster q aus d_j heißen Nachbarn (bez. der Moore- oder der von Neumann-Nachbarschaft), wenn es mindestens einen Punkt x in p und einen Punkt y in q gibt, so daß x und y unmittelbare Nachbarn (bez. der Moore oder von Neumann-Nachbarschaft) sind.

Definition 4.5: Sei $(d_1,\ldots,d_m)$ eine durch $(b_1,\ldots,b_m)$ und N festgelegte Partition von c , $e(d_i)$ die Eulerzahl bez. der von Neumann-Nachbarschaft von d_i . Die symmetrische mxm-Matrix $\Omega := (\omega_{ij})$ definiert durch:

$$\omega_{ij} := \begin{cases} e(d_i)+e(d_j) - e(d_i \text{ or } d_j) & \text{falls } i \neq j \\ \#b_i - e(d_i) & \text{sonst} \end{cases}$$

wird Nachbarschaftsmatrix (der Muster) bez. der von Neumann-Nachbarschaft genannt.

Bemerkungen:

a) Analog kann eine Nachbarschaftsmatrix bez. der Moore-Nachbarschaft definiert werden.

b) Die Elemente ω_{ij} der Nachbarschaftsmatrix Ω werden als untere Schranken der Anzahl von "Berührungsstellen" zwischen Mustern aus d_i und d_j $(i \neq j)$ betrachtet. Dies ist dann der Fall, wenn benachbarte Muster aus d_i mit einem (oder mehreren) gemeinsamen Nachbarn aus d_j ein einfach zusammenhängendes Muster bilden. Genauer: Für je zwei Muster p, q aus d_i und d_j $(i \neq j)$ resp. sei z die Menge aller Punkte aus p, die einen unmittelbaren Nachbarn in q haben und aller Punkte aus q, die einen unmittelbaren Nachbarn in p haben. Sei z_{ij} die Anzahl aller solcher Mengen z in d_i und d_j und sei $Z := (z_{ij})$ die resultierende symmetrische mxm-Matrix. Dann ist leicht nachzuprüfen, daß $\omega_{ij} \leq z_{ij}$.

In Fig. 4.2 bis 4.4 ist die Matrix Ω für die Partition des Bildes von Fig. V bezüglich des Oktagons, des Quadrats und des Dreiecks von Fig. 4.1a als Partitionsraster respektiv dargestellt. Analoge Matrizen für das Bild von Fig. II sind in Fig. 4.5 bis 4.7 dargestellt. Für das Bild von Fig.VI ist die Matrix Ω bezüglich des Oktagons in Fig. 4.8 angegeben.

Fig 4.1a: Partitionsraster

Wie diese Matrizen nun benutzt werden, um den Ähnlichkeitsgrad zwischen der Form der Objekte einer entdeckten Klasse und der Form des diesen Objekten zugeordneten Rasters zu quantisieren, wollen wir als Definition angeben.

Definition 4.6: Seien $(d_1,\ldots,d_m)$ eine Partition eines Binärbildes c bez. eines Partitionsrasters N, $A(d_i)$ die Fläche der '1'-Komponenten von d_i und $\Omega = (\omega_{ij})$ die Nachbarschaftsmatrix von c. Ferner sei ξ_i die folgendermaßen definierte Größe:

$$\xi_i := \frac{A(d_i)}{A(d_i) + \sum_{j=1}^{m-i} \omega_{i,m-j} \frac{A(d_j)}{e(d_j)}}$$

ξ_i wird Ähnlichkeitsindex der i-ten Klasse der Objekte von c und des Partitionsrasters N genannt. $\xi := (\xi_1,\ldots, \xi_m)$ wird Vektor der Ähnlichkeitsindizes oder einfach Ähnlichkeitsvektor der Objekte von c und des Rasters N genannt.

Bemerkung: ξ_i läßt sich wegen $A(d_i) = k_i.n_i^2.A(N)$ (asymptotisch), wobei $k_i = \#b_i$, auch darstellen durch

$$\xi_i = \frac{1}{1 + \sum_{j=1}^{m-i} \omega_{i,m-j}.(k_i)^{-1}.(n_j/n_i)^2}$$

Definition 4.7 Sei ξ ein Ähnlichkeitsvektor der Objekte von c und des Rasters N und ξ_0 : $0 < \xi_0 \leq 1$. Die Form der Objekte der i-ten Klasse von c heißt dann ähnlich mit der von N (in bezug auf ξ_0), wenn gilt: $\xi_i > \xi_0$.

Jede Komponente des Ähnlichkeitsvektors ξ dient als Richtlinie zur Entscheidung, ob Objekte einer Klasse sich tatsächlich der Form des ihnen zugeordneten Rasters annäheren, oder ob das Raster sie lediglich aufgrund seines besten Abschneidens (unter den anderen Rastern) identifiziert hat, seine Form jedoch von der der Objekte völlig verschieden ist. Natürlich ist dies nur eine quantisierte Entscheidungshilfe, die bei einer geeigneten Wahl der Raster zur Lösung wohldefinierter Aufgaben (aus dem Feld der Charakterisierung und Klassierung von Proben) beitragen kann.

In Fig. 4.2 bis 4.8 sind die Ähnlichkeitsvektoren in der letzten Zeile jeweils eingetragen. Bei den Bildern von Fig. II und VI kann die Aussagekraft der errechneten Ähnlichkeitsvektoren morphologisch nachvollzogen werden.

4.3.1.2 Ein Algorithmus zur Partitionierung in Muster unterschiedlicher Form

Nach Definition 4.2 ist in einer Partition eines Binärbildes c in Muster unterschiedlicher Form jedes Muster einer Klasse d_i homothetisch zu N_i. Die Ermittlung jeder Klasse d_i erfolgt analog zur Partitionierung in homothetische Muster von Algorithmus 4.2 , wobei hier zusätzlich noch geprüft werden muß, welches Raster N_i der Klasse d_i zugeordnet wird. Als Prüfkriterium wird die maximale Fläche der Muster von d_i herangezogen. Die einzelnen Schritte des Algorithmus lauten:

Algorithmus 4.3.3 Partitionierung in Muster unterschiedlicher Form

Sei c ein Binärbild, $\zeta = \{N_i,\ i \in I \subset \mathbb{N}\}$ eine endliche Menge von konvexen Rastern, und i ein Indexzähler.

0) Setze i=1

1) Bestimme für alle Elemente N_j (j e I) von ζ den Schritt $n_i(N_j)$ e IN, so daß gilt:

$$\underline{b}_i := E_{N_j}^{n_i(N_j)}(c) \neq \emptyset \text{ und } E_{N_j}(\underline{b}_i) = \emptyset$$

Falls $n_i(N_j) = 1$ für alle j e I, setze $b_i := c$ und STOP

2) Bestimme für alle j e I

$$\underline{n}_i(N_j) := \{n_i(N_j).A^{1/2}(N_j)\}$$

Selektiere N_i^* für das gilt:

$$\underline{n}_i(N_i^*) := \max_j \{\underline{n}_i(N_j),\ j \text{ e } I\}$$

und nenne $\underline{b}_i^*$ das zugehörige $\underline{b}_i$

3) Reduziere $\underline{b}_i^*$ zu $b_i := R(\underline{b}_i^*)$ gemäß Algorithmus 4.1

4) Bilde $c := c \underline{\text{ and not }} D_{N_i^*}^{n_i}(b_i)$

5) Setze i := i+1 und wiederhole Schritte 1 bis 5

Algorithmus 4.3.3 liefert ein m-Tupel $(b_1,\ldots,b_m)$ von Konfigurationen und ein m-Tupel $(N_1^*,\ldots,N_m^*)$ von selektierten Rastern, die eine Partition von einem Binärbild c gemäß Def.4.2 festlegen. Er unterscheidet sich von Algorithmus 4.2 insofern, daß in Schritt 2) die allerletzte Erosion für alle Raster bestimmt wird und die dazu jeweils benötigte Anzahl von Erosionsschritten als Kriterium zur

Selektion des passenden Rasters für jede Klasse d_i von Mustern gewählt wird. Der Vergleich der Erosionsschritte $n_i(N_i)$ (bis zur allerletzten Erosion) mit den verschiedenen Rastern ermöglicht die Normierung $\underline{n}_i(N_j) = n_i(N_j).A^{1/2}(N_j)$.

4.3.1.2.1 Datenkompression

Bei einer Partition $(d_1,\ldots,d_m)$ eines Binärbildes c in Muster unterschiedlicher Form werden die m-Tupel $(b_1,\ldots,b_m)$, $(n_1,\ldots,n_m)$ und $(N_1^*,\ldots,N_m^*)$ als Code von c betrachtet. Auch hier muß der Rest des Bildes nach Entfernung aller Musterklassen weitercodiert werden, wenn Datenkompression erzielt werden soll. Infolgedessen sind die mit dieser Partition zu erwartenden Kompressionsfaktoren von der Weitercodierung des Restes abhängig und somit mit denen der Partition in homothetische Muster vergleichbar. Einige Experimente haben dies auch bestätigt. Daher ist der zusätzliche Aufwand nur durch eine bessere Abschätzung der Größenverteilung sowie eine mögliche Gewinnung von weiteren Aussagen über die Form der Objekte im Bild gerechtfertigt.

4.3.1.2.2 Größen- und Formverteilung

Abschätzungen der Objektklassen im Bild, der Anzahl der Objekte je Klasse, der mittleren Fläche eines Objektes einer jeden Klasse..., sowie eine daraus resultierende Größenverteilung können analog zur Partition in homothetische Muster auch im Falle einer Partition in Muster unterschiedlicher Form unmittelbar aus dem Code $(d_1,\ldots,d_m)$, $(N_1,\ldots,N_m)$ und $(n_1,\ldots,n_m)$ ermittelt werden. Zusätzlich noch kann in diesem Fall eine Formverteilung der Objekte in Bezug auf die verwendeten Raster erstellt werden, die dann mit Hilfe der approxi-

mierten Berührungsmatrix Ω im Sinne von Definition 4.7 korrigiert werden kann. Eine solche Matrix Ω für das Bild von FIG. VI ist in Fig.4.9 dargestellt. Dabei wurden die Raster von Fig. 4.1 als Partitionsraster benutzt.

Eine leichte Modifikation des Partitionierungsalgorithmus 4.3 führt zu einer genaueren Größen- und Formverteilung, bedarf allerdings der Rekonstruktion von allen Objekten einer jeden Klasse aus dem Ergebnis der jeweiligen allerletzten Erosion. Die Schritte des Algorithmus im einzelnen lauten:

Algorithmus 4.3.4: Größen- und Formverteilung

Sei c ein Binärbild, $\zeta = \{N_i,\ i \in I \subset \mathbb{N}\}$ eine endliche Menge von konvexen Rastern, und i ein Indexzähler.

0) Setze i=1

1) Bestimme für alle Elemente N_j ($j \in I$) den Schritt $n_i(N_j) \in \mathbb{N}$, so daß gilt:

$$\underline{b}_i := E_{N_j}^{n_i(N_j)}(c) \neq \emptyset \quad \text{und} \quad E_{N_j}(\underline{b}_i) = \emptyset$$

Falls $n_i(N_j) = 1$ für alle $j \in I$, setze $b_i := c$ und STOP

2) Bestimme für alle $j \in I$

$$\underline{n}_i(N_j) := n_i(N_j).A^{1/2}(N_j)$$

Selektiere N_i^* für das gilt:

$$\underline{n}_i(N_i^*) := \max_j \{\underline{n}_i(N_j),\ j \in I\}$$

und nenne $\underline{b}_i^*$ das zugehörige $\underline{b}_i$

3) Rekonstruiere aus $\underline{b}_i^*$ die Objektklasse P_i. Dabei verfahre folgendermaßen:

   a) Bilde $P := D_{N_i^*}^{n_i}(\underline{b}_i^*)$

   b) Dilatiere P mit dem Moore-Raster und bilde den Durchschnitt mit c so lange, bis die Fläche von P in zwei aufeinanderfolgenden Schritten konstant bleibt (vgl. Bemerkung a) zu Algorithmus 4.1).

   c) Setze $P_i := P$

4) Bilde $c := c \underline{\text{ and not }} P_i$

5) Setze i := i+1 und wiederhole die Schritte 1 bis 5

Algorithmus 4.3.4 berechnet Objektklassen P_1, P_2, ... , die nach Form und Größe ihrer Objekte sortiert werden. Form und Größe werden ja von der Menge der verwendeten Raster impliziert. Dabei ergeben sich folgende Verteilungen:

Definition 4.8: Es bezeichne $card(P_i)$ die Anzahl der Objekte von P_i. Die reellen Funktionen F_n und F_a definiert durch

$$F_n(i) := \frac{card(P_i)}{card(c)} \quad ; \qquad F_a(i) := \frac{A(P_i)}{A(c)}$$

werden Größenverteilungen bezüglich der Anzahl und der Fläche der Objekte respektiv genannt.

Aus solchen Verteilungsfunktionen können bekanntlich Mittelwerte, Varianzen, Standardabweichungen... errechnet werden. Wir wollen hier

darauf nicht weiter eingehen und werden lediglich die ersten sieben mit Algorithmus 4.4 bestimmten Klassen (nach Größe und Form) der Objekte von Fig. I in Fig. I.a bis I.h visualisieren. Die dabei benutzten Raster werden in Fig. 4.10 dargestellt.

5. Parallele Konturcodierung

Während in den vorherigen Kapiteln die Codierungsalgorithmen auf der Zerlegung von Binärbildern in geometrische Muster basierten, wird in diesem Kapitel die Frage einer parallelen Codierung mit Hilfe der Kontur behandelt. Konturcodierung ist auf den ursprünglich von /Freeman (1961)/ entwickelten und später ausführlich untersuchten sequentiellen Kettencode und seine Varianten /Sidhu und Boute (1972), Freeman (1974), Morrin (1976), Sobel (1978), Cederberg (1979), Chakravarty (1981) .../ zurückzuführen, der aufgrund einer hohen Datenkompression und einer weiteren Verwendung des Codes zur Merkmalextraktion den in der Praxis wohl am meisten benutzten Code darstellt.

Die Bezeichnung "parallele Konturcodierung" kann leicht zu unterschiedlichen Interpretationen führen: Während in /Duff (1982)/ die parallele Konturcodierung sich auf die Generierung des Kettencodes mit Hilfe paralleler Transformationen (vgl. Definition 1.6 von Kapitel 1) bezieht, wird in /Tuomenoksa et al.(1983)/ die Erzeugung des Kettencodes auf mehrere Prozessoren "Processing Elements (PE)" verteilt, wobei jeder Prozessor einen Teil des Kettencodes sequentiell generiert. Auch bei den sogenannten "single pass"-Algorithmen wird manchmal der Begriff "paralleler Algorithmus" verwendet. In dem beispielsweise von /Chakravarty (1981)/ angegebenen Algorithmus werden Linienbilder (z.B. Bilder, die nach einer Konturextraktion entstehen) mit einem Moore-Raster zeilenweise abgetastet (scan). Jeder 1-Pixel wird mit seiner Coderichtung entweder in eine bereits existierende Liste eingetragen oder als Anfangspunkt einer neuen Liste aufgenommen. Die Listen von Teilkurven derselben Linie werden anschließend aneinandergehängt. Auf dieser Weise findet die Konturverfolgung für alle Konturen gleichzeitig statt. "Single pass" Algorithmen werden neben der Codierung auch besonders zur Merkmalextrak-

tion in Echtzeit (real time) herangezogen /vgl. z.B. Agrawala und Kulkanari (1977), Pavlidis (1978), Lutz (1979)/.

Ein Algorithmus, der ausschließlich mit parallelen Transformationen einen Kettencode erzeugt, ist uns nicht bekannt, obgleich die Existenz eines solchen Algorithmus nach /Pecht (1986)/ formal nachgewiesen werden kann. Untersuchungen in dieser Richtung werden hier nicht verfolgt. Stattdessen wollen wir einen Algorithmus angeben, der jedem Konturpunkt aufgrund seiner lokalen Umgebung (Nachbarschaft) eine Coderichtung (d.h. die Lage des von ihm aus zu erreichenden Nachbarn in einer Konturverfolgung) zuordnet. Somit werden alle Konturpunkte, die eine gleiche Nachbarschaft aufweisen, mit Hilfe von globalen Transformationen gleichzeitig bestimmt. Dies führt dann zu einer Liste der Koordinaten von Konturpunkten mit der jeweiligen Coderichtung, die anschließend zu einem Kettencode sortiert wird. Da die Koordinaten der Konturpunkte durch ein Abtasten (zeilenweise) des die Kontur enthaltenen Bildes erfolgt, ist diese Liste in einem gewissen Sinne bereits sortiert, und infolgedessen kann der Kettencode mit Hilfe eines einfachen, in Hardware leicht realisierbaren "look-up table" Mechanismus erzeugt werden.

Die Abbildung der Menge der Konturpunkte auf die Menge der Coderichtungen mit Hilfe von Nachbarschaften der Konturpunkte ist nicht immer injektiv. Dies ist insbesondere dann der Fall, wenn Linien im Binärbild vorkommen. Auch Zacken in der Kontur von (nicht linienhaften) Objekten können zu einer nicht injektiven Abbildung führen. Wir werden im ersten Abschnitt dieses Kapitels zunächst (einfache) Konturen codieren, in denen einem Konturpunkt höchstens zwei Coderichtungen zugeordnet werden können. Im zweiten Abschnitt werden wir einen Algorithmus zur Codierung von beliebigen Binärbildern angeben, verzichten jedoch auf die zusätzlich benötigte Information zur Unterscheidung von Objekten und Löchern, wenn diese gemeinsame Kontur-

punkte besitzen. Binärbilder mit solchen Konturen werden infolgedessen bei der Decodierung mit Hilfe des im dritten Abschnitt beschriebenen Algorithmus nicht berücksichtigt.

5.1 Parallele Konturcodierung von einfachen Konturen

Die Einführung von einfachen Konturen erweist sich aufgrund der oben erwähnten Nicht-Injektivität der Abbildung der Menge der Konturpunkte auf die Menge der Coderichtungen als nützlich, um einen leichteren Einstieg in das zu behandelnde Codierungsproblem zu verschaffen und die Arbeitsweise des vorgeschlagenen Algorithmus besser nachvollziehen zu können. Von diesem didaktischen Grund abgesehen, stellt der in diesem Abschnitt beschriebene Algorithmus einen Spezialfall des im nächsten Abschnitt beschriebenen Algorithmus zur Codierung von beliebigen Konturen dar.

Zunächst wollen wir einige zum Teil aus der Literatur bekannte Begriffe in Form von Definitionen präzisieren.

Definition 5.1 : Sei $B : Z^2 \to \{0,1\}$ ein NxN-Binärbild und $tr(B)$ der Träger von B (vgl. Def.1.1). Eine Abbildung $\underline{K} : Z^2 \to \{0,1\}$ heißt Kontur von B genau dann, wenn gilt :

1) $tr(\underline{K}) \subseteq tr(B)$

2) $\forall\ p \in tr(B) : p \in tr(\underline{K}) \Leftrightarrow \exists e = (e_1,e_2) \in Z^2 :$
$|e| = 1$ und $\underline{K}(p+e) = 0$

Bemerkungen :

a) Oft wird mit "Kontur" der Träger $tr(\underline{K})$ bezeichnet. Wir werden auch gelegentlich mit Kontur $tr(\underline{K})$ meinen und dafür einfach K schreiben.

b) Äquivalent zur obigen Definition der Kontur ist folgende Formulierung: Ein Bildpunkt p eines Binärbildes gehört zur Kontur genau dann, wenn p einen direkten Nachbarn bezüglich des von Neumann-Rasters im Hintergrund besitzt.

Die Bestimmung der Kontur mit Hilfe von globalen Transformationen erfolgt unmittelbar aus der Definition 5.1:

Satz 5.1 : Sei B ein Binärbild, H das von Neumann-Raster und $E_H(B)$ die Erosion von B mit H. Dann gilt : $K = B \text{ and not } E_H(B)$.

Definition 5.2 : Sei K eine Kontur. Besitzt jeder Punkt von K genau zwei (direkte) Nachbarn (bezüglich des Moore-Rasters), so heißt K eine einfache geschlossene Kontur. Gilt dies für alle bis auf zwei Punkte von K, die je genau einen (direkten) Nachbarn besitzen, so heißt K eine einfache offene Kontur. Die beiden Punkte werden Anfangs- bzw. Endpunkt genannt.

Bemerkung : Eine einfache offene Kontur wird auch digitaler Bogen genannt /Rosenfeld (1974)/.

Definition 5.3 : Sei K eine einfache Kontur und $e = (e_1,e_2) \in Z^2$ mit $||e|| = 1$. Eine alle Punkte von K enthaltende Folge $(p)_{i=1,n}$ von n Punkten $p_i=(x_i,y_i) \in K$ definiert durch :

1) $y_1 = \min_i \{ y_i \}$ $\qquad$ $x_1 = \min_i \{ x_i \,/\, y_i = y_1 \}$

2) $\forall\, k > 1 : (x_k,y_k) = (x_{k-1},y_{k-1}) + (e_1,e_2)$, mit $(x_k,y_k) \neq (x_{k-1},y_{k-1})$, falls (x_k,y_k) kein Endpunkt ist.

3) $(x_n,y_n) = (x_1,y_1) - (e_1,e_2)$

heißt Konturkette.

Bemerkungen:

a) Konturketten von einfachen geschlossenen Konturen bestehen aus paarweise verschiedenen Punkten : $\forall\ i \neq j: p_i \neq p_j$.

b) Ist n die Anzahl der Konturpunkte einer offenen einfachen Kontur, so ist die entsprechende Konturkette 2n-1 lang. In diesem Fall wird die Kontur hin- und zurückverfolgt.

c) Ist die einfache Kontur geschlossen und gilt für p_2: $x_2 > x_1$ (bzw. $x_2 < x_1$), so heißt $(p)_{i=1,n}$ Konturkette im Uhrzeigersinn (bzw. dem Uhrzeigersinn entgegengesetzt).

Definition 5.4: Sei F die Menge aller (einfachen) Konturketten $(p)_{i=1,n}$ und $D = \{0,\ldots,7\}$. Die Nachbarn eines Konturpunktes p_i bezüglich des Moore-Rasters werden mit N_j $(j=0,\ldots,7)$ gemäß Fig.5.1 bezeichnet. Ferner seien $k : F \to D^n$ und $h : Z^2 \to D$ die folgendermaßen definierten Abbildungen :

$$k((p)_{i=1,n}) = (h(p_1),\ldots,h(p_n))$$

und

$$h(p_i) = j \iff p_{i+1} = N_j$$

Die Abbildung k wird Kettencode genannt. Oft wird auch mit Kettencode die Menge $k((p)_{i=1,n})$ bezeichnet.

Ziel einer Kettencodierung ist die Generierung eines Kettencodes im Sinne der Definition 5.4. Die uns bekannten Algorithmen zur Konturcodierung erzeugen alle den Kettencode, indem sie einen Startpunkt auswählen, in dessen Nachbarschaft nach dem nächsten Konturpunkt suchen, dem in Abhängigkeit seiner Position zum Startpunkt eine Coderichtung gemäß Definition 5.4 zuordnen und so fort. D.h. : Jeder Konturpunkt wird mindestens einmal besucht, dessen Nachbarschaft geprüft und in Abhängigkeit seines Vorgängers codiert. Diese Vorgehensweise wird als sequentielle Konturverfolgung bezeichnet.

Die in diesem und im folgenden Abschnitt beschriebenen Algorithmen unterscheiden sich von den klassischen Algorithmen in zwei wesentlichen Punkten : a) Die Prüfung der Nachbarschaft findet für alle Konturpunkte gleichzeitig statt. Dies führt dazu, daß nach genau acht Nachbarschaftsoperationen jeder Konturpunkt eine Coderichtung erhält. Dabei werden aus vier Punkten bestehende Nachbarschaften (vgl. Fig. 5.1) herangezogen. b) Die eigentliche Konturverfolgung wird darauf zurückgeführt, daß aus einer nach aufsteigenden (x,y)-Koordinaten sortierten Liste von Konturpunkten und jeweiligen Coderichtungen die Konturkette und somit gleichzeitig auch der Kettencode generiert werden.

Algorithmus 5.1: Parallele Codierung von einfachen Konturen

Sei B ein NxN-Binärbild von Objekten mit einfachen Konturen. Ferner seien N_i (i=0,...,7) die in Fig.5.1 aufgeführten Nachbarschaften.

1) Bestimme mit Hilfe der globalen Transformation von Satz 5.1 die Kontur K der Komponenten von B.

2) Ordne jedem Konturpunkt p =(x,y) e K gemäß der Abbildung a: K → IN definiert durch :

$$a(p) := N.y + x$$

die Adresse a(p) zu. Sei L die nach aufsteigenden Elementen sortierte Menge a(K). Mit anderen Worten: L ist die nach aufsteigenden y-Koordinaten und bei gleichen y-Koordinaten nach aufsteigenden x-Koordinaten sortierte Liste der Konturpunkte K. Seien $\underline{p}_i$ (i=1,2,...) die Elemente von L.

3) Ordne jedem Punkt von L den Index i seiner Nachbarschaft N_i im zugehörigen Objekt zu. Sei I : L → D diese Abbildung.

4) Seien d_x, d_y, $A : D \to N$ die gemäß Tabelle 5.1 definierten Abbildungen, und sei $\underline{p}$ das erste (bzw. das erste nach Schritt 5) nicht markierte) Element von L. Bilde die Konturketten und die zugehörigen Kettencodes der Objekte von B folgendermaßen :

- Der Startpunkt $p_1 = (x_1, y_1)$ der aktuellen Konturkette und seine Coderichtung $h(p_1)$ werden definiert durch :

$$p_1 := \underline{p}$$

$$h(p_1) := \max_i \{(i+4)_{mod8} \;/\; i \in I(\underline{p})\}$$

- Der i-te Punkt $p_i = (x_i, y_i)$ der aktuellen Konturkette und seine Coderichtung $h(p_i)$ werden definiert durch :

$$p_i := (x_{i-1} + d_x(h(p_{i-1})),\; y_{i-1} + d_y(h(p_{i-1})))$$

bzw. $$a(p_i) := a(p_{i-1}) + A(h(p_{i-1}))$$

$$h(p_i) = \begin{cases} I(p_i) & \text{falls } \#I(p_i) = 1 \\ I(p_i) \setminus \{(h(p_{i-1}) + 4)_{mod8}\} & \text{sonst} \end{cases}$$

5) Aktualisiere L, in dem jedes gefundene $\underline{p}_i$ in L markiert wird und wiederhole Schritte 4 und 5 (auf die nicht markierten Elemente von L) bis alle Elemente von L markiert sind.

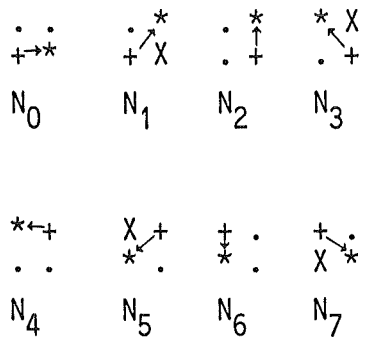

Fig.5.1: Nachbarschaften zur Generierung des Kettencodes. Der Pfeil zeigt die Richtung der Konturverfolgung am Konturpunkt '+' an.

'*': Objektpunkte;
'X': Don't care Punkte;
'.': Hintergrundpunkte

i	$d_x(i)$	$d_y(i)$	A(i)
0	1	0	1
1	1	-1	-N+1
2	0	-1	-N
3	-1	-1	-N-1
4	-1	0	-1
5	-1	1	N-1
6	0	1	N
7	1	1	N+1

Tabelle 5.1: Die Abbildung A und die entsprechenden Variationen d_x, d_y der x-, y-Koordinaten in Abhängigkeit von der Coderichtung i. Dabei bedeutet N die Zeilenlänge im Binärbild.

Die Parallelität des oben beschriebenen Algorithmus bezieht sich auf die Schritte 1 und 3, in denen einerseits alle Konturpunkte des Bildes mit Hilfe von globalen Transformationen gleichzeitig bestimmt werden (Schritt 1), andererseits alle Konturpunkte, die eine der Nachbarschaften N_i besitzen, auch gleichzeitig für alle Objekte im Bild bestimmt werden (Schritte 2 und 3). Das bedeutet: jede Nachbarschaft N_i wird ein Mal für alle Punkte und nicht für jeden Punkt einzeln geprüft.

Im Falle einer Implementierung des Algorithmus auf einem Bitschichtrechner mit der (oder einer ähnlichen) in Kapitel 2, Abschnitt 2.2.1 beschriebenen Architektur kann Schritt 2 des Algorithmus von einer Hardware-Einrichtung (Adreßspeicher) zum schnellen Einlesen der Koordinaten unterstüzt werden (vgl. Fig. 2.2). Dabei werden während des Ablegens des Ergebnisses einer logischen Operation zwischen zwei Bitschichten oder eines Transfers des Inhaltes einer Bitschicht auf eine andere die Koordinaten aus dem Bitstrom abgezapft und in ein RAM (Random Access Memory) nach aufsteigenden Adressen aufgelistet. Dieser Seiteneffekt hat keinen Einfluß auf die Dauer einer logischen Bildoperation.

Die eigentliche Konturverfogung findet in Schritt 4 statt. Da die Liste L der Konturpunkte beim Einlesen der Koordinaten bereits sortiert ist, besteht diese Konturverfolgung lediglich darin, anhand der Coderichtung des aktuellen Punktes einen "Offset" zu dessen Adresse gemäß Tabelle 5.1 zu addieren, um den nächsten Punkt zu ermitteln. Im Falle von einfachen Konturen werden in Schritt 3 jedem Punkt von L höchstens zwei Coderichtungen zugeordnet. Schritt 4 kann sich ebenso hardwaremäßig in Form einer "look up table" realisieren lassen.

Mit dem Markieren der Elemente von L in Schritt 5 unter Verwendung

der vollständigen Liste L wird gewährleistet, daß im Falle mehrerer Objekte alle Konturen erfaßt werden.

Algorithmus 5.1 stellt eine Umformulierung der sequentiellen Konturverfolgung in eine mit hohem Parallelisierungsgrad zu verzeichnende Kettencodierung dar, die mit einer leicht zu realisierenden Hardware-Einrichtung unterstüzt werden kann. Gerade bei solchen typischen sequentiellen Algorithmen , wie etwa der Kettencodierung oder der "single pass" Algorithmen, führen derartige Umformulierungen, die die Vorteile verschiedener Architekturen geschickt ausnutzen, i. allg. zu schnelleren Ausführungszeiten. Viele Aufgaben der Mustererkennung können durch die Kombination sequentieller und paralleler Algorithmen effizienter gelöst werden. Mit Hilfe paralleler Algorithmen können beispielsweise Vorverarbeitungsschritte (Bildreinigung...), sowie Codegenerierung (Mittelachsentransformation, Kettencode,...) schnell durchgeführt werden. Bildtransformations- und Merkmalextraktionsalgorithmen, die in geeigneter Weise auf codierte Daten anwendbar sind, werden dann vom Hostrechner (i. allg. ein sequentieller Rechner) übernommen.

5.2 Parallele Codierung von beliebigen Konturen

Aus didaktischen Gründen wurde zunächst der Algorithmus 5.1 zu parallelen Codierung einfacher Konturen beschrieben. Die Einschränkung auf diese Klasse von Konturen vereinfacht die eigentliche Konturverfolgung von Schritt 4 des Algorithmus 5.1, da jedem Konturpunkt höchstens zwei Coderichtungen zugeordnet werden können. Bei beliebigen Konturen muß aus der Menge der Coderichtungen, die jeder Konturpunkt besitzt, die richtige Coderichtung ermittelt werden. In diesem Abschnitt wird eine Verallgemeinerung des Algorithmus 5.1, insbesondere des Schrittes 4, zur parallelen Kettencodierung beliebiger Konturen beschrieben.

Algorithmus 5.2 :

Schritt 1 bis Schritt 3 wie in Algorithmus 5.1.

4) Seien d_x, d_y, $A : D \to \mathbb{N}$ die gemäß Tabelle 5.1 definierten Abbildungen, und sei $\underline{p}$ das erste (bzw. das erste nach Schritt 5 nicht markierte) Element. Bilde die Konturketten und die zugehörigen Kettencode der Objekte von B folgendermaßen :

- Der Startpunkt $p_1 = (x_1, y_1)$ der aktuellen Konturkette und seine Coderichtung $h(p_1)$ werden definiert durch :

$$p_1 := \underline{p}$$

$$h(p_1) := \max_i \{(i+4)_{mod8} \;/\; i \in I(\underline{p})\}$$

- Der i-te Punkt $p_i = (x_i, y_i)$ der aktuellen Konturkette und seine Coderichtung $h(p_i)$ werden definiert durch :

$$p_i := (x_{i-1}+d_x(h(p_{i-1})),\; y_{i-1}+d_y(h(p_{i-1})))$$

bzw. $$a(p_i) := a(p_{i-1}) + A(h(p_{i-1}))$$

Sei $(d_j)_{j=0,7}$ eine Sequenz von Elementen d_j definiert durch :

$$d_j(k,r) := (k + (r-j+8)_{mod8})_{mod8} \qquad k,r \in \mathbb{N}$$

Dann ist $h(p_i)$ das erste Element $d_j(h(p_{i-1}),r)$ der Sequenz $(d_j)_{j=0,7}$, das in $I(p_i)$ vorkommt und in L nicht markiert ist. r nimmt dabei den Wert 3 ein, da mit der Definition von $h(p_1)$ eine Konturverfolgung im Uhrzeigersinn festgelegt wird. Ersetzt man in der Definition von $h(p_1)$ 'max' durch 'min', so erfolgt eine Konturverfolgung in die dem Uhrzeigersinn entgegengesetzte Richtung, und r nimmt folglich den Wert 5 ein. Sind alle Elemente d_j, die in $I(p_i)$ vorkommen, bereits markiert, so ist $h(p_i)$ das letzte dieser Elemente.

5) wie in Algorithmus 5.1

Die im Anschluß an Algorithmus 5.1 formulierten Bemerkungen gelten natürlich auch für den allgemeineren Algorithmus 5.2. Insbesondere wird im Schritt 4 gewährleistet, daß stets der am weitestens links (bzw. rechts im Falle der Konturverfolgung in der dem Uhrzeigersinn entgegengesetzten Richtung) von der Verfolgerrichtung liegende Nachfolger des aktuellen Konturpunktes ermittelt wird. Das Markieren der Punkte in Schritt 5 bei der weiteren Verwendung weiterhin der vollständigen Liste L garantiert die Generierung geschlossener Konturketten für innere und äußere Konturen (bei Objekten mit Löchern) mit gemeinsamen Konturpunkten.

5.3 Decodierung

Die Bildrekonstruktion aus dem Kettencode erfolgt im allgemeinen in zwei Etappen /vgl z.B. Rosenfeld und Kak (1982), Pavlidis(1982)/. Zunächst werden die Konturen aus dem Kettencode generiert (contour tracing) und anschließend werden die Konturen aufgefüllt und somit Objekte und Hintergrund des Bildes erzeugt. Dabei wird die sogenannte Paritätsprüfung (parity check) verwendet. Eine die Kontur schneidende Linie wird zwischen ungeraden und geraden Schnittpunkten aus Einsen und in allen anderen Fällen aus Nullen bestehen. Konturen von besonderen Objekten im Originalbild (Linien der Breite 1, Objekte mit isolierten Punkten etc...) werden gesondert behandelt.

Wenn Bitschichtrechner zur Bildcodierung und -decodierung benutzt werden, dann ist die erste Etappe der Bildrekonstruktion, die Konturgenerierung aus dem Kettencode, aus Effizienzgründen vom sequentiellen Hostrechner auszuführen. Die zweite Etappe, das Auffüllen der Konturen, erfolgt sinnvollerweise im Bitschichtrechner. Der folgende Algorithmus beschreibt explizit die einzelnen Schritte einer solchen Bildrekonstruktion.

Algorithmus 5.3 : Bildrekonstruktion aus dem Kettencode mit einem Bitschichtrechner

Sei $k_j(p_{i=1,n_j})$ (j=1,...,m) der j-te Kettencode von m Konturen eines NxN-Binärbildes.

1) Erzeuge aus jedem Kettencode k_j die entsprechende Sequenz der Konturpunkte $(p)_{i=1,n_j}$ (in Form von Koordinaten) gemäß Def. 5.5

2) Setze die Konturpunkte auf eine Bitschicht.

3) Entferne aus den Konturpunkten alle Punkte P mit der Nachbarschaft von Fig. 5.3.

4) Bestimme (mit Hilfe der XOR-Operation) alle Konturpunkte an den ungeraden Stellen in x-Richtung, die einen gemäß Fig. 5.1 N_4 Nachbarn mit dem Wert 0 besitzen.

5) Dilatiere diese Punkte in x-Richtung bis zu den Konturpunkten an den geraden Stellen.

6) Addiere zum Ergebnis von Schritt 4 die in Schritt 3 entfernten Punkte.

```
0 0 0        X X X
0 P 0        0 P 0
X X X        0 0 0
```

Fig.5.3 : Nachbarschaft eines isolierten Konturpunktes P. 'X' bedeutet don't care Punkt

Schritt 1 des Algorithmus erfolgt natürlich im sequentiellen Hostrechner. Dort werden auch die Konturpunkte auf einen einer Bitschicht entsprechenden Speicherbereich (page) gesetzt, der dann vollständig auf eine Bitschicht im Bitschichtrechner transferiert

wird. Diese beiden Aktionen werden als Schritt 2 zusammengefaßt. In Schritt 3 soll verhindert werden, daß isolierte Punkte im Sinne von Fig. 5.3 zu unerwünschten Linien wachsen, wenn Schritt 4 ausgeführt wird. In Schritt 4 werden die Punkte bestimmt, die wachsen dürfen. Dieses Wachstum erfolgt in Schritt 5 als eine Dilatation in x-Richtung. Nach jedem Wachstumsschritt wird die Fläche (Anzahl der 1-Punkte) in der Ergebnis-Bitschicht gemessen. Die Konturen sind aufgefüllt, wenn in zwei aufeinanderfolgenden Messungen die Fläche konstant bleibt. Mit Schritt 6 wird das Ergebnis vervollständigt.

Im Spezialfall von einfach zusammenhängenden Objekten (Objekte ohne Löcher) wird nach dem Setzen der Konturpunkte auf eine Bitschicht der aus Einsen bestehende Rand zu den Konturen hin dilatiert. Die anschließende Negation der Bitschicht liefert das gewünschte Ergebnis. Bei dieser bekannten Prozedur wird nach jeder Dilatation mit dem Einheitsvektor als strukturierendem Element die Richtung (modulo 4 bzw. 8) gewechselt.

Eine Beschleunigung dieser Prozedur kann jedoch folgendermaßen erzielt werden:

1) Dilatiere den Rand in einer der 4 (bzw. 8) Richtungen und messe die resultierende Fläche so lange, bis keine Flächenveränderung mehr eintritt.
2) Wechsele die Richtung und wiederhole Schritt 1 so lange, bis keine Flächenänderung mehr für die 4 (bzw.8) Richtungen eintritt.

Man kann sich leicht klarmachen, daß diese Beschleunigung darauf zurückzuführen ist, daß nach der Dilatation in einer Richtung (bis zur Abbruchbedingung) alle gewonnenen Punkte dazu beitragen, neue Nachbarn vom Zustand 0 in den Zustand 1 zu versetzen, wenn die Richtung gewechselt wird.

6. Schlußbetrachtung

In der vorliegenden Arbeit wurden hauptsächlich auf heuristischen Methoden basierende Algorithmen zur Codierung von stochastischen Binärbildern mit parallelen Rechnern, insbesondere mit den sogenannten Bitschichtrechnern entwickelt. Der Begriff "parallel" wurde dabei auf Rechner bezogen, deren Architektur für die gleichzeitige Manipulation aller Bildpunkte während einer Bildoperation konzipiert ist, oder zumindest derartige Manipulationen besonders unterstüzt. Der Schwerpunkt der Codierung wurde einerseits auf die Datenkompression und andererseits auf die Beschreibung der stochastischen Szenen in Form einer Klassifikation ihrer Komponenten nach Größe und soweit möglich auch nach Form gelegt.

In Anlehnung an die klassische Mittelachsentransformation (MAT), bei der die lokalen Maxima eines Binärbildes mit Hilfe von von Neumann- bzw. Moore-Raster berechnet und zur Codierung des Bildes herangezogen werden, wurde zunächst die Frage untersucht, inwieweit weitere Raster zu besseren Kompressionsfaktoren führen können. Es hat sich herausgestellt, daß die Verwendung mehrerer Raster zur Codierung eines Binärbildes als ein Optimierungsproblem formuliert werden kann, wobei zur Annäherung an dessen optimale Lösung nur besonders schnelle heuristische Methoden in Frage kommen. Der intuitiven Heuristik kommt dabei aufgrund der während des Codierungsprozesses zu manipulierenden großen Datenmengen und der daraus resultierenden langen Ausführungszeiten eine besondere Bedeutung zu. Aus den zahlreichen Möglichkeiten, das Problem anzugehen, kristallisierten sich zwei heuristische Codierungsmethoden heraus, die auf das Nutzen-Kosten-Verhältnis der Suchprobleme, in diesem Fall den Kompressionsfaktor und die Ausführungszeit, zielten. Die Verwendung der nach diesen Methoden erstellten Code zur Szenenbeschreibung wurde im Hinblick auf eine Unterteilung der Bildkomponenten in Klassen

untersucht, aus denen Größenverteilungen hergeleitet werden können. Zur Bestimmung der Anzahl von Objekten in einem Bild wurde die sogenannte Eulerzahl herangezogen, zu deren Berechnung in diesem Zusammenhang ein besonders schneller Algorithmus entwickelt wurde.

Als Alternative zu dieser Codierungsart wurde eine weitere, auf der Partitionierung des Bildes basierende Codierungsmethode entwickelt, die neben der Klassifikation der Objekte nach ihrer Größe zusätzlich noch Aussagen, wenn auch in bescheidenem Maße, über deren Form erlaubt. Dabei wurden Ähnlichkeitsmatrizen aufgestellt, mit deren Hilfe Ähnlichkeitsindizes zur Beurteilung des benutzten Rasters im Hinblick auf Formwiedergabe der Objekte - mit anderen Worten zur Quantisierung der Formbeschreibung - berechnet werden können. Diese Codierungsmethode kann zu relativ hohen Datenkompressionen führen, wenn lediglich eine partielle Rekonstruktion des Originalbildes angestrebt wird. Ist dagegen eine exakte Rekonstruktion erforderlich, so ist i. allg. die mit dieser Methode erzielte Datenkompression sehr niedrig und für diesen Zweck allein nicht ausreichend.

Die Frage einer weiteren Verarbeitung der codierten Daten stand im Rahmen dieser Arbeit nicht im Vordergrund und wurde somit nicht weiter verfolgt. Es ist jedoch zu erwarten, daß angesichts der Verwandtschaft der hier entwickelten Codierungsmethoden mit der MAT (zumindest vom Ansatz her) eine ähnliche Weiterverarbeitung der Codes möglich sein kann.

Das klassische Beispiel eines zur Weiterverarbeitung besonders geeigneten Codes ist der Kettencode (auch Freeman Code genannt). Dieser Code ist aber auch beispielhaft für eine sequentielle Codierung. Die Parallelisierung des Kettencodes insbesondere im Hinblick auf die Hardwarestruktur des Bitschichtrechners wurde in dieser Arbeit untersucht. Dabei wurde ein Algorithmus entwickelt, der die

Stärken dieser Architektur mit denen des sequentiellen Hostrechners kombiniert. Mit diesem Algorithmus ist eine besonders schnelle Konturverfolgung zu erwarten. Die weitere Verarbeitung wird dann hauptsächlich vom sequentiellen Hostrechner übernommen. Somit können die in geeigneter Weise aufeinander abgestimmten Vorteile der parallelen und der sequentiellen Algorithmen zur effizienten Behandlung von Mustererkennungsproblemen führen, die meistens Echtzeitlösungen fordern.

Eine noch offene, unter Umständen vorteilhafte Codierungsart ist eine auf der Basis der Reproduzierbarkeit der Muster zu entwickelnde Codierung. Von der Theorie der Zellularautomaten her ist bekannt, daß mit Hilfe einfacher Operationen (z.B. XOR-Operation) beliebige endliche Muster nach einer endlichen Anzahl von Schritten reproduziert werden können. Bei der Implementierung dieser Reproduzierbarkeit wurden die sich nach jedem Schritt ergebenden Muster verfolgt, und es wurde beobachtet, daß völlig anders geartete Bilder während dieses Prozesses entstehen. Es erhebt sich die berechtigte Frage, ob mit Hilfe der einen oder anderen Codierungsmethode (z.B. der Konturcodierung) eines dieser Bilder nicht zu einem höheren Kompressionsfaktor als das Originalbild selbst führen kann.

Literaturverzeichnis

Agrawala, A. K., Kulkanari, A. V. (1977): "A Sequential Approach to the Extraction of Shape Features", Computer Graphics and Image Processing 6, 538-557

Amoroso, S., Cooper, G. (1971): "Tesselation Structures for Reproduction of Arbitrary Patterns", J. of Comput. and Syst. Scien. 5, 455-464

Arcelli, C., Cordella, L. P., Levialdi, S. (1981): "From Local Maxima to Connected Skeletons", IEEE Trans. on Pattern Analysis and Machine Intelligence, Vol. PAMI-3, No. 2, 134-143

Arps, R. B. (1979): "Binary Image Compression", in Image Transmission Techniques, New York, Academic Press, 219-276

Arps, R. B. (1980): "Bibliography on Image Compression", Proceedings of the IEEE, Vol. 68, No. 7, 922-924

Artificial Intelligence (1981) - Special Volume on Computer Vision -, Vol. 17, No. 1-3, North-Holland, Amsterdam

Ascher, R. N., Nagy, G. (1974): "A Means for Achieving a High Degree of Compaction on Scan-Digitized Printed Text", IEEE Trans Comput., Vol. C-23, No. 11, 1174-1179

Ballard, D., Brown, C.M. (1982): "Computer Vision", Englewoods Cliffs, New Jersey

Batcher, K. E. (1980): "Design of a Massively Parallel Processor", IEEE Trans. Comput. C-29, 836-840

Beucher, S., Lantuejoul, Ch. (1979): "Use of Watersheds in Contour Detection", Int. Workshop on Image Processing, CCETT, Rennes

Blum, H. (1967): "A Transformation for Extracting New Descriptors of Shape", in Models for the Perception of Speech and Visual Form, (ed. Wathen-Dunn, W.), MIT Press, Cambridge, Massachusetts, 362-380

Blum, H., Nagel, R. (1978): "Shape Description Using Weighted Symmetric Axis Features", Pattern Recognition 10, 167-180

Burks, A. W. (ed.) (1970): "Essays on Cellular Automata", Univ. of Illinois

Chakravarty, I. (1981): " A Single-Pass, Chain Generating Algorithm for Region Boundaries", Computer Graphics and Image Processing 15, 182-193

Cederberg, L. T. R. (1979): "Chain-Link Coding and Segmentation for Raster Scan Devices", Computer Graphics and Image Processing 10, 224-234

Delfiner, P. (1972): "A Generalization of the Concept of Size", J. Microscopy, Vol. 95, Pt 2, 203-216

Duda, R. O., Hart, P. E. (1973): "Pattern Classification and Scene Analysis", Wiley, New York

Duff, M. J. B. (1982): "Parallel Algorithms and Their Influence on the Specification of Application Problems", in Preston, K. Jr., and Uhr, L., (eds.), Multicomputers and Image Processing -Algorithms and Programs-, Academic Press, S. 261-274

Duff, M. J. B. (1982): "Special Hardware for Pattern Processing", Proc. of 6-th Inter. Conf. Pattern Recogn., Munich, Germany, 368-379

Elias, P. (1955): "Predective Coding", IRE Trans. Inf. Theory IT-1, 16-33

Feilmeier, M. (ed.) (1977): "Parallel Computers -Parallel Mathematics", IMACS, Amsterdam, New York, Oxford

Flanders, P. M., Hunt, D. J., Reddaway, S. F., Parkinson, D. (1977): "Efficient High Speed Computing with the Distributed Array Processor", in: Kuck, D. J., Lawrie D. H., Sameh, A. H., (eds.), "High Speed Computer and Algorithm Organization", Academic press, 113-128

Flynn, M. J. (1966): "Very High-Speed Computing Systems", Proc. IEEE, Vol.54, No.12, 1901-1909

Frank, A. J., Daniels, J. D., Unangst, D. R. (1980): "Progressive Image Transmission Using a Growth-Geometry Coding", Proc. IEEE, Vol. 68, No. 7, 597-909

Freeman, H. (1961): " On the Encoding of Arbitrary Geometric Configurations", IRE Trans. on Electronic Computers , Vol. EC-10

Freeman, H. (1974): "Computer Processing of Line-Drawing Images", Comput. Surveys 6, 57-97

Friehmelt, R., Nasraoui, A., Pecht, J., Stiller, P. (1985): " PACOS-I: An Interactive System for Development of Image Analysis Algorithms", Acta Stereologica (1986),Vol.5/2, 185-188

Gray, S. B. (1971): "Local Properties of Binary Images in Two Dimensions", IEEE Trans. Electron. Comput. 14, 434-443

Hadwiger, H. (1950): "Minkowski Addition und Subtraktion beliebiger Punktmengen und die Theoreme von Erhard Schmidt", Mathematische Zeitschrift, Band 53, Heft 3, 210-218

Huang, T. (1977): "Coding of Two-Tone Images", IEEE Trans. on Commun., Vol. COM-25, No. 11, 1402-1424

Kameswara Rao, C.V., Prasada, B., Sarma, K. R. (1976): "A Parallel Shrinking Algorithm for Binary Patterns", Comput. Graphics and Image Processing 5, 265-270

Kazmierczak, H. (1980): "Erfassung und maschinelle Verarbeitung von Bilddaten -Grundlagen und Anwendung", Springer Verlag, Wien

Kunt, M., Johnsen, O. (1980): "Block Coding of Graphics: A Tutorial Review", Proc. IEEE, Vol. 68, No. 7, 770-786

Kruse, B. (1973): "A Parallel Picture Processing Machine", IEEE Trans. Comput. C-22, No.12, 1075-1087

Laemmel, A. E. (1951): "Coding Processes for Bandwidth Reduction in Picture Transmission", Rep. R-246-51, PIB-187. Microwave Res. Inst., Polytechnic Institute of Brooklyn, New York

Levialdi, S. (1972): "On Shrinking Binary Picture Patterns", Comm. ACM 15, 7-10

Lutz, R. K. (1979): "An Algorithm for the Real Time Analysis of Digitized Images", Computer Journal Vol. 23, No. 3, 262-269

Märgner , V., Zamperoni, P. (1977): "Einige Experimente zur datenreduzierten Darstellung von digitisierten Mustern durch die Mittelachsen-Transformation", Informatik Fachberichte Nr. 8, Springer-Verlag, 212-222

Marr, D. (1982): "Vision", Freeman, San Francisco

Matheron, G. (1967): "Elêments pour une théorie des milieux poreux", Masson, Paris

Matheron, G. (1974): "Random Sets and Integral Geometry", Wiley, New York

Meyer, F. (1979): "Cytologie Quantitative et Morphologie Mathematique" Thèse de Docteur Ingenieur, Ecole des Mines de Paris

Minsky, M. L., Papert, S. (1969): "Perceptrons -An Introduction to Computational Geometry", MIT Press, Cambridge, Massachusetts

Montanari, U. (1969): "Continuous Skeletons from Digitized Images", J. ACM 16, 534-549

Morrin, T., H. (1976): "Chain-Link Compression of Arbitrary Black-White Images", Computer Graphics and Image Processing 5, 172-189

Mott-Smith, J. C., Baer, T. (1972): "Area and Volume Coding of Pictures", in: Huang, T. S. and Tretiak, O. H. (eds.), Picture Bandwidth Compression, Gordon and Beach, New York, 449-486

Musmann, H. G., Preuss, D. (1977): "Comparison of Redundancy Reduction Codes for Facsimile Transmission of Documents", IEEE Trans. on Commun. Vol. COM-25, No. 11, 1425-1433

Nasraoui, A. (1982): "Theorie und Implementierung eines Software-Paketes zur Unterstützung des Algorithmenentwurfs für Bitschichtrechner", Diplomarbeit, TU Braunschweig

Nasraoui, A. (1984): "Description and Coding of Binary Stochastic Images", in: Vollmar R. und Golze U. (eds.), Beiträge zur Theorie der Polyautomaten, Informatik-Skripten 8, TU Braunschweig, 90-105

Nasraoui, A., Vollath, D. (1985): "The Distance-Weighted Erosion and Dilation - A New Tool for Preprocessing of 3-D Images", Acta Stereologica (1986), Vol. 5/2, 189-194

Nasraoui, A. (1986): "Neighborhood Image Transformations and Their Application to Distortion Problems", wird bei Pattern

Recognition eingereicht

Nawrath, R., Serra, J. (1979): "Quantitative Image Analysis: Theory and Instrumentation", Microscopica Acta, Vol. 82, No. 2, 101-111

Niemann, H. (1983): "Klassifikation von Mustern", Springer Verlag, Berlin

Nilsson, N. J. (1982): "Principles of Artificial Intelligence", Springer-Verlag

Pavlidis, T. (1968): "Analysis of Set Patterns", Pattern Recognition, Vol.1, 165-178

Pavlidis, T. (1978): "A Minimum Storage Boundary Tracing Algorithm and Its Application to Automatic Inspection", IEEE Trans. on Systems Man Cybernet., Vol. SMC-8, No. 1, 66-69

Pavlidis, T. (1980): "A Thinning Algorithm for Discrete Binary Images", Comput. Graphics Image Processing 13, 142-157

Pavlidis, T. (1982): "Algorithms for Computer Graphics and Image Processing", Springer Verlag

Pecht, J. (1980): "Ein weiterer Ansatz zur Mustertransformation und -erkennung in Zellularen Räumen", Diss. TU Braunschweig

Pecht, J. (1981): "Entwurf von Bitschichtrechner-Algorithmen mit Hilfe von zellularen Netzen als Modelle", Vorlesungsmanuskript, TU Braunschweig

Pecht, J., Vollath, D., Gruber, P. (1983): "A Fast Bit Plane Processor for Quantitative Image Processing in a Mini-Computer Environment: Hardware and Software Architecture", in: Schüssler, H.W. (ed.), Signal Processing II: Theories and Applications, Proc. EUSIPCO-83, Elsevier Publishers B.V,

North-Holland, 809-812

Pecht, J. (1984): "PACOS.V -A Portable FORTRAN IV Package for Virtual Array Processing with Parallel Processors", in: Vollmar, R., Golze, U. (eds.), Beiträge zur Theorie der Polyautomaten, Informatik-Skripten 8, TU Braunschweig, 106-114

Pecht, J. (1985): "Speeding-up Successive Minkowski Operations with Bit-Plane Computers", Pattern Recognition Letters 3, 113-117

Pecht, J. (1986): Private Mitteilung

Pfaltz, J. L., Rosenfeld, A. (1967): "Computer Representation of Planar Regions by Their Skeletons", Commun. ACM, Vol. 10, No. 2, 119-125

Pratt, W. K. (1978): "Digital Image Processing", Wiley, New York

Pratt, W. K., Capitant, P. J., Wen-Hsiung Chen, Hamilton E. R., Wallis, R. H. (1980): "Combined Symbol Matching Facsimile Data Compression System", Proc. IEEE, Vol. 68, No. 7, 786-796

Preston, K., Duff, M. (1984): "Modern Cellular Automata - Theory and Application", Plenum Press, New York

Pugh, A. (ed.) (1983): "Robot Vision" Springer Verlag

Reeves, A. (1984): "Parallel Computer Architecture for Image Processing", Computer Vision, Graphics, and Image Processing 25, 68-88

Rosenfeld, A., Pfaltz, J. L. (1966): "Sequential Operations in Digital Picture Processing", J. ACM, Vol. 13, No. 4, 471-494

Rosenfeld, A. (1970): "Connectivity in Digital Pictures", J. ACM, Vol. 17, No. 1, 146-160

Rosenfeld, A. (1974): "Digital Straight Line Segments", IEEE Trans. on Comput., Vol. C-23, No. 12, 1264-1269

Rosenfeld, A. (1979): "Picture Languages: Formal Models for Picture Recognition", Academic Press, New York

Rosenfeld, A., Kak, A. C. (1982): "Digital Picture Processing", Academic Press, New York, London

Serra, J. (1969): "Introduction à la Morphologie Mathematique", Les Cahiers du Centre de Morphologie Mathematique, Fasc. 3, Fontainebleau

Serra, J. (1982): "Image Analysis and Mathematical Morphology", Academic Press

Shannon, C. E., Weaver, W. (1949): "The Mathematical Theory of Communication", Urbana Univ. Press

Sobel, I. (1978): "Neighborhood Coding of Binary Images for Fast Contour Following and General Binary Array Processing", Computer Graphics and Image Processing 8, 127-135

Stefanelli, R., Rosenfeld, A. (1971): "Some Parallel Thinning Algorithms for Digital Pictures", J. of th Assoc. for Comput. Mach., Vol. 18, No. 2, 255-264

Sternberg, S. R. (1978): "Cytocomputer Real-Time Pattern Recognition", Proc. 8th Auto. Imagery Pattern Recogn. Sym., 205-214

Stone, H.S. (1973): "Problems of Parallel Computation", in: Traub, J.F. (ed.), Complexity of Sequential and Parallel Numerical Algorithms, Academic Press, London, S. 1-16

Tuomenoksa, D. L., Adams III, G. B., Siegel, H. J., Mitchell, O. R. (1983): "A Parallel Algorithm for Contour Extraction:

Advantages and Architectural Implications", Proc. IEEE Comput. Soc. Conf. on Computer Vision and Pattern Recognition, Washington, DC, USA, 336-344

Unger, S. H. (1958): "A Computer Oriented Toward Spatial Problems", Proc. of IRE, Vol. 46, 1744-1750

Vollath, D. (1982): Das Bildanalysensystem PACOS:, Pract. Met. 19, Teil 1: S. 7-23, Teil 2: S. 94-103

Vollath, D., Friehmelt, R., Nasraoui, A., Pecht, J., Stiller, P. (1984): "PACOS - Ein preiswertes und flexibles System für die Bildverarbeitung", Kfk Nachrichten, Jahrgang 16, 4/84, 231-241

Vollmar, R. (1979): "Algorithmen in Zellularautomaten", Teubner Verlag, Stuttgart

Wolfowitz, J. (1978): "Coding Theorems of Information Theory", Springer Verlag, Berlin Heidelberg New York

Yamashita, M. (1985): "Parallel and Sequential Transformations on Digital Images", Pattern Recognition, Vol. 68, No. 1, 31-41

Zakharov, V. (1984): "Parallelism and Array Processing", IEEE Trans. on Comput., Vol. C-33, No. 1, 45-77

Zamperoni, P. (1980): "Analyse und Synthese von Binärbildern durch Zerlegung in Elementarmuster", Diss. TU Braunschweig

Zusne, L. (1970): "Visual Perception of Form", Academic Press, London

Anhang:

Im folgenden werden Zeichnungen, Bilder und Tabellen aufgelistet.

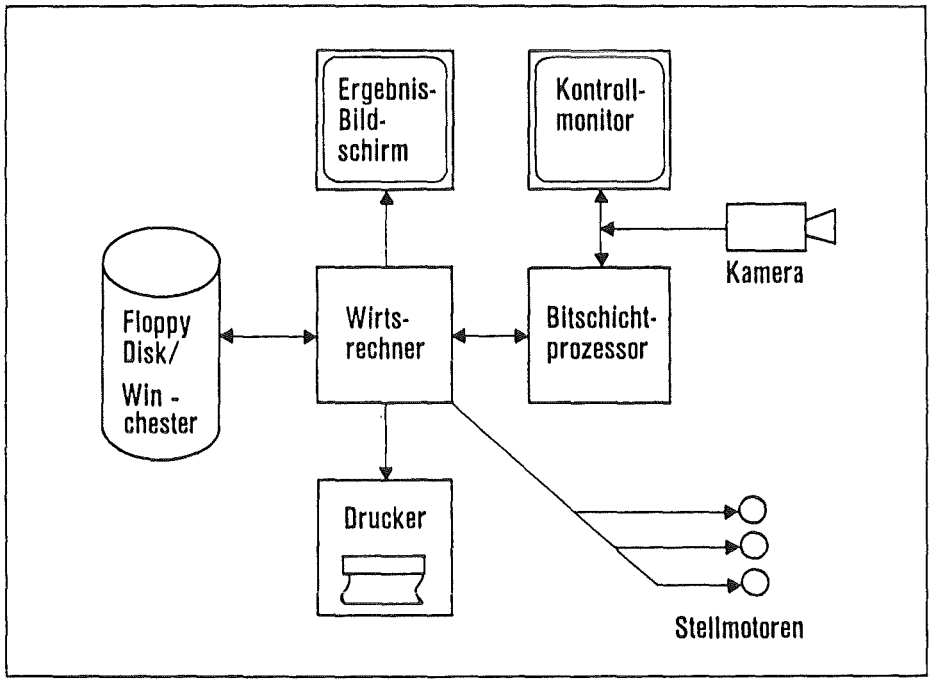

Fig. 2.1: Hardwarestruktur des Bildverarbeitungssystems PACOS

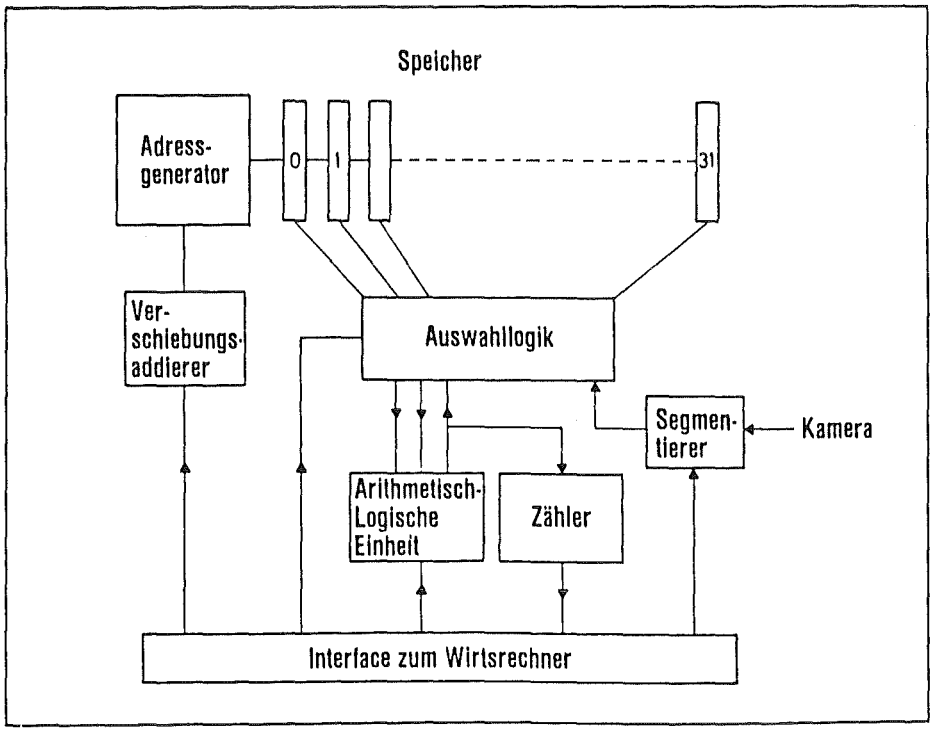

Fig. 2.2: Hardwarestruktur des Bitschichtrechners

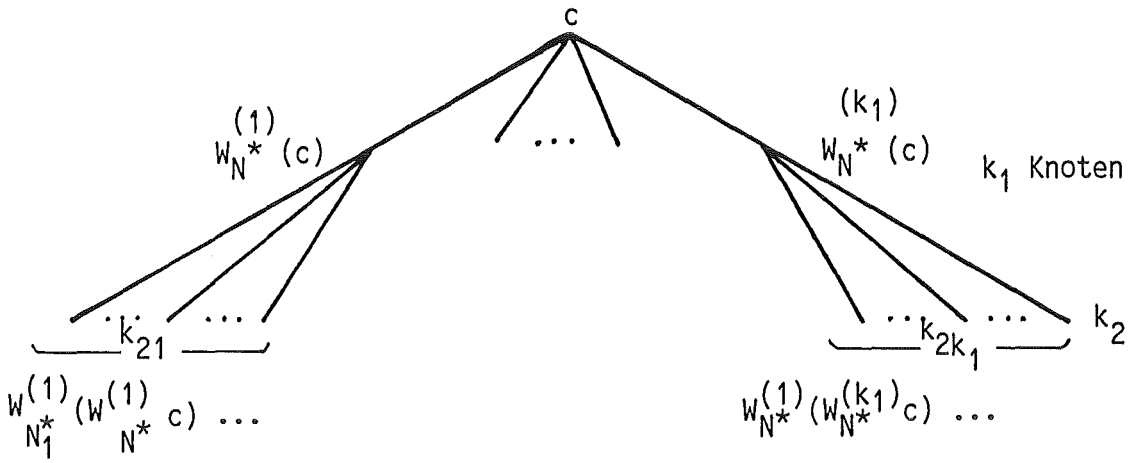

Fig. 3.1: Ein Suchbaum zur Ermittlung eines optimalen Rasters N^* und weitere Codierung der Wurzelzellen $W_{N^*1}^{(k_1)}$ c mit den jeweiligen optimalen Rastern N_1^*, N_2^* ,...

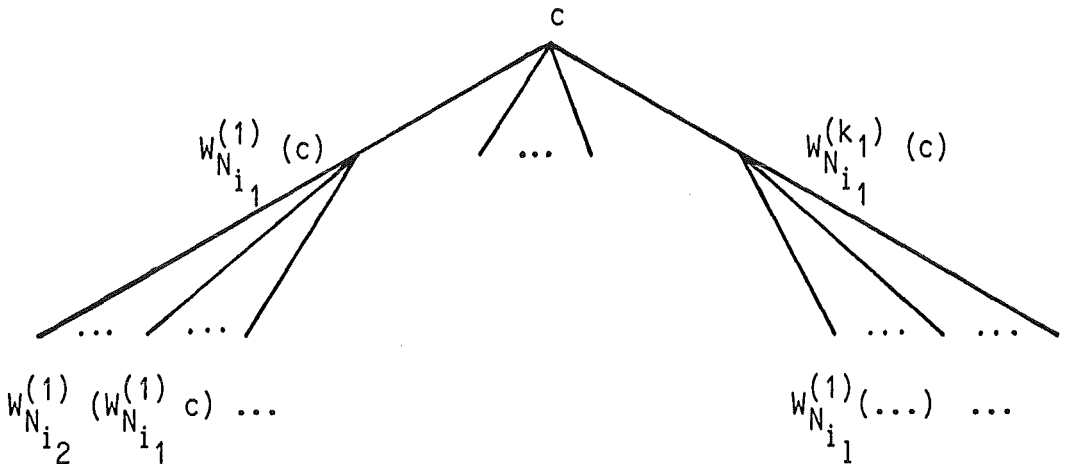

Fig. 3.2: Die ersten beiden Nachfolgergenerationen einer Baumstruktur zur optimalen Sequenz von Rastern N_{i_1}, N_{i_2},... mit der ein Binärbild c codiert werden kann.

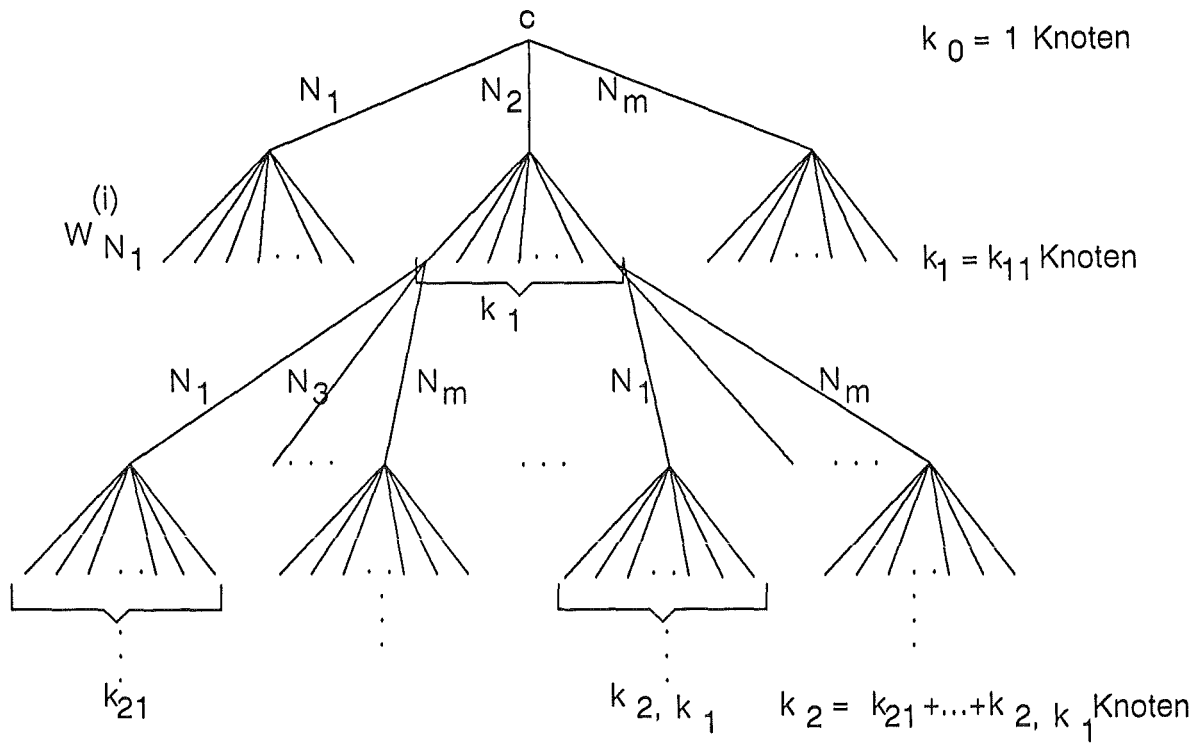

Fig. 3.1a: Ein Suchbaum zur Ermittlung der optimalen Raster für die Codierung von der Konfiguration c und deren Nachfolgerkonfigurationen.

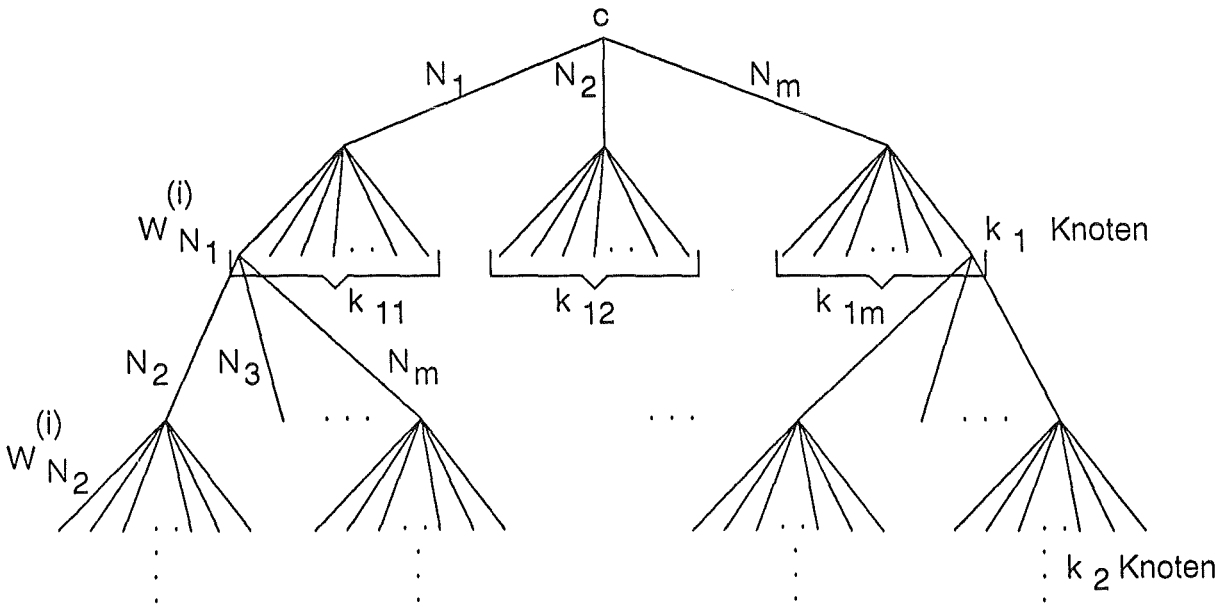

Fig. 3.2a: Ein Suchbaum zur Codierung einer Konfiguration c und deren Nachfolgerkonfigurationen mit Rastern N_1, N_2, ... , N_m.

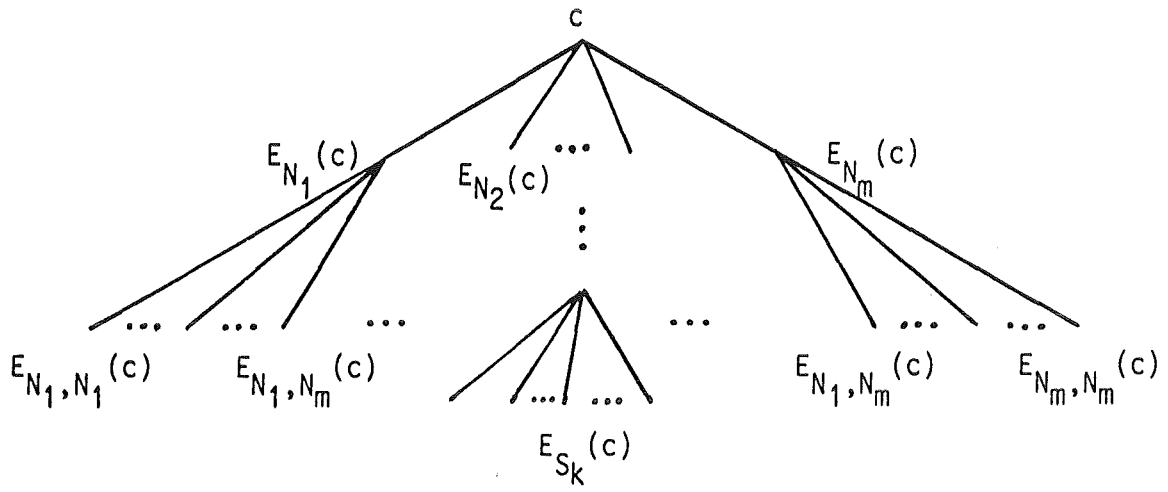

Fig. 3.3: Eine Baumstruktur als Modell des Ansatzes zur Ermittlung einer optimalen Sequenz von Rastern $S_k=(N_{i_1},\ldots,N_{i_k})$, die die Wurzelzellen W_{S_k} minimisiert

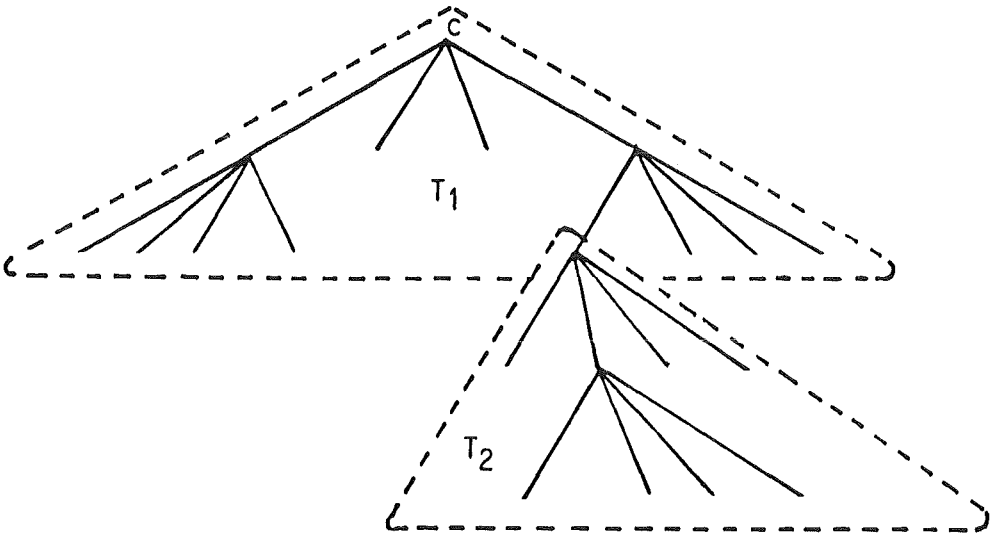

Fig. 3.5: Ein gleichmäßiges Suchen eines Baumes T. Die Tiefe der Unterbäume T_1 und T_2 (durch unterbrochene Linien eingegrenzt) ist dabei $d_1=2$. Dem optimalen Knoten werden $d'=1$ Knoten zugerechnet.

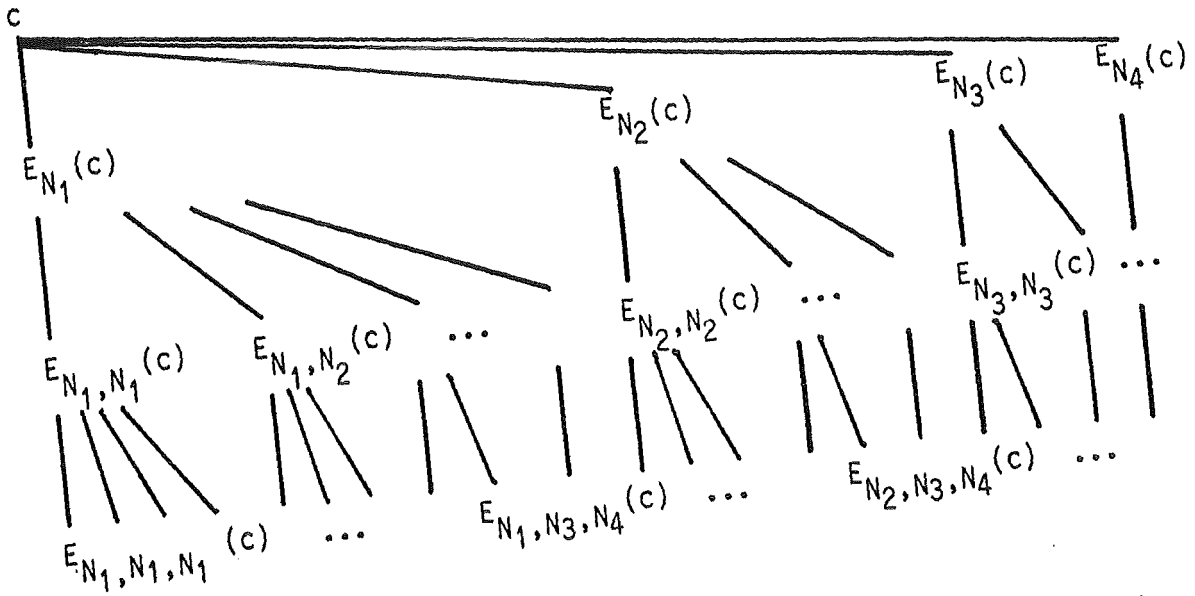

Fig. 3.4: Der Suchbaum T^* abgeleitet vom vollständigen Baum T bei einer Anzahl von Rastern m=4

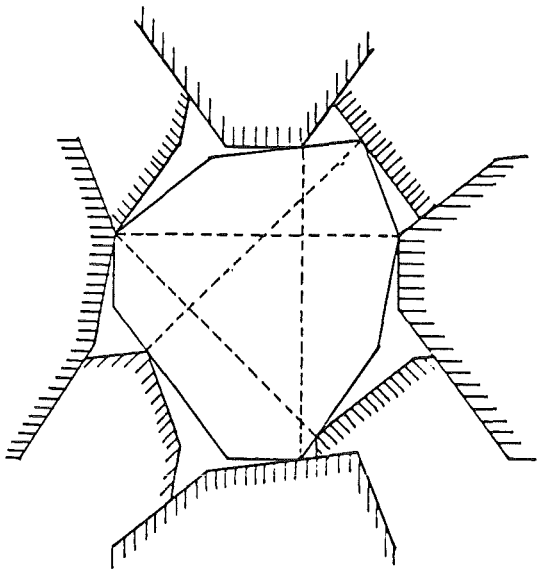

Fig. 4.1: Die auf Überlappungen von konvexen Objekten (hier das 8-seitige Polygon in der Bildmitte) zu überprüfenden Zonen (schraffiert)

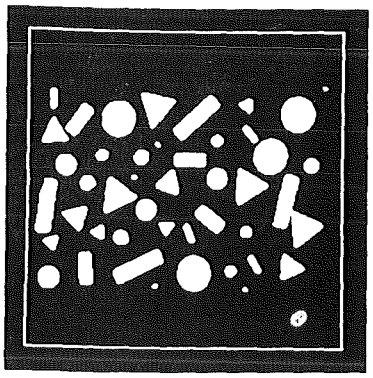

Fig. I

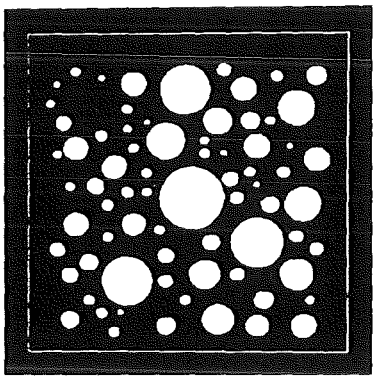

Fig. II

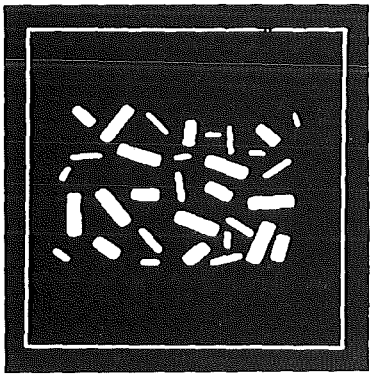

Fig. III

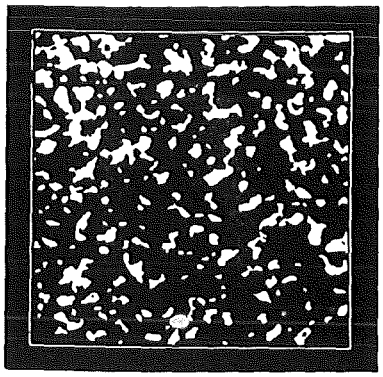

Fig. IV

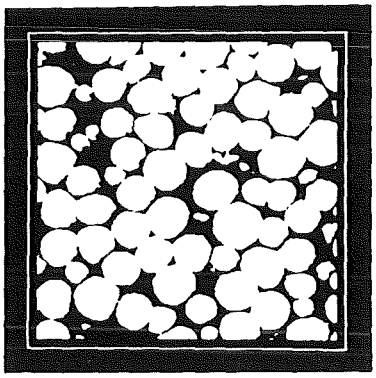

Fig. V

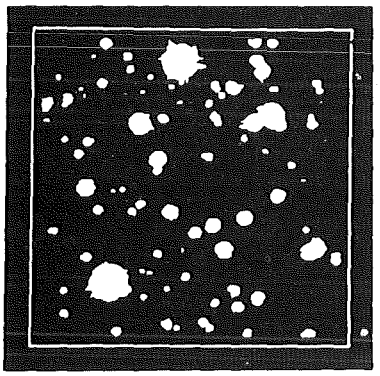

Fig. VI

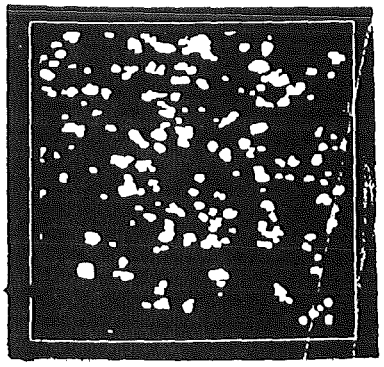

Fig. VII

Fig. I-III: Geometrische Modelle für Partikel unterschiedlicher Form

Fig. IV-VII: Verschiedene Proben aus der Materialforschung

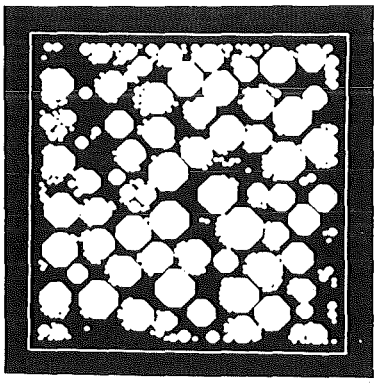

Fig. Va:

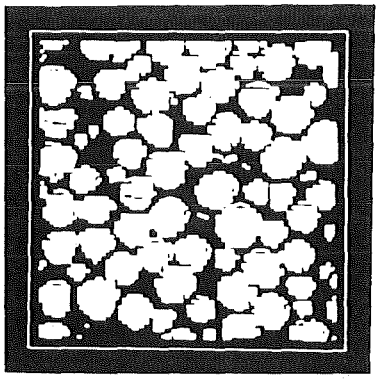

Fig. Vb:

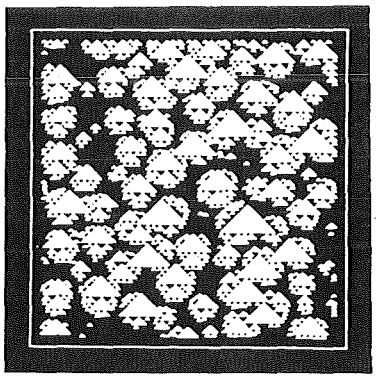

Fig. Vc:

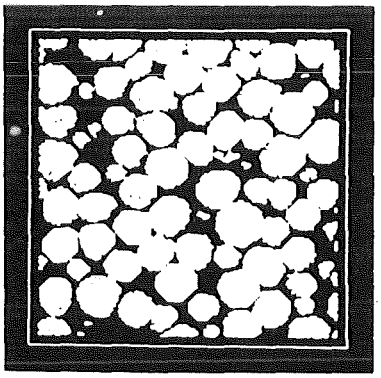

Fig VA:

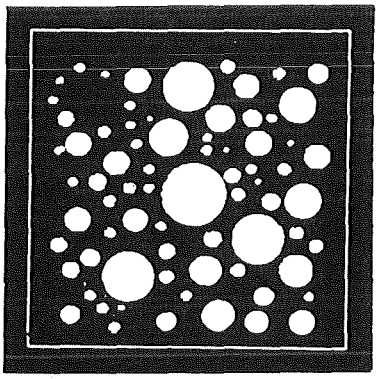

Fig. IIA:

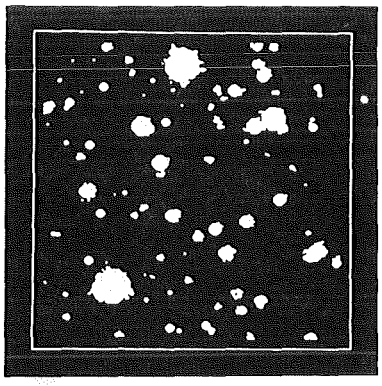

Fig. VIA:

Fig Va-c: Partielle Rekonstruktion des Bildes von Fig. V nach einer Partition in oktagonale (a), quadratische (b) und dreieckige (c) Muster.

Fig. IIA, VA, VIA: Partielle Rekonstruktion der Bilder von Fig. II, V, VI nach einer Partition in oktagonale Muster und eine Approximation des dabei entstandenen Restes.

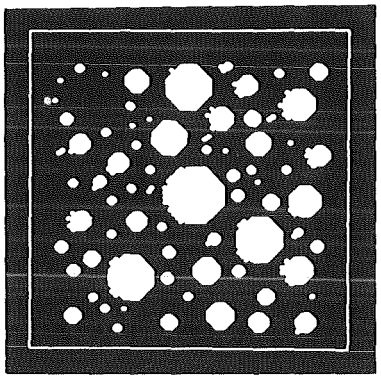

Fig. IIB

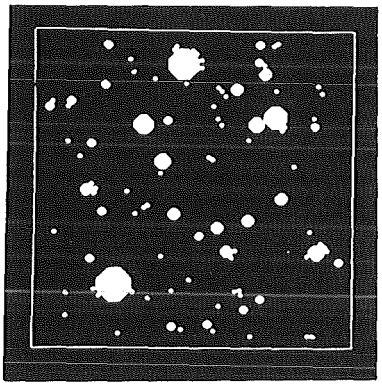

Fig. VIB

Fig. IIB, VIB: Partielle Rekonstruktion der Bilder von Fig. II und Fig. VI nach einer Partition in oktagonale Muster

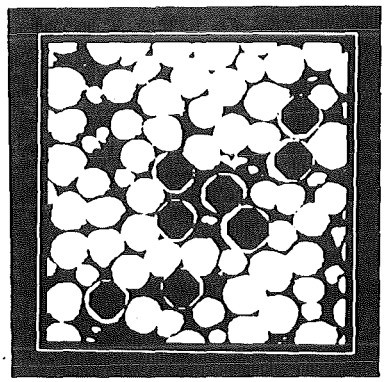

Fig. V1

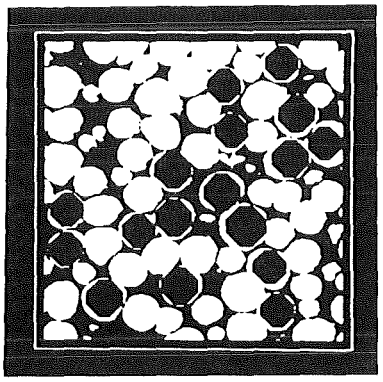

Fig. V1a

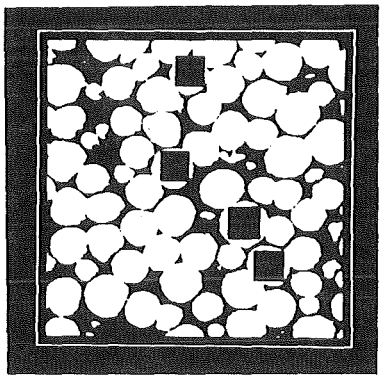

Fig. V2

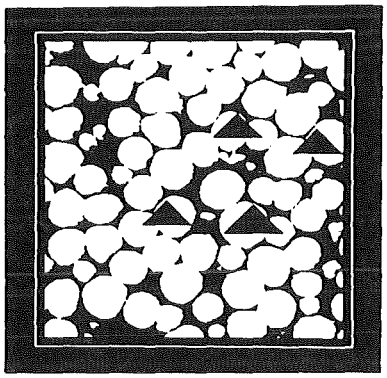

Fig. V3

Fig. V1-3: Klassifikation von überlappenden Objekten nach den größten einschreibbaren oktagonalen (1,1a), quadratischen (2) und dreieckigen (3) Mustern

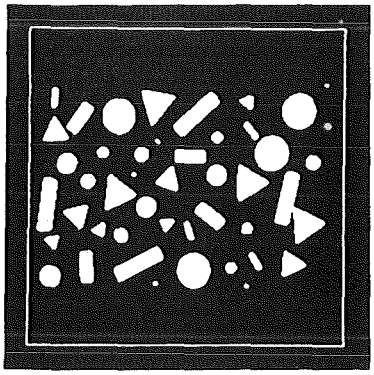

Fig. I

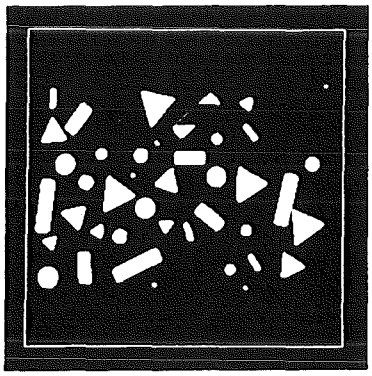

Fig. Ib

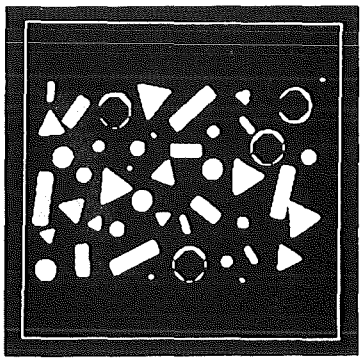

Fig. Ia

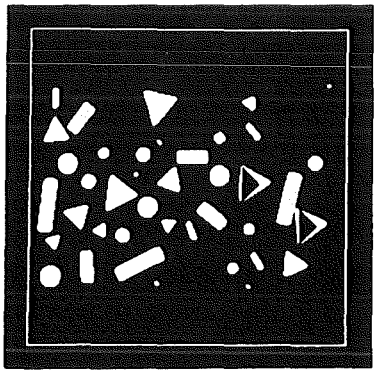

Fig. Ic

Fig. I, Ia-c: Klassifikation der Muster von Fig. I nach Größe und Form mit Hilfe der Raster von Fig. 4.2

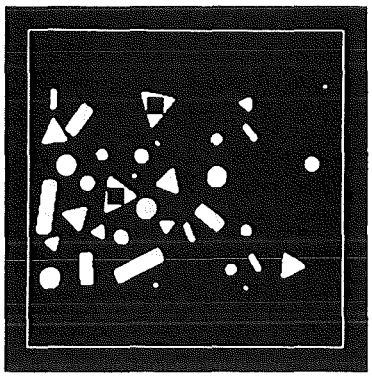

Fig. Id:

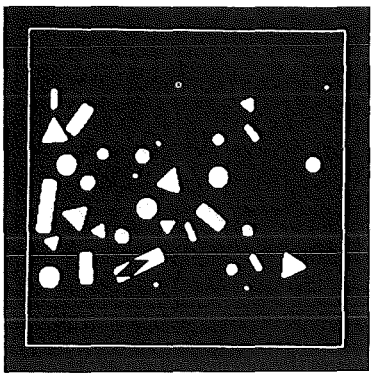

Fig. If

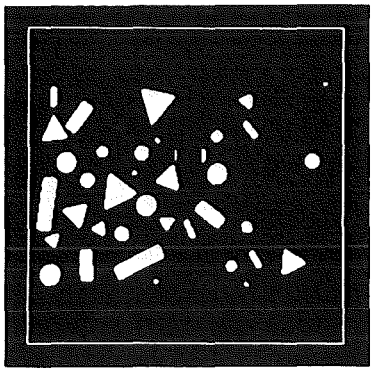

Fig. Ie

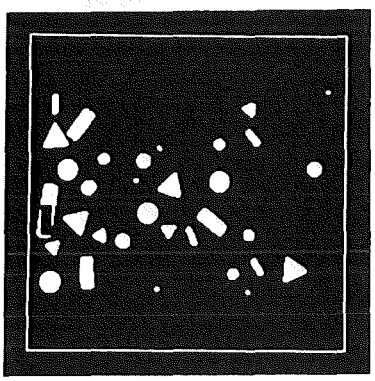

Fig. Ig

Fig. Id-g: Fortsetzung der Fig. Ia-c zur Klassifikation der Muster von Fig. I nach Größe und Form

Musterklassen	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
Muster/Klasse	1	7	11	14	12	7	7	8	13	37	276
Schritte	11	10	9	8	7	6	5	4	3	2	1
11	0	0	0	0	0	0	0	0	0	0	10
10	0	0	4	1	0	1	0	0	0	1	21
9	0	4	1	1	3	1	0	0	0	4	39
8	0	1	1	0	0	0	0	0	0	6	89
7	0	0	3	0	0	0	1	0	2	5	45
6	0	1	1	0	0	0	0	1	1	3	22
5	0	0	0	0	1	0	0	2	2	3	19
4	0	0	0	0	0	1	2	0	3	2	21
3	0	0	0	0	2	1	2	3	1	4	12
2	0	1	4	6	5	3	3	2	4	6	36
1	10	21	39	89	45	22	19	21	12	36	183
Ähnlichkeits-index * 10^2	92	61	74	88	84	81	68	69	80	80	60

Fig. 4.2: Nachbarschaftsmatrix bei der Partitionierung des Bildes von Fig. V in oktagonale Muster

Musterklassen	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13
Muster/Klasse	1	3	10	10	13	10	8	4	8	14	52	153	551
Schritte	13	12	11	10	9	8	7	6	5	4	3	2	1
13	0	0	0	0	0	0	0	0	0	0	0	4	5
12	0	0	1	1	0	1	0	1	0	0	1	5	16
11	0	1	0	0	3	0	0	0	0	2	9	22	45
10	0	1	0	0	0	0	1	0	0	0	5	19	46
9	0	0	3	0	0	0	0	0	1	3	7	23	43
8	0	1	0	0	0	1	0	0	1	1	3	13	33
7	0	0	0	1	0	0	0	0	1	0	2	4	23
6	0	1	0	0	0	0	0	0	0	0	2	4	6
5	0	0	0	0	1	1	1	0	0	1	3	5	11
4	0	0	2	0	3	1	0	0	1	0	5	10	16
3	0	1	9	5	7	3	2	2	3	5	11	27	39
2	4	5	22	19	23	13	4	4	5	10	27	51	111
1	5	16	45	46	43	33	23	6	11	16	39	111	290
Ähnlichkeits-index * 10^2	89	54	71	82	79	81	83	78	73	69	65	66	65

Fig. 4.3: Nachbarschaftsmatrix bei der Partitionierung des Bildes von Fig. V in quadratische Muster

Musterklassen	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
Muster/Klasse	1	3	8	9	13	14	19	29	41	111	628
Schritte	12	10	9	8	7	6	5	4	3	2	1
12	0	0	0	0	0	0	0	1	4	3	9
10	0	0	0	0	1	1	0	3	4	8	25
9	0	0	2	2	2	1	0	7	8	12	64
8	0	0	2	0	0	1	7	3	2	15	60
7	0	1	2	0	0	0	4	7	5	21	79
6	0	1	1	1	0	1	1	5	6	13	85
5	0	0	0	7	4	1	1	3	5	25	84
4	1	3	7	3	7	5	3	1	5	25	94
3	4	4	8	2	5	6	5	5	3	23	88
2	3	8	12	15	21	13	25	25	23	11	172
1	9	25	64	60	79	85	84	94	88	172	323
Ähnlichkeits-index * 10^2	66	57	53	57	60	61	61	64	64	67	66

Fig. 4.4: Nachbarschaftsmatrix bei der Partitionierung des Bildes von Fig. V in dreieckige Muster

Musterklassen	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
Muster/Klasse	1	3	1	2	3	3	8	7	11	20	75
Schritte	15	12	9	8	7	6	5	4	3	2	1
15	0	0	0	0	0	0	0	0	0	0	15
12	0	0	0	0	0	0	0	0	0	0	10
9	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0	0	0	6
7	0	0	0	0	0	0	0	0	0	0	5
6	0	0	0	0	0	0	0	0	0	0	3
5	0	0	0	0	0	0	0	0	0	0	15
4	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	2
2	0	0	0	0	0	0	0	0	0	0	2
1	15	10	0	6	5	3	15	0	2	2	44
Ähnlichkeits-index * 10^2	94	98	100	93	91	98	88	100	98	93	63

Fig. 4.5: Nachbarschaftsmatrix bei der Partitionierung des Bildes von Fig. II in oktagonale Muster

Musterklassen	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13
Muster/Klasse	1	1	2	1	2	3	1	10	3	10	23	66	232
Schritte	17	14	13	10	9	8	7	6	5	4	3	2	1
17	0	0	0	0	0	0	0	0	0	0	3	3	4
14	0	0	0	0	0	0	0	0	0	0	0	4	6
13	0	0	0	0	0	0	0	0	0	0	3	9	11
10	0	0	0	0	0	0	0	0	0	0	0	2	6
9	0	0	0	0	0	0	0	0	0	0	0	3	14
8	0	0	0	0	0	0	0	0	0	0	0	3	16
7	0	0	0	0	0	0	0	0	0	0	0	1	5
6	0	0	0	0	0	0	0	0	0	0	0	2	28
5	0	0	0	0	0	0	0	0	0	0	0	0	1
4	0	0	0	0	0	0	0	0	0	0	0	0	9
3	3	0	3	0	0	0	0	0	0	0	4	7	9
2	3	4	9	2	3	3	1	2	0	0	7	16	45
1	4	6	11	6	14	16	5	28	1	9	9	45	106
Ähnlichkeits-index * 10^2	87	89	82	87	86	87	84	90	98	94	74	70	68

Fig. 4.6: Nachbarschaftsmatrix bei der Partitionierung des Bildes von Fig. II in quadratische Muster

Musterklassen	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11
Muster/Klasse	1	2	1	1	4	5	3	13	29	47	273
Schritte	12	10	9	8	7	6	5	4	3	2	1
12	0	0	0	1	0	0	0	1	1	3	12
10	0	0	0	0	0	2	0	0	2	4	19
9	0	0	0	0	1	0	0	0	1	1	9
8	1	0	0	0	0	0	0	1	1	3	8
7	0	0	1	0	0	0	1	2	1	5	26
6	0	2	0	0	0	0	0	1	4	2	29
5	0	0	0	0	1	0	0	0	2	2	14
4	1	0	0	1	2	1	0	0	8	4	45
3	1	2	1	1	1	4	2	8	0	12	64
2	3	4	1	3	5	3	2	4	12	3	62
1	12	19	8	8	26	29	14	45	64	62	116
Ähnlichkeits-index * 10^2	51	61	53	59	63	66	65	61	70	72	70

Fig. 4.7: Nachbarschaftsmatrix bei der Partitionierung des Bildes von Fig. II in dreieckige Muster

Musterklassen	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Muster/Klasse	1	1	1	1	3	8	17	68
Schritte	9	8	6	5	4	3	2	1
9	0	0	0	0	0	0	0	3
8	0	0	0	0	0	0	0	6
6	0	0	0	1	0	0	0	3
5	0	0	0	0	0	0	0	0
4	0	0	1	0	0	0	0	3
3	0	0	0	0	0	0	0	6
2	0	0	0	0	0	0	0	3
1	3	6	3	0	3	6	3	16
Ähnlichkeits-index * 10^2	96	91	65	100	94	92	96	64

Fig. 4.8: Nachbarschaftsmatrix bei der Partitionierung des Bildes von Fig. VI in oktagonale Muster

Muster	Quadrat			Hexagon			Drei-eck	Oktagon							
Musterklassen	Q1	Q2	Q3	Q1	Q2	Q3	Q1	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Muster/Klasse	3	16	15	1	1	21	1	1	1	1	1	3	8	13	41
Schritte	3	2	1	3	2	1	1	9	8	6	5	4	3	2	1
3	0	0	0	0	0	0	1	0	0	0	0	0	0	0	1
2	0	0	5	0	0	1	0	0	1	1	0	1	0	0	1
1	0	5	0	1	0	1	0	3	2	1	0	0	1	1	3
3	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	1	1	0	2	0	3	2	1	1	2	4	0	10
1	1	0	0	0	0	0	0	0	0	0	0	0	1	0	0
9	0	0	3	0	0	3	0	0	0	0	0	0	0	0	3
8	0	1	2	0	0	2	0	0	0	0	0	0	0	0	4
6	0	1	1	0	0	1	0	0	0	0	0	1	0	0	0
5	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
4	0	1	0	0	0	2	0	0	0	1	0	0	0	0	3
3	0	0	1	0	0	4	1	0	0	0	0	0	0	0	5
2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	1
1	1	1	3	0	0	10	0	3	4	0	0	3	5	1	5
Ähnlichkeits-index * 10^2	91	23	2	82	100	2	4	93	88	65	98	53	90	97	6

Fig. 4.9: Nachbarschaftsmatrix bei der Partitionierung des Bildes von Fig. VI in quadratische, hexagonale, dreieckige und oktagonale Muster

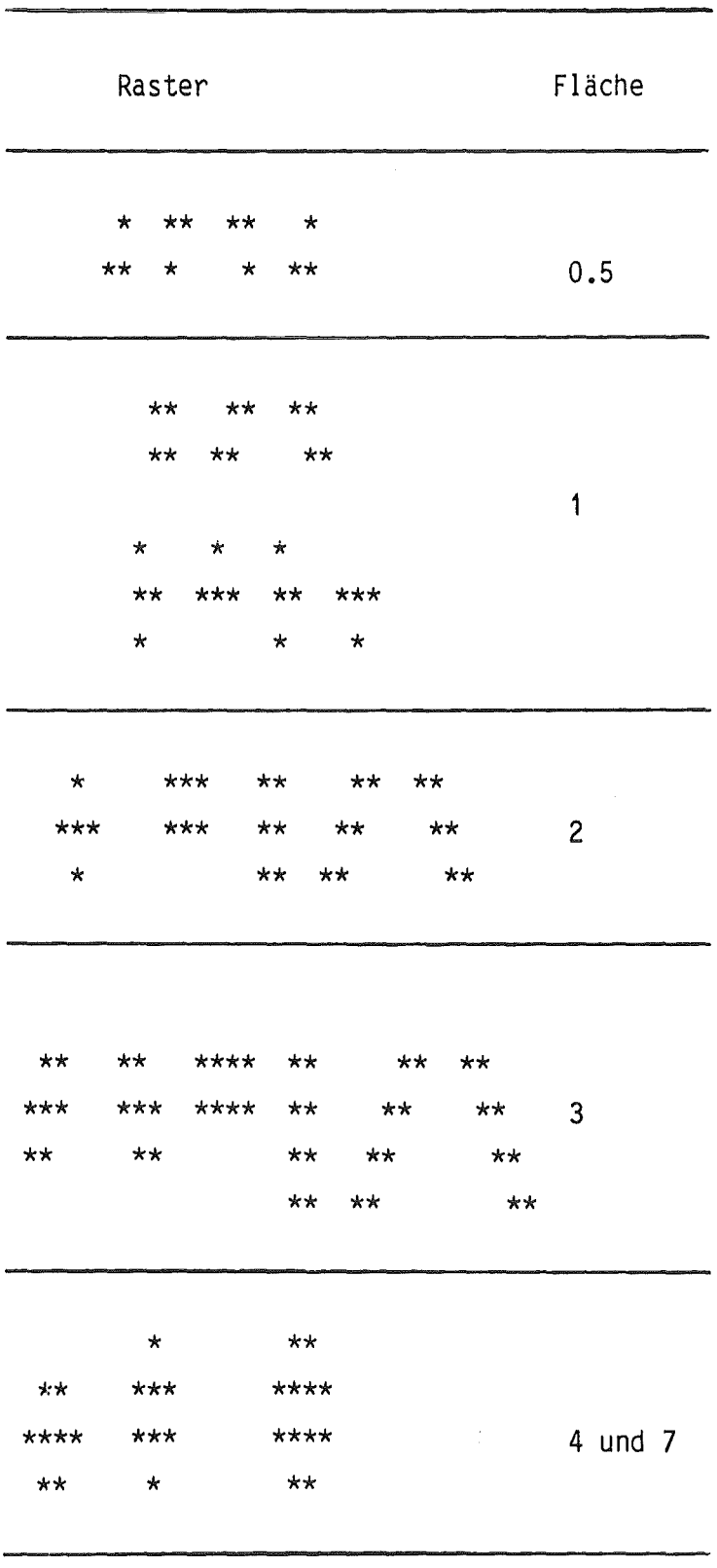

Fig. 4.10: Die zur Klassifikation der Muster von Fig. I herangezogene Menge von Rastern unterschiedlicher Form