Internal Report 95–06

Matchen von Kantenbildern mit einem dynamischen Neuronennetz

by

Michael Rinne

Ruhr-Universität Bochum
Institut für Neuroinformatik
44780 Bochum

IR-INI 95–06
Juni 1995
ISSN 0943-2752

Matchen von Kantenbildern mit einem dynamischen Neuronennetz

Michael Rinne[1], Institut für Neuroinformatik, Ruhr-Universität Bochum, FRG

[1]Tel. +49(0)234/700-7921, email Michael.Rinne@neuroinformatik.ruhr-uni-bochum.de

Kurzfassung

In dieser Arbeit wurde das Pyramidenmatching in neuronaler Dynamik implementiert. Hierbei wurde das Dynamic Link Matching mit der Dyadischen Wavelettransformation nach S. G. Mallat kombiniert. Die gewünschte Abbildung mit ihrer sehr großen Anzahl an Verbindungen (im Anwendungsbeispiel 2^{28} Links) wurde mit Hilfe einer Pyramide approximiert. Dabei wird auf den dünn besetzten Schichten eine Abbildung aufgebaut und diese durch die Dynamik der dichter besetzten Schichten verfeinert.

In dieser Arbeit konnte weiterhin gezeigt werden, daß es möglich ist auf eine explizite Ähnlichkeitsfunktions für die Merkmale zu verzichten. Statt dessen können die Merkmale direkt in die Dynamik der Neuronen eingehen und so die Bewegung der laufenden Blobs beeinflussen. Die Änderung dieser Aktivitätszentren bestimmt ihrerseits die Entwicklung der Abbildung.

Mittels der oben erwähnten Methoden konnten erfolgreich reale Bilder in einer mehrskaligen Kantenrepräsentation aufeinander abgebildet werden. Dabei bieten die Kantenbilder eine größere Beleuchtungsinvarianz.

Danksagung

Ich möchte mich an dieser Stelle bei Prof. Dr. Christoph von der Malsburg dafür bedanken, daß er mir diese sehr interessante Diplomarbeit am Institut für Neuroinformatik der Ruhr-Universität Bochum ermöglicht und mir mit seinen Anregungen stets weitergeholfen hat.

Für die Betreuung dieser Arbeit möchte ich mich bei Prof. W. Menzel und Dr. H. Braun vom Institut für Logik, Komplexität und Deduktionssysteme der Universität Karlsruhe bedanken.

Mein besonderer Dank gilt dem Dipl.-Math. Rolf Würtz, der durch seine Beratung und Diskussionsbereitschaft sehr viel zum Gelingen dieser Diplomarbeit beigetragen hat.

Ich danke allen Kollegen für die angenehme und anregende Arbeitsatmosphäre. Ein besonderer Dank geht an Uta Schwalm, Rolf Würtz, Stefan Zadel, Laurenz Wiskott und Tanja Zoppa für ihren Einsatz beim Korrekturlesen, ihre Hilfe, sowie ihre Geduld, die sie im Verlauf dieser Arbeit aufbrachten.

Inhaltsverzeichnis

Kapitel 1

Einleitung

1.1 Neurobiologische Grundlagen

In diesem Abschnitt soll auf die für diese Arbeit wichtigen und einige damit verbundenen biologischen Modellen eingegangen werden.

1.1.1 Das Sehsystem des Menschen

Das Sehsystem des Menschen läßt sich grob in den optischen Apparat des Auges, die Retina, die seitlichen Kniehöcker und den visuellen Cortex gliedern. Der optische Apparat bildet das einfallende Licht auf die Retina ab, in der eine erste Vorverarbeitung der Information stattfindet, bevor diese über den Sehnerv weitergeleitet wird.

Retina

Die Retina hat zwei wichtige Funktionen. Die erste ist die der Photorezeption. Hierzu gibt es zwei grundlegende Typen von Photorezeptoren, einmal die Zapfen, die für das Form- und Farbsehen bei Tageslicht verantwortlich sind, zum anderen die wesentlich häufigeren Stäbchen, die für das Sehvermögen bei geringer Helligkeit wichtig sind. Die Verteilung dieser Rezeptoren ist nicht einheitlich, vielmehr sind in der Fovea, dem Punkt des schärfsten Sehens, fast ausschließlich Zapfen konzentriert. Außerhalb der Fovea sind die Zapfen fast gleichmäßig über die Retina verteilt, die Konzentration der Stäbchen nimmt mit zunehmender Entfernung von der optischen Achse ab. Die zweite wichtige Aufgabe der Retina besteht in der Vorverarbeitung der rezipierten visuellen Information. Hierzu verfügt die Retina neben Rezeptorzellen über eine komplexe Maschinerie von Interneuronen. Die Interneurone können in vier Klassen eingeteilt werden:

- Horizontalzellen
- Bipolarzellen
- amakrine Zellen
- Ganglienzellen

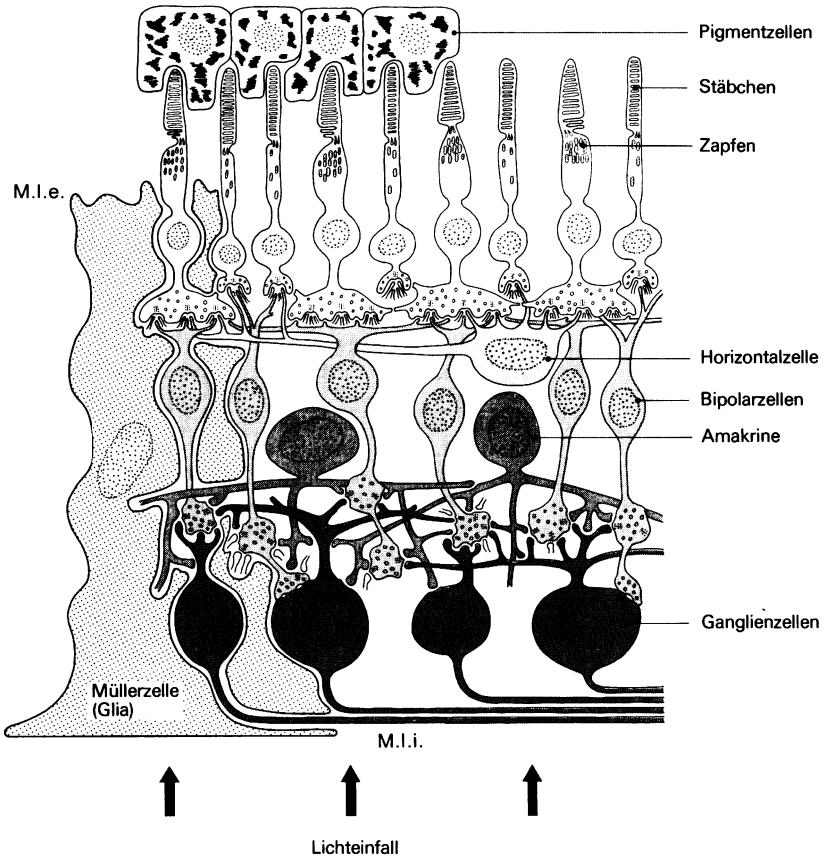

Abbildung 1.1: Schematischer Aufbau der Retina, aus [ST80].

Die Rezeptorzellen besitzen Synapsen mit den Horizontal- und Bipolarzellen. Eine Bipolarzelle reagiert nicht nur auf die direkt präsynaptischen Photorezeptoren, sondern auch auf benachbarte Rezeptoren, über die dazwischengeschalteten Horizontalzellen. Das rezeptive Feld der Bipolarzelle wird somit vergrößert. Bedingt durch die Art der lateralen Interaktionen durch die Horizontalzellen entsteht bei den Bipolarzellen ein Umfeldantagonismus. Zentrum und Umfeld üben funktionell umgekehrte, anatagonistische Wirkungen auf die Zelle aus. Es kommt zur Bildung von sogenannten ON- bzw. OFF-Zentrum-Zellen. Die Bipolarzellen sind über Synapsen mit den Ganglienzellen verbunden. Der meist konzentrisch ausgebildete Umfeldantagonismus der Bipolarzellen wird an postsynaptische Ganglienzellen weitergegeben. Die Eigenschaften der Ganglienzellen werden jedoch durch zusätzliche laterale Interaktionen, die durch die amakrinen Zellen vermittelt werden, noch modifiziert. Die Ganglienzellen lassen sich in mehrere Unterklassen einteilen:

- α-Zellen haben große rezeptive Felder und reagieren zum Teil auf Bewegungen.
- β-Zellen haben kleine rezeptive Felder und reagieren auf stationäre Beleuchtung. Etwa 80% der Ganglienzellen sind von diesem Typ.
- γ-Zellen liefern Information für Kopf- und Augenbewegungen.

Sehbahn

Die Ganglienzellen in den äußeren (*temporalen*) Hälften jeder Retina projizieren ihre Information über den Sehnerv in die Gehirnhälfte, die auf der gleichen Seite wie das Auge liegt (*ipsilaterale* Projektion), diejenigen in den inneren (*nasalen*) Hälften projizieren in die andere Gehirnhälfte (*kontralaterale Projektion*). Die geordnete Aufspaltung der Ganglienaxone geschieht beim Eintritt des optischen Nerven in das Gehirn, am optischen Chiasma. Dies bewirkt, daß Information aus der rechten Hälfte des Sehfeldes von der linken Gehirnhälfte verarbeitet wird und umgekehrt. Die meisten Ganglienzellen senden ihre Axone in die seitlichen Kniehöcker. Die seitlichen Kniehöcker sind in sechs Schichten unterteilt. Dabei erhalten die Schichten eins, vier und sechs nur Information aus der kontralateralen Retina und die Schichten zwei, drei und fünf nur aus der ipsilateralen. Die Sehbahn verläuft weiter zum primären visuellen Cortex. Dabei erhalten benachbarte Neurone Information über benachbarte Bereiche des visuellen Feldes. Die Abbildung 1.2 veranschaulicht den Verlauf der Sehbahn.

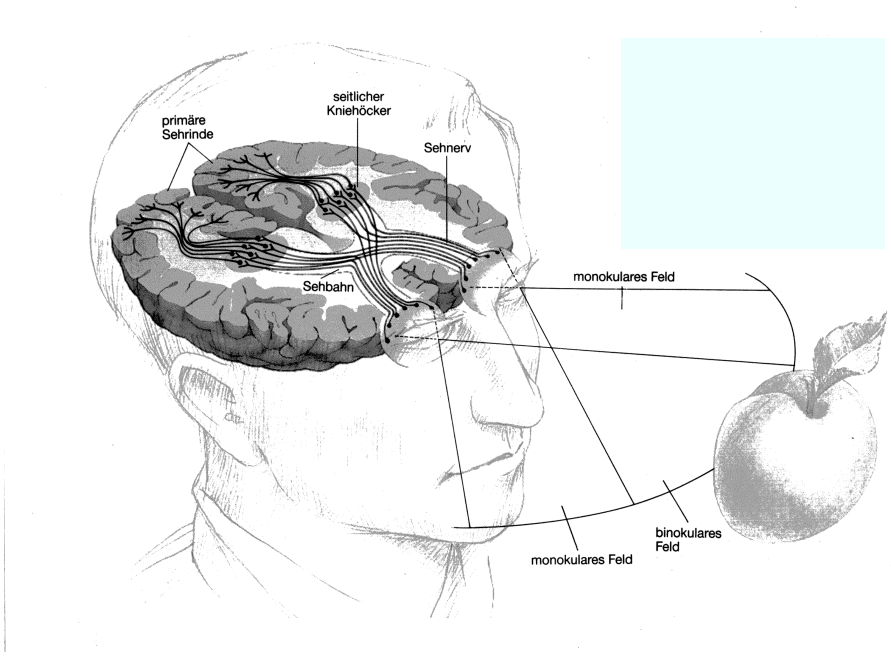

Abbildung 1.2: Verlauf der Sehbahn, aus [Sha92].

Die Abbildung der Retina auf den visuellen Cortex ist nachbarschaftserhaltend, das heißt benachbarte Neurone der Retina werden auf benachbarte Zellen im viusellen Cortex abgebildet. Derartige nachbarschaftserhaltende Abbildungen gehören zu den wichtigen Ordnungsprinzipen in natürlichen Nervensystemen. Die Frage nach der Entwicklung dieser Abbildungen ist von besonderer Bedeutung. Ein Modell, das sich mit der Retinotopie, der nachbarschaftserhaltenden Abbildung der Retina auf den Cortex befaßt wurde von v. d. Malsburg 1973 vorgestellt, siehe [vdM73]. Die nachbarschaftserhaltenden Abbildungen werden in den nachfolgenden Kapiteln eingehender behandelt, siehe beispielsweise Abschnitt 2.3 bzw. 2.4.

Primärer visueller Cortex

Der primäre visuelle Cortex ist, parallel zur Oberfläche, in sechs Schichten unterteilt. Die Axone aus den seitlichen Kniehöckern enden hauptsächlich in der Schicht vier, von wo aus die Information in die anderen Schichten gelangt. Die Neurone der Schicht vier erhalten monokulare Information, die Neurone der übrigen Schichten binokulare Information, wobei die Topographie erhalten bleibt. Die sogenannten einfachen Zellen reagieren besonders gut auf einen Balken in bestimmter Orientierung. Der visuelle Cortex ist auch senkrecht zur Oberfläche in zwei Richtungen geordnet, diese vertikalen Schichten werden als Säulen bezeichnet. In der einen Richtung sind die Säulen nach der okulären Dominanz geordnet, das heißt es dominieren entweder die Signale aus dem rechten oder linken Auge. Die Ordnung der anderen Richtung bezieht sich auf die Orientierung der Balken, durch die eine Zelle erregbar ist. Man sprich von Orientierungs-Säulen. Benachbarte Säulen haben dabei eine ähnliche Orientierung, es handelt sich wieder um eine nachbarschaftserhaltende Abbildung. In Abbildung 1.3 wird nochmals der Verlauf der Sehbahn gezeigt und die Organisation des primären visuellen Cortex dargestellt.

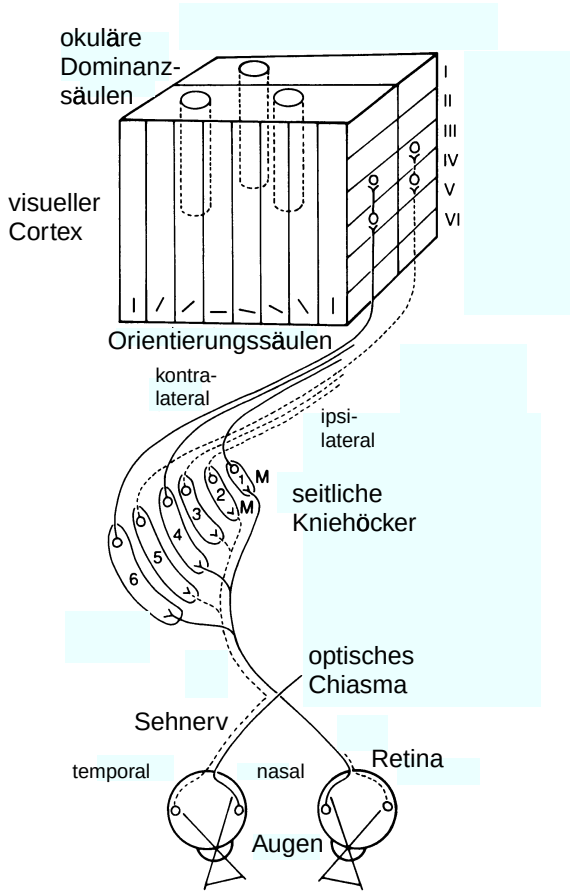

Abbildung 1.3: Verlauf der Sehbahn und Organisation des primären visuellen Cortex, aus [She88].

Die beschriebenen Organisationsprinzipien des visuellen Cortex und die Eigen-

schaften seiner einfachen Neurone legen die Vermutung nahe, daß Linien und Kanten für die Verarbeitung der visuellen Information beim Menschen von großer Bedeutung sind. Es gibt Untersuchungen, die zeigen, daß sich die rezeptiven Felder der einfachen Neurone angemessen durch eine zweidimensionale Gaborfunktion beschreiben lassen, wie sie beispielsweise für die Vorverarbeitung der visuellen Information, in dem im Abschnitt 2.4.3 beschriebenen Verfahren, verwendet wird. In dieser Arbeit ist der Schwerpunkt auf die Kanten gelegt worden. Wie im Kapitel 3 beschrieben können Bilder relativ robust durch die in ihnen vorkommenden Kanten beschrieben werden. Soll entschieden werden, ob ein abgebildetes Objekt mit einem bekannten identisch ist, so kann dies modelliert werden, indem eine nachbarschaftserhaltende Abbildung zwischen den beiden Kantenrepräsentationen der Objekte aufgebaut wird. In Abhängigkeit der resultierenden Abbildung kann eine Entscheidung über die Beziehung zwischen den Objekten getroffen werden.

1.2 Das Matching-Problem

In seiner einfachsten Definition ist das Matching-Problem ein reines Zuordnungsproblem, bei dem es darum geht, nach bestimmten Kriterien Paare von Objekten zu finden. Ist ein beliebiger Graph gegeben, so ist ein Matching eine Teilmenge der Kanten, in der kein Knoten zweimal vorkommt. Abbildung 1.4 zeigt ein Beispiel zu dieser Definition, die aus [Sed88] stammt.

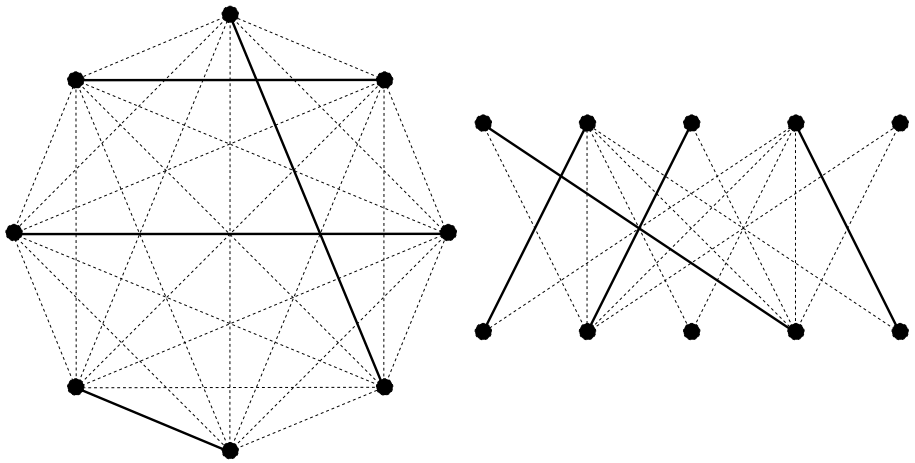

Abbildung 1.4: Matching für Knoten von Graphen.

Wie sich im rechten Teil der Abbildung 1.4 erkennen läßt, gibt es auch für dieses einfache Zuordnungsproblem bereits Fälle, bei denen es nicht möglich ist, alle Objekte zu Paaren zusammenzufassen. Das rechte Teilbild zeigt darüber hinaus einen zweiseitigen Graphen (auch bipartiter Graph genannt). Dies ist ein Graph, bei dem sich die Objekte so in zwei Teilmengen zerlegen lassen, daß es keine Kante zwischen zwei Knoten in der gleichen Teilmenge gibt. Die beiden Teilmengen werden in diesem Beispiel durch die obere bzw. untere Knotenreihe gebildet. Ein solcher Graph ist typisch für die Probleme in unserem Alltag. Ein gutes Beispiel hierfür ist die Zuordnung von Studenten zu Studienplätzen für das Studium der Informatik durch das Verteilungsverfahren der **Z**entralen **V**ergabestelle für **S**tudienplätze. Dies Beispiel

führt uns auch sogleich von dem einfachen Paarungsproblem zu dem Matching-Problem für markierte Graphen. Bei der ZVS darf jeder Student mehrere Universitäten angeben, an denen er bevorzugt studieren würde. Die Universitäten haben ihrerseits die Möglichkeit, sich aus den vielen Bewerbern einen bestimmten Prozentsatz durch ein Benennungsverfahren auszuwählen. Für weitere Komplikationen wird gesorgt, in dem die Wünsche bestimmter Studentengruppen bevorzugt berücksichtigt werden. Die weitverbreitete Unzufriedenheit unter den Erstsemestern belegt, welch geringer Prozentsatz dieser zahlreichen Wünsche erfüllt werden kann. In den Graphen werden die Wünsche repräsentiert, indem sie den Kanten als Merkmal angeheftet werden, so entstehen sogenannte markierte Graphen. Bei eindimensionalen Merkmalsvektoren lassen sich die Merkmale als die Länge der Kanten interpretieren. In diesem Sinne sind die beiden Lösungen, die in Abbildung 1.4 vorgeschlagen wurden schlechte Lösungen, da die Summe der einzelnen Kantenlängen zu groß ist. Abbildung 1.5 zeigt für beide Probleme eine mögliche optimale Lösung.

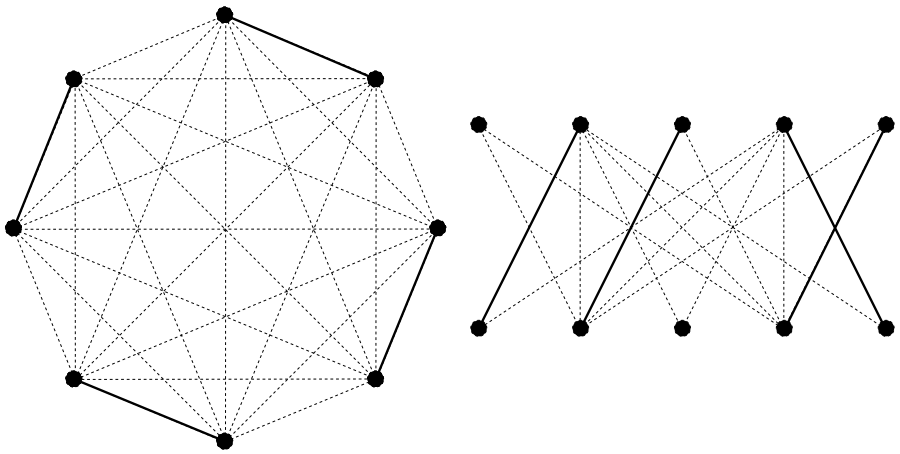

Abbildung 1.5: Optimale Lösungen für markierte Graphen, wobei die Kosten für eine Kante durch deren Länge ausgedrückt wird.

Beim Matchen zweier Bilder kommt ein weiteres Problem hinzu. Die Abbildung soll nachbarschaftserhaltend sein, das heißt, benachbarte Bildpunkte in dem einen Bild sollen mit benachbarten Punkten in dem anderen Bild verbunden werden. Diese Forderung steht in der Regel im Gegensatz zu der Maximierung der Ähnlichkeiten für die Paarungen. Sollen zwei Bilder in ihrer Repräsentation durch zwei gleichgroße Gitter mit Grauwerten miteinander gematcht werden, so gibt es bei einer strengen Nachbarschaftsbeziehung nur acht mögliche Abbildungen. Diese acht Abbildungen bestehen aus der Identität, drei Drehungen und den vier Spiegelungen, sie sind in Abbildung 1.6 dargestellt. Bei derartigen Forderungen kann eine Lösung des Problem leicht durch Ausprobieren dieser acht möglichen Abbildungen gefunden werden.

Sollen reale Bilder gematcht werden, so kann die Forderung nach einer strengen Nachbarschaftserhaltung nicht aufrecht erhalten werden, denn es müssen beispielsweise auch Drehungen um beliebige Winkel möglich sein. Die geeignete Kodierung der Nachbarschaft gehört bereits zu dem Matching-Problem. Bei verschiedenen Algorithmen kann die gleiche Nachbarschaftsbeziehung sowohl vorteilhaft als auch

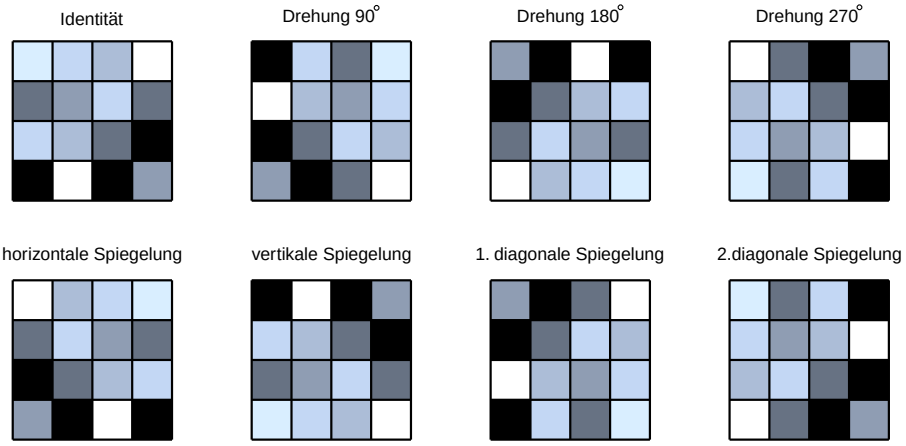

Abbildung 1.6: Die acht möglichen Abbildungen bei strenger Nachbarschaftsbeziehung.

nachteilig sein. Bei dem Vergleich der einzelnen Algorihmen sollte dies stets im Auge behalten werden. Die verschiedenen Verfahren sollten in der Lage sein, reale Bilder aufeinander matchen zu können und sich nicht die Aufgabe vereinfachen, indem sie ausschließlich die acht obengenannten Abbildungen erlauben.

Zusammenfassend läßt sich das im weiteren betrachtete Matching-Problem charakterisieren als ein Paarungsproblem für markierte, bipartite Graphen unter Maximierung der Summe der Ähnlichkeiten der gepaarten Objekte und gleichzeitiger Erhaltung einer gewissen Nachbarschaftsbeziehung, in der diese Objekte stehen.

Kapitel 2

Bisherige Arbeiten

In diesem Kapitel soll anhand einer Auswahl von neuronalen Modellen die generelle Problematik des Matching von Bildern erläutert werden. Weiterhin wird das Dynamic Link Matching vorgestellt, das mit einer in Kapitel 4 beschrieben Erweiterung als Grundlage für das in dieser Arbeit vorgestellte Matching-Verfahren dient. Für allgemeinere Darstellungen von Neuronalen Netzen sei auf die Bücher [HKP91], [HN90] und [Bra91], sowie die Diplomarbeit [Zad91] verwiesen.

2.1 Assoziativspeicher

Bei einem assoziativen Speicher wird der Inhalt y^k nicht unter Adressen abgelegt, sondern es werden Tupel (x^k, y^k) gespeichert, wobei die x^k Schlüssel sind, die mit dem Inhalt y^k assoziiert werden sollen. Für die Speicherung der Tupel (x^k, y^k) gibt es mehrere Möglichkeiten, von den eine Auswahl nachfolgend besprochen wird. Zu erwähnen sind noch die konventionellen Assoziativspeicher, die im wesentlichen aus einem **R**andom **A**ccess **M**emory und einer Vergleichslogik bestehen. Beim Auslesen wird der Inhalt des Suchregisters an den mit eins markierten Bitpositionen mit den entsprechenden Bits der Speicherzellen verglichen. Der Vergleichsindikator zeigt an, welche Speicherzellen dem Schlüssel, der durch Such- und Maskenregister präsentiert wird, entsprechen. Der hardwaremäßige Aufwand ist relativ groß. Außerdem ist das System nicht in der Lage, Schlüssel mit leichten Fehlern korrekt zu assoziieren. Die Stärke der neuronalen Modelle liegt gerade in dieser Fehlerkorrektur der Schlüssel durch Ausgabe des richtigen oder zumindest eines ähnlichen Musters.

2.1.1 Korrelationsmatrix-Modell

Bei dem Korrelationsmatrix-Modell wird die Zuordnung zwischen dem Schlüssel x^k und dem Inhalt y^k durch eine Gewichtsmatrix $W = (w_{ij})$ modelliert. Der Inhalt y^k zum Schlüssel x^k läßt sich mittels

$$y_i^k = \sum_j w_{ij} x_j^k \tag{2.1}$$

berechnen. Dabei sei y_i^k die i-te Komponente des k-ten Mustervektors $y^k = (y_1^k, y_2^k, ..., y_n^k)$. Die einzelnen w_{ij} lassen sich als Stärke der Verbindung zwischen

den Komponenten x_i und y_i interpretieren.

Für das Speichern der Tupel (x^k, y^k) wird eine Hebbsche Lernregel

$$\Delta w_{ij} = y_i^k x_j^k \tag{2.2}$$

verwendet. Genaugenommen handelt es sich hierbei weniger um eine Lernregel, als um ein iteratives Verfahren, das nach dem Speichern von n Tupel zur folgenden Gewichtsmatrix führt:

$$w_{ij} = \sum_k y_i^k x_j^k \ . \tag{2.3}$$

An dieser Darstellung kann sofort abgelesen werden, daß das Modell für orthonormale Vektoren die Assoziation korrekt durchführt. In diesem Fall können höchstens n Tupel gespeichert werden, wobei n der Dimension der Schlüssel x^k entspricht.

Wird die Gewichtsmatrix mit folgender Formel gebildet, so lassen sich n linear unabhängige Vektoren speichern:

$$W = YX^+ \tag{2.4}$$

wobei $Y = (y^1 y^2 \cdots y^m)$ eine $n \times m$ Matrix ist, deren Spalten aus den Vektoren y^k gebildet werden und X^+ die Pseudoinverse der Matrix $X = (x^1 x^2 \cdots x^m)$ ist, welche sich durch die Eigenschaften

$$\begin{aligned} XX^+X &= X \ , \\ X^+XX^+ &= X^+ \ , \\ XX^+ &= (XX^+)^T \ , \\ X^+X &= (X^+X)^T \end{aligned}$$

charakterisieren und nach der Gleichung

$$X^+ = X^T(X^TX)^{-1} \tag{2.5}$$

berechnen läßt. Die Pseudoinverse kann durch ein neuronales Netz mittels einer sogenannten Delta-Lernregel approximiert werden:

$$\Delta w_{ij} = \delta(y_i^k - \sum_j w_{ij} x_j^k) x_j^k \ . \tag{2.6}$$

Korrelationsmatrix-Modell mit Schwellwert

Untersuchen wir nun das Verhalten der Korrelationsmatrix bei Präsentation eines Schlüssels $x := x^m + x^r$, der aus der Überlagerung des Schlüssels x^m und einem Rauschen x^r entstanden ist. Die Ausgabe des Assoziativspeichers sieht nun folgendermaßen aus

$$y = Wx = W(x^m + x^r) =: y^m + y^r \ . \tag{2.7}$$

Dies Ergebnis ist als die Ausgabe einer Überlagerung des gewünschten Inhalts y^m mit dem Rauschen y^r zu interpretieren.

Reduzieren wir den Wertebereich der Komponenten der Vektoren x und y auf die Werte -1 und $+1$, so ist es möglich, die Ausgabe durch eine Schwelle zu verbessern, die als zusätzliches Gewicht W_{i0} gelernt wird und durch eine zusätzliche Schlüsselkomponente x_0 angesprochen wird, die konstant den Wert 1 annimmmt. Hierbei berechnet sich die Ausgabe mittels

$$y_i = \sigma(\sum_j w_{ij} x_j) \tag{2.8}$$

wobei σ eine Schwellwertfunktion ist, die negative Werte auf -1 und positive auf $+1$ abbildet.

Durch die Verwendung der Schwelle zerfällt der Eingaberaum in Klassen, deren Prototypen den abgespeicherten Mustern entsprechen. Bei der Assoziation des Schlüssels x^k mit dem Inhalt y^k wird eine Klassifikation vorgenommen. Da die Schlüssel auch als Eingabemuster und die Inhalte als Ausgabemuster bezeichnet werden, spricht man bei dieser Klassifikation auch von einer Mustererkennung oder vom Pattern-Matching. Wird die Korrelationsmatrix als Autoassoziator betrieben, das heißt $x^k = y^k \quad \forall k \in \{1, 2, ..., n\}$, so werden fehlerhafte oder unvollständige Muster korrigiert bzw. ergänzt.

Wir wollen nun die Eigenschaften dieses sehr einfachen Mustererkenners untersuchen. Die Fähigkeit zur Ergänzung und Korrektur eines Musters mit leichten Fehlern zeigt eine gewisse Robustheit dieses Modells. Speichern wir jedoch das linke Bild aus Abbildung 2.1 und präsentieren anschließend nacheinander die drei anderen Bilder der Abbildung, so wird der Assoziativspeicher nicht in der Lage sein, das linke Bild zu assoziieren. Dies ist eine äußerst unerwünschte Eigenschaft des Modells, denn wir wollen die Objekte unabhängig von ihrer Position und Orientierung im Bild erkennen. In diesem Zusammenhang wird auch von Transformationsinvarianz gesprochen. Offensichtlich ist dieses Modell gegen keine der Transformationen aus Abbidlung 2.1 invariant.

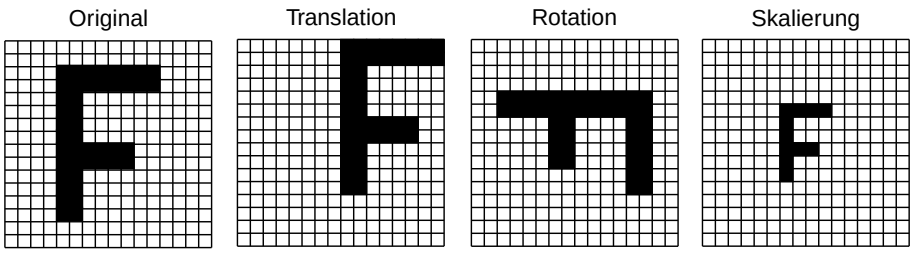

Abbildung 2.1: Darstellung verschiedener Transformationen.

2.1.2 Invarianz durch Vorverarbeitung

Eine Möglichkeit, die Invarianz gegen Translation, Rotation und Skalierung zu erlangen, ist die Verwendung der Fouriertransformation zur Vorverarbeitung der Bild-

daten. Durch die Fouriertransformation werden die reellen Merkmale der Bilder in komplexe Zahlen $k = a + ib$ überführt. Die komplexen Zahlen lassen sich auch in der Form $k = re^{i\varphi}$ darstellen, wobei $r = \sqrt{a^2 + b^2}$ der Betrag der Zahl k ist und $\varphi = \arctan(\frac{b}{a})$ der Drehwinkel gegen die positive x-Achse. Bei dieser Darstellung ist die Information über die Position des Objektes nur in der Phase φ enthalten. Durch Bildung des Betrages erhalten wir aus der Fouriertransformierten das Leistungsspektrum, das translationsinvariant ist. Eine Skalierung des Orginalbildes mit dem Faktor s resultiert in einer Skalierung der Fouriertransformierten um den Faktor $\frac{1}{s}$. Wird das Bild um den Winkel α gedreht, so äußert sich dies in einer Rotation der Fouriertransformierten um den Winkel $-\alpha$. Die Fouriertransformierte hängt von den beiden Ortsfrequenzen k_x und k_y ab. Wird die Fouriertransformierte bzw. ihr Betrag in Polarkoordinaten dargestellt, wobei die Achse für den Radius r logarithmisch sein soll, so wird aus einer Rotation (Änderung des Drehwinkels) ebenso wie aus einer Skalierung (Änderung des Radius) eine Verschiebung. Anschließend folgt eine weitere Fouriertransformation, alle Transformationen drücken sich nun nur noch durch Phasenunterschiede aus. Wird die Phase vernachlässigt, so erhalten wir die gewünschte Invarianz gegen Translation, Rotation und Skalierung [FH88].

Doch wie so oft hat auch dieses Verfahren eine Kehrseite. Die oben beschriebene Invarianz wird erkauft auf Kosten einer erhöhten Empfindlichkeit gegenüber Verzerrungen. Noch gravierender wirkt sich eine Veränderung des Hintergrundes aus, da sich diese Änderung nach der Fouriertransformation in jedem einzelnen Merkmal wiederfindet. Für genauere Informationen siehe Abschnitt 6.1.

Außerdem ist diese Transformation nicht umkehrbar. Es ist zwar möglich, ein Signal allein aus dem Betrag seiner Fouriertransformierten zurückzugewinnen, jedoch nur, wenn dies Signal eine endliche Ausdehnung hat, siehe [Hay82]. Diese Forderung ist spätestens für die zweite Fouriertransformation nicht mehr erfüllt.

2.1.3 Hopfield-Modell

Der Physiker J.J. Hopfield zeigte 1982 eine Verbindung zwischen magnetischen Phänomenen in Festkörpern und einem Netz aus rückgekoppeleten binären Neuronen auf. Dazu führte er eine Energiefunktion ein, die das Verhalten des Systems beschreibt, siehe [Hop82].

Ising-Modell

Das Ising-Modell dient der Beschreibung von magnetischen Eigenschaften eines Körpers. Dabei wird von einer Gitterstruktur ausgegangen, an deren Knotenpunkten sich Teilchen befinden, deren magnetische Momente nur nach oben oder unten zeigen können. Diese binäre Eigenschaft des Teilchens i wird als Spin s_i bezeichnet, wobei s_i nur die Werte -1 oder +1 annehmen kann. Die Stärke der Kopplung zwischen Teilchen i und Teilchen j wird durch w_{ij} repräsentiert und ist in einem Magneten notwendigerweise symmetrisch, es gilt $w_{ij} = w_{ji}$. Das Teilchen i besitzt ein lokales Magnetfeld h_i, das sich aus der Überlagerung eines extern angelegten Magnetfeldes h^e und eines durch die anderen Spins produzierten internen Feldes ergibt:

$$h_i = \sum_j w_{ij} s_j + h^e \ . \tag{2.9}$$

Das lokale Magnetfeld h_i beeinflußt den Spin s_i. Bei geringer Temperatur wird

der Spin sich parallel zu dem Magnetfeld ausrichten, das heißt er erhält das Vorzeichen des lokalen Feldes h_i.

Die Wechselwirkungen des Systems können auch durch eine Energiefunktion beschrieben werden. Eine zu Gleichung 2.9 passende Energiefunktion lautet

$$H = -\frac{1}{2}\sum_i \sum_j w_{ij} s_i s_j - h^e \sum_i s_i \ . \tag{2.10}$$

Symmetrisches Hopfield-Modell

In Anlehnung an das Ising-Modell bezeichnen wir die Neuronen des Hopfield-Netzes mit s_i, die nur die Zustände -1 oder $+1$ annehmen können. Die Verbindungen zwischen den einzelnen Neuronen sind symmetrisch angelegt, das heißt es gilt $w_{ij} = w_{ji}$. Der Zustand eines Neurons s_i zum Zeitpunkt $t+1$ berechnet sich nach

$$s_i(t+1) = \sigma(\sum_j w_{ij} s_j(t)) \ , \tag{2.11}$$

dabei steht σ für die in Abbildung 2.2 dargestellte Ausgabefunktion.

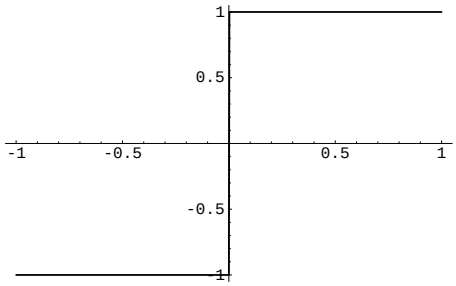

Abbildung 2.2: Ausgabefunktion σ für das Hopfield-Modell

Ein Neuron s_i ist stabil, wenn gilt $s_i(t) = s_i(t-1)$. Das gesamte Netzwerk befindet sich in einem stabilen Zustand, wenn alle Neuronen stabil sind. Ein stabiler Zustand wird auch als stationärer Zustand bezeichnet, da er in der Zeit unveränderlich ist. Das Hopfield-Modell stellt einen Autoassoziativspeicher dar, in dem die Muster durch stabile Zustände repräsentiert sind. Ein Modell wird als Autoassoziativspeicher bezeichnet, wenn die Schlüssel x^k und Inhalte y^k identisch sind ($x^k = y^k$). Bei einer Ausleseoperation wird dem Netzwerk zum Zeitpunkt t_0 ein Schlüssel präsentiert, indem dieser als Zustand $S(t_0)$ vorgegeben wird. Das System wird anschließend nach Gleichung 2.11 entwickelt. Erreicht das Modell einen stabilen Zustand, so repräsentiert dieser das assoziierte Muster. Durchläuft das System hingegen periodisch eine Zustandsfolge, so wird die Berechnung abgebrochen und das Muster gilt als nicht erkannt. Die Ausbildung solcher Grenzzyklen ist nur beim Hopfield-Netz mit synchronem Update möglich, das heißt in einem Zeitschritt werden alle Neurone gleichzeitig nach Gleichung 2.11 geändert. Im Gegensatz dazu gibt es beim asynchronen Update keine Grenzzyklen, hierbei wird zu einem Zeitpunkt genau ein Neuron aktualisiert. Die Reihenfolge, in der die Neuronen bearbeitet werden, kann fest oder zufällig sein.

Betrachten wir die Energiefunktion des Hopfield-Netzes

$$H = -\frac{1}{2}\sum_i \sum_j w_{ij} s_i s_j \ , \tag{2.12}$$

im Gegensatz zum Ising-Modell wird hier nicht explizit ein äußeres Feld berücksichtigt. Es ist möglich, die entsprechenden Schwellen als nullte Komponente in den Gewichtsvektor aufzunehmen, wie in Abschnitt 2.1.1 beschrieben. Die Energie des Systems kann man sich als eine Gebirgslandschaft vorstellen, wobei entlang der z-Achse die Energie aufgetragen ist und die 2^N Zustände, die ein Netzwerk mit N Neuronen besitzt, in der x-y-Ebene repräsentiert sind. Die Abbildung 2.3 zeigt ein Beispiel für eine solche Energielandschaft. Es sei noch angemerkt, das dies nur ein Bild zur Veranschaulichung ist. Bei einer echten Projektion der Zustände würde man ähnliche Zustände auf benachbarte Gitterpunkte abbilden, so daß ein ganz anderes Bild entstehen würde.

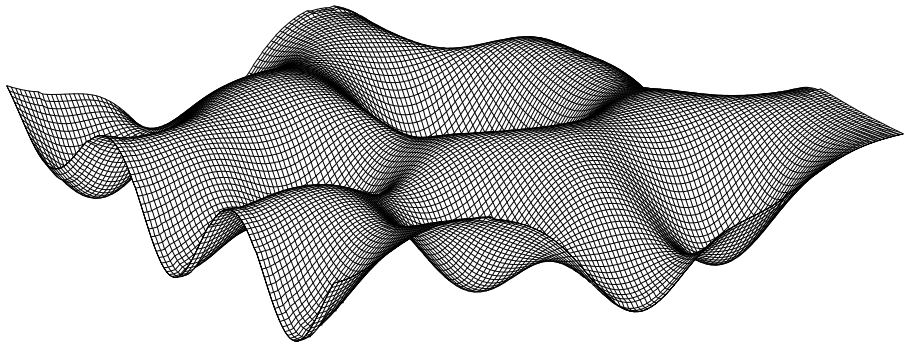

Abbildung 2.3: Beispiel für die Energielandschaft eines Hopfield-Netzes. Das Gitter wurde mit einem 128×128 Feld erzeugt und enthält somit 2^{14} Knotenpunkte, das heißt, diese Energielandschaft gehört zu einem Netzwerk mit 14 Neuronen.

Die Minima der Energiefunktion, also die Talsohlen der Gebirgslandschaft aus Abbildung 2.3, entsprechen den stabilen Zuständen und somit den gespeicherten Mustern. Anschaulich kann man sich die Arbeitsweise dieses Assoziativspeichers so vorstellen, daß eine Kugel in die Energielandschaft gelegt wird, dies entspricht der Vorgabe eines bestimmten Zustandes beim Auslesen. Die Kugel wird nun von der Gravitation nach unten gezogen. Dies Modell ist sehr stark vereinfacht, so daß physikalisch wichtige Elemente, wie zum Beispiel die Trägheit der Kugel, völlig außer acht gelassen werden. Die Kugel soll nur ein Indikator sein, der auf seinem Weg in die Talsohle die monotone Abnahme der Energie des Systems beschreibt, denn genau das gleiche geschieht, wenn das System nach Gleichung 2.11 entwickelt wird. Schließlich hat die Kugel die Talsohle erreicht, wo sie liegen bleibt und sich nicht mehr bewegt, sie befindet sich in einem stationären Zustand. Die Abbildung 2.4 zeigt den Assoziationsvorgang für ein sehr einfaches Netzwerk.

Für das Abspeichern der Muster x^μ wird die folgende Hebbsche Lernregel verwendet:

$$w_{ij} = \frac{1}{N}\sum_\mu x_i^\mu x_j^\mu \ . \tag{2.13}$$

Die Normierung $\frac{1}{N}$ spielt für die Dynamik keine Rolle und wird in der praktischen Implementation zumeist weggelassen, um Speicher zu sparen und mit Inte-

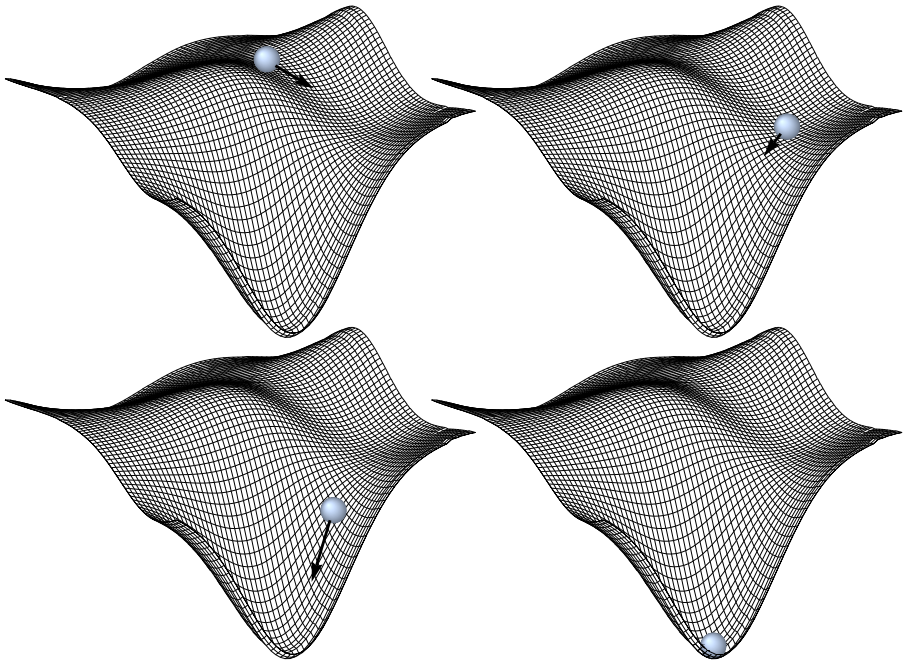

Abbildung 2.4: Die Assoziation eines gespeicherten Musters in einem Hopfield-Modell.

gerarithmetik rechnen zu können. Gibt es p Muster und N Neuronen, so wird das Verhältnis $\alpha = \frac{p}{N}$ als Speicherkapazität bezeichnet. Weiterhin bezeichnet $\alpha_c = \frac{p_c}{N}$ die kritische Grenze der Speicherkapazität, ab der die Leistung des Netzwerks zusammenbrechen wird. Um diese Grenze zu bestimmen, wird die maximale Anzahl der Muster ermittelt, die mit einer Wahrscheinlichkeit von $\mathcal{P}^\mu \geq 0.95$ korrekt assoziiert werden, wenn als Startzustand das Muster selbst vorgegeben wird. Die Speicherkapazität hängt stark von den gewählten Mustern ab. Für orthogonale Muster kann die Kapazität α sogar den Wert zwei erreichen (es werden N orthogonale Muster und ihre Komplemente gespeichert). Bei zufällig gewählten Mustern liegt die kritische Kapazität für große N bei $\alpha_c = 0.138$.

Für die Problematik der Invarianzen gelten genau die gleichen Aussagen, die über das Korrelationsmatrix-Modell gemacht wurden. Allerdings sind die Leistungen dieses Modells auf dem Gebiet der Fehlerkorrektur wesentlich besser, da die Fehler schrittweise über mehrere Iterationen behoben wird, wohingegen bei der Korrelationsmatrix nur ein einziger Assoziationsschritt vorgenommen wurde. Aufgrund der ungenügenden Invarianzeigenschaften muß auch dieses Modell als ungeeignet für die allgemeine Aufgabe des Matchings von Bildern angesehen werden.

Asymmetrisches Hopfield-Modell

Wird die Forderung nach einer symmetrischen Verbindungsstruktur fallengelassen, so ist es möglich, eine Projektion in ein Folgemuster zu erlernen, das heißt es wird eine Mustersequenz abgespeichert, wobei jedes Bild, das letzte ausgenommen, mit dem jeweiligen Folgebild assoziiert wird. Zunächst wird ein Muster in einem Hopfield-Netz gespeichert, anschließend wird ein zweites Muster präsentiert, mit dem eine Assoziation zum ersten gelernt wird. Dieses Verfahren kann auf weitere

Muster ausgedehnt werden, so daß letztendlich eine ganze Musterfolge gespeichert ist. Die stationären Zustände werden wieder mit Gleichung 2.13 gelernt. Für das Lernen aufeinanderfolgender Muster x^μ und $x^{\mu+1}$ wird die Lernregel

$$w_{ij} = \frac{1}{N} \sum_\mu x_i^{\mu+1} x_j^\mu \tag{2.14}$$

verwendet. Werden die symmetrischen Gewichte, die nach Gleichung 2.13 berechnet wurden, mit w_{ij}^s und die asymmetrischen Gewichte nach Formel 2.14 mit w_{ij}^a bezeichnet, so berechnen sich die Gesamtgewichte des Netzwerks nach

$$w_{ij} = \lambda w_{ij}^s + w_{ij}^a \ . \tag{2.15}$$

Beim Auslesen wird wiederum ein Muster präsentiert und die Zustände mittels Gleichung 2.11 entwickelt. Dabei wird das präsentierte Muster zunächst mit dem nächstgelegenen gespeicherten Muster assoziiert, anschließend wird von diesem Muster in das Folgemuster assoziiert. Die Assoziation des Folgemusters wiederholt sich, bis schließlich die stationären Zustände erreicht sind.

Bei geschickter Wahl der Muster ist es möglich, Transformationsinvarianzen für spezielle Muster zu erlernen, siehe [CK89]. So kann beispielsweise ein Muster gespeichert werden und dann eine leicht gedrehte Version dieses Musters mit dem ursprünglichen assoziiert werden. Zu diesem Muster lernt man eine Assoziation von einem Muster, das noch etwas stärker rotiert wurde usw. Die Abbildung 2.5 zeigt das Auslesen einer Mustersequenz.

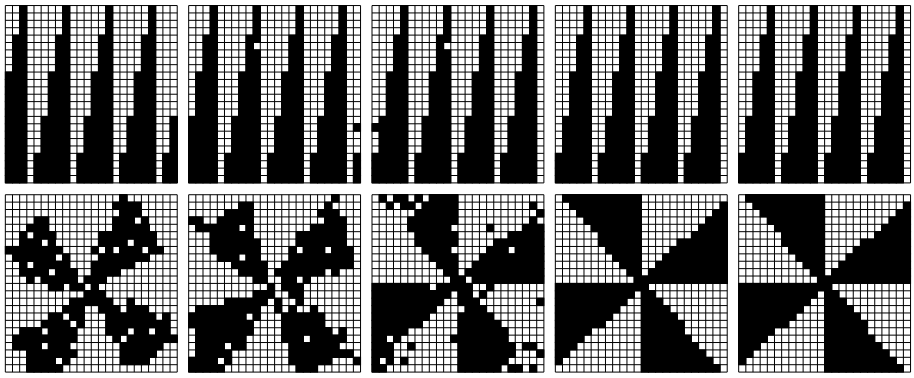

Abbildung 2.5: Die Assoziation einer Musterfolge in einem asymmetrischen Hopfield-Modell (nach [CK89]). Die obere Reihe zeigt eine Verschiebungsinvarianz, die untere ist invariant gegen Rotation.

Das System beherrscht immer nur Invarianz gegenüber einer Art der Transformation. Mehrere Invarianzen (beispielsweise Translation und Rotation zusammen) lassen sich nur unvollständig lernen.

Interessant ist die Tatsache, daß nach dem hier vorgestellten Modell die Erkennung eines rotierten Objekts umso länger dauern würde, je stärker die Rotation wäre. Dies läßt sich beim Menschen mittels psychophysikalischer Experimente bestätigen. Bei der Taube gibt es jedoch Experimente, die zeigen, daß es bei diesen Tieren keine derartige Verzögerung gibt.

2.2 Drei-Lagen-Perzeptron

Der Psychologe F. Rosenblatt stellte 1958 ein Modell vor, mit dem er den Vorgang der optischen Wahrnehmung (Perzeption) untersuchte [Ros58]. Um entscheiden zu können, ob ein Bild ein bestimmtes Objekt darstellt, wird bei Rosenblatt das Bild nicht Pixel für Pixel mit einem Muster verglichen, sondern es werden gewisse Eigenschaften zu dem Bild assoziiert und anschließend wird aufgrund der Eigenschaften des Bildes eine Entscheidung getroffen. Das Perzeptron besteht aus einer Eingabeschicht, die als Retina bezeichnet wird, einer Assoziationsschicht und einem Neuron als Ausgabeelement. Die Abbildung 2.6 zeigt den Aufbau des Drei-Lagen-Perzeptrons.

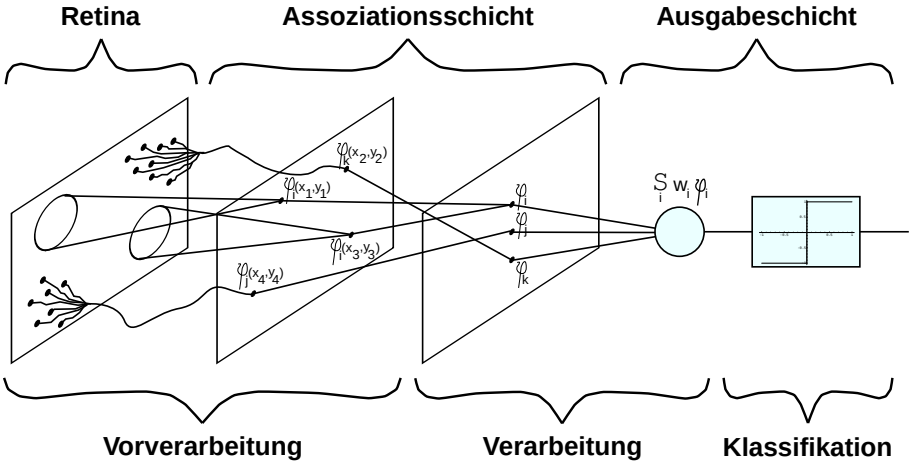

Abbildung 2.6: Der Aufbau eines Drei-Lagen-Perzeptrons.

Bei Rosenblatt werden drei Schritte in der Verarbeitung von Bildern unterschieden:

- **Vorverarbeitung des Bildes**
  Die Vorverarbeitung ist typisch für die Retina und unabhängig von den präsentierten Bildern. Durch diesen Schritt werden die Bilder in eine Menge von Merkmalen überführt, die in der Regel Prädikate darstellen.

- **Verarbeitung der Merkmale**
  Hier werden die Merkmale so bearbeitet, daß es möglich ist aufgrund dieser Ergebnisse eine Entscheidung zu treffen, ob das Bild in eine gegebene Musterklasse paßt. Die Verarbeitung hängt von der jeweiligen Aufgabenstellung ab, daher werden die Parameter explizit eingestellt oder durch Präsentation von Mustern erlernt.

  Bei der Verarbeitung wird folgende Aktivierungsfunktion benutzt:

$$a(x_1, x_2, \ldots, x_n) = \sum_j w_j x_j - \theta \ . \tag{2.16}$$

  Ist ein Merkmal typisch für die Musterklasse, so ist das zugehörige Gewicht positiv. Ein negatives Gewicht wird für ein untypisches Merkmal vergeben.

Wenn das Merkmal nicht signifikant für die Musterklasse ist, so erhält sein Gewicht den Wert Null.

- **Klassifikation**
  Abschließend wird anhand der Ergebnisse der ersten beiden Stufen entschieden, ob das präsentierte Bild zu der aufgabenspezifischen Musterklasse gehört.

  Dieser Schritt wird durch eine Stufenfunktion realisiert:

$$\sigma(a(\vec{x})) = \begin{cases} 1 & : \quad a(\vec{x}) \geq 0 \\ 0 & : \quad a(\vec{x}) < 0 \end{cases} \quad . \tag{2.17}$$

Um die Gewichte des Perzeptrons zu erlernen, können verschiedene Lernregeln eingesetzt werden, beispielsweise eine Hebbsche Lernregel oder eine Delta-Lernregel. Es gibt auch eine Perzeptron-Lernregel, die etwa in der Mitte zwischen Hebb- und Delta-Lernregel anzusiedeln ist

$$\Delta w_i = \gamma x_i \ . \tag{2.18}$$

Die folgenden Definitionen gehen auf Minsky und Papert zurück, die in ihrer Arbeit [MP88] die Fähigkeiten des Perzeptrons untersucht haben:

- **Durchmesserbegrenztes Perzeptron**
  Die Merkmale φ_i hängen nur von Punkten ab, die innerhalb eines Kreises mit Radius k liegen.

- **Perzeptron der Ordnung k**
  Ein Perzeptron ist von der Ordnung k, wenn alle Merkmale φ_i von maximal k Punkten abhängen.

- Kann mittels eines durchmesserbegrenzten Perzeptrons eine logische Aussage umgesetzt werden, so ist die Aussage *durchmesserbegrenzt.*

- Der *Träger* $S(\varphi)$ eines Merkmals φ ist die Menge der Bildpunkte, die in dieses Merkmal eingehen. Die Anzahl dieser Punkte wird durch $|S(\varphi)|$ ausgedrückt.

- Eine Aussage ist von der *Ordnung* k, wenn es ein Perzeptron von k-ter Ordnung gibt, das diese Aussage realisieren kann.

Die Vorverarbeitung des Perzeptrons extrahiert Merkmale für Punktmengen. Die Verarbeitung erfolgt allein auf der Grundlage der Existenz dieser Merkmale im Bild. Die gesamte Information über die Lokalisation der Merkmale bleibt somit unberücksichtigt. Der Abstraktion von einzelnen Pixeln zu Merkmalen für Pixelgruppen ist sicherlich ein wichtiger Schritt bei der Bildverarbeitung. Es erscheint jedoch befremdlich, daß dabei die Information über den Ort fallen gelassen wird. Schließlich ist es nicht egal, ob eine Nase am Kinn, auf der Stirn oder in der Mitte dazwischen plaziert ist. Im Prinzip können translationsinvariante Merkmale aus dem Leistungsspektrum gewonnen werden, jedoch ergeben sich damit die im Abschnitt 6.1 näher untersuchten Probleme.

Ein wirklich flexibler Objekterkennungsmechanismus muß in der Lage sein, die Relativpositionen von einzelnen Merkmalen zu berücksichtigen.

2.3 Selbstorganisierende Karten

Das Kohonen-Modell ist ein Beispiel für selbstorganisierende Karten, er wurde 1982 von T. Kohonen vorgestellt [Koh82], weitere Beschreibungen und seine Anwendung auf Beispielprobleme findet sich in [Koh84] und [RMS91]. Der Algorithmus fällt in die Kategorie des Wettbewerbslernen(*competitive learning*).

Das Kohonen-Modell besteht aus einer Neuronenschicht, wobei jedes einzelne Neuron einen Zeiger in den Musterraum repräsentiert. Ziel des Algorithmus ist es, eine Abbildung des Musterraums auf die Neuronenschicht zu lernen. Dies geschieht, indem die Zeiger so verändert werden, daß der Musterraum möglichst vollständig und gleichmäßig abgedeckt wird, wobei stets gefordert wird, daß benachbarte Punkte im Musterraum auf benachbarte Neurone abgebildet werden. Eine Abbildung, die diese Eigenschaften besitzt, wird als *nachbarschaftserhaltende* Abbildung bezeichnet, oft wird auch der Begriff *topologie-erhaltend* verwendet, der jedoch in der Mathematik eine etwas andere Bedeutung trägt und deshalb mit Vorsicht zu verwenden ist. Das Kohonen-Modell hat die Eigenschaft, daß alle leichten Variationen eines Musters auf das gleiche Neuron abgebildet werden, der Algorithmus ist demnach ein *Vektorquantisierer*. Die Abbildung 2.7 zeigt den generellen Aufbau des Kohonen-Modells.

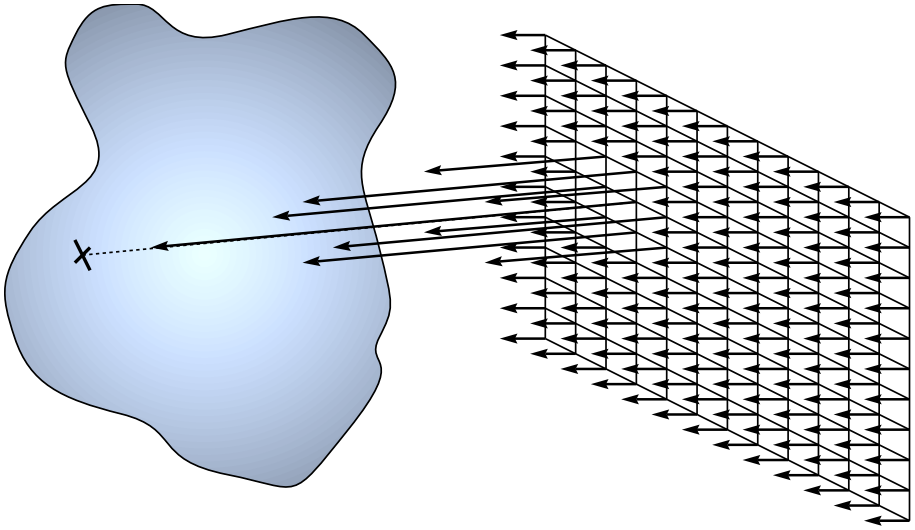

Abbildung 2.7: Das Kohonen-Modell.

In jedem Iterationsschritt wird ein zufälliger Vektor x^μ im Musterraum ausgewählt. Anschließend wird ermittelt, welches Neuron den Gewichtsvektor $\vec{w}_{i^\mu}$ besitzt, der diesem Punkt am nächsten kommt, es wird als Gewinner-Neuron (*winner-neuron*) bezeichnet. Im Unterschied zu anderen Algorithmen des Wettbewerbslernens, wo nur dieses Gewinner-Neuron an die aktuelle Eingabe angepaßt wird, darf hier eine kleine Gruppe aktiver Neurone ihre Gewichte in bezug auf die Eingabe ändern. Diese Gruppe wird angeführt von dem Gewinner-Neuron und beinhaltet weiterhin die Neuronen in seiner Nachbarschaft. Die Wahl dieser Nachbarschaft wird vom Anwender bestimmt und ist entscheidend für das Verhalten des Algorithmus. Die Nachbarschaft kann für den gesamten Algorithmus fest sein oder sich im Laufe der Zeit verkleinern, was sich günstig auf die Konvergenz des Verfahrens auswirkt. Ebenso steht es dem Anwender frei, ob die aktiven Neuronen ihre Gewichte alle mit der gleichen Stärke ändern, oder ob diese Stärke mit wachsendem Abstand

vom Gewinner-Neuron abnimmt. Die Lernregel für das Kohonen-Modell lautet

$$\Delta w_{ij} = \lambda h(i, i^\mu)(x_j^\mu - w_{ij}) \ . \tag{2.19}$$

Dabei steht $h(i, i^\mu)$ für eine Funktion, die den Wert für die Stärke liefert, mit der das Neuron i seine Gewichte verändern darf, wenn das Gewinner-Neuron den Index i^μ trägt.

2.3.1 Verwendung von Merkmalen

Um mit dem Kohonen-Algorithmus das Matching-Problem lösen zu können, muß er noch so erweitert werden, daß er die Neuronen in Abhängigkeit von der Ähnlichkeit ihrer Merkmale zu dem Merkmal des Mustervektors verändert. Aus diesem Grund wird die Lernregel 2.19 um einen Term $T(\vec{x}^\mu, \vec{w}_{i^\mu})$ erweitert, der die Ähnlichkeit zwischen dem Mustermerkmal und dem Merkmal des Gewinner-Neurons berechnet. Die Funktion T liefert normalerweise Werte aus dem Interval $[0, 1]$. Die neue Lernregel sieht nun folgendermaßen aus:

$$\Delta \vec{w}_i = \lambda h(i, i^\mu)(\vec{x}^\mu, \vec{w}_{i^\mu}) T(\vec{x}^\mu, \vec{w}_{i^\mu}) \ . \tag{2.20}$$

Ein wenig Vorsicht ist bei der Auswahl des Gewinner-Neurons geboten. Wird unter allen Neuronen gesucht, so wäre es denkbar, daß immer nur unähnliche Neuronen gewinnen und somit bei einem Ähnlichkeitswert von Null kein Fortschritt erzielt wird. Wird das Gewinner-Neuron jedoch nur unter ähnlichen Neuronen ermittelt, so werden die Gewichte stets verändert.

Nachteilig an diesem Verfahren ist die Tatsache, daß die Gruppe, die ihre Gewichte verändern darf, immer nur auf der Grundlage eines einzelnen Neurons ausgewählt wird. Es wäre sehr wünschenswert, wenn diese Entscheidung bereits auf die gesamte Gruppe bezogen würde. Dadurch ließe sich verhindern, daß bei einer großen Nachbarschaft viele unähnliche Neurone geändert würden.

2.4 Dynamic Link Matching

Das Dynamic Link Matching erzeugt eine Abbildung zwischen zwei Bildern, indem es jedem Element a der Bilddomäne X ein Element b der Modelldomäne zuordnet. Diese Zuordnung ist nicht im Sinne einer Eins-zu-eins-Korrespondenz zu verstehen, vielmehr wird zu Beginn jede Zelle der einen Schicht mit allen Einheiten der anderen Schicht verbunden. Die Verbindungen werden durch den Prozeß des Dynamic Link Matching verändert. Da die Verbindungen einer Dynamik unterliegen, spricht man auch von dynamischen Verbindungen oder englisch „Dynamic Link“, woher der Name des Verfahrens kommt, siehe [Beh93] für einen Vergleich zwischen Kohonen-Algorithmus und Dynamic Link Matching. Es gibt zwei verschiedene Formulierungen des Dynamic Link Matching, siehe [KV93] und [KMvdM94]:

- Ein neurobiologisch orientiertes Modell, daß die Aktivitäten von Neuronengruppen durch Differentialgleichungen simuliert.
- Eine Abstraktion des neuronalen Modells, das eine Energiefunktion verwendet, um die aufwendige Iteration der Differenzialgleichungen zu vermeiden.

Beide Versionen folgen dem gleichen Prinzip, und der eigentliche Unterschied ist in dem Rechenaufwand zu sehen, der bei dem zweiten Verfahren etwa um den Faktor 10 bis 15 geringer ist.

2.4.1 Neuronales Modell

Wie schon erwähnt, werden die Bild- und Modelldomäne unterschieden, im weiteren kurz als Schicht X und Schicht Y bezeichnet. Jede Zelle a der Schicht X ist mit jeder Zelle b der Schicht Y durch ein Link J_{ab} verbunden. Sind die beiden Schichten quadratisch und hat jede die Kantenlänge n, so gibt es n^4 Verbindungen. Der initiale Zustand dieser Verbindungen ist für den Algorithmus von keinerlei Bedeutung. Die Zelle einer Schicht kann durch ein Neuron repräsentiert werden, das einen bestimmten Aktivitätszustand x_a hat und mit dem lokalen Merkmal (*Feature*) f_a eines Eingabemusters, beispielsweise dem Grauwert eines Bildes, korrespondiert.

Dynamik der neuronalen Aktivität

Die Aktivität x_a eines Neurons a in der Schicht X läßt sich durch eine Differentialgleichung beschreiben:

$$\dot{x}_a = -\alpha x_a + \sum_{a'} g(a - a')\sigma(x_{a'}) + \varrho_a \ . \tag{2.21}$$

Die einzelnen Terme dieser Gleichung haben foldende Bedeutung:

$\dot{x}_a$ entspricht der zeitlichen Änderung der Aktivität des Neurons a der Schicht X.

$\sigma(x_a)$ steht für eine Ausgabefunktion, die der Begrenzung der Schichtaktivität dient. Abbildung 2.9 zeigt eine Auswahl möglicher Funktionen.

$-\alpha x_a$ ist ein sogenanntes Aktivitätsleck, daß die Aktivität der Zelle exponentiell gegen Null streben läßt, wenn die anderen beiden Terme keinen von Null verschiedenen Beitrag liefern.

$\sum_{a'} g(a - a')\sigma(x_{a'})$ entspricht der Faltung der Neuronenschicht mit einem Interaktionskern, der die Wechselwirkung zwischen den einzelnen Neuronen einer Schicht beschreibt. Für die Faltung gelten zyklische Randbedingungen. Abbildung 2.8 zeigt die typische Gestalt eines solchen Kerns.

ϱ_a repräsentiert eine Quelle, die Rauschen erzeugt und der Symmetriebrechung dient.

Durch die Differentialgleichung für Schicht X bildet sich eine Gruppe benachbarter Neurone aus, die alle eine hohe Aktivität haben, während die restlichen Neuronen in dieser Schicht eine verschwindend geringe Aktivität besitzen. Die Gruppe aktiver Neurone wird im folgenden *Blob* genannt werden. Im wesentlichen wird das Verhalten der Differentialgleichung durch den Interaktionskern bestimmt, der dafür sorgt, daß solche Neurone an Aktivität gewinnen, deren Nachbarn aktiv sind und für die es keine entfernten Zellen gibt, die eine starke Erregung besitzen. Da die Differentialgleichung einem stabilen Gleichgewichtszustand entgegenstrebt, verändert sich der Blob ab dem Erreichen diese Zustandes nicht mehr. Abbildung 2.10 zeigt einige Momentaufnahmen aus dem Blobgenerierungsprozeß.

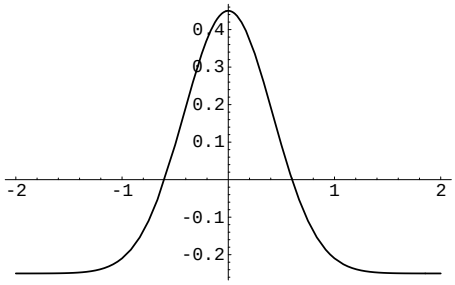

Abbildung 2.8: Faltungskern für die Wechselwirkung der Neuronen innerhalb einer Schicht

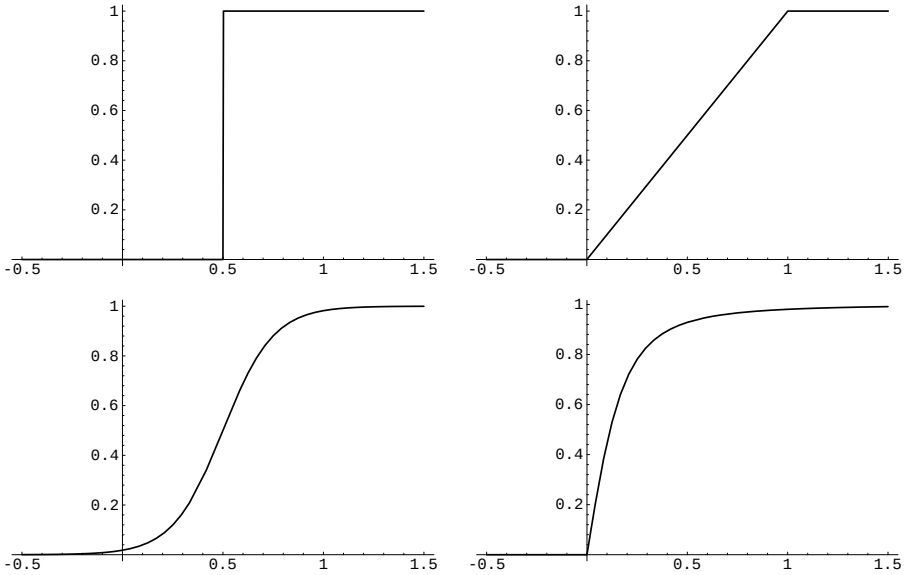

Abbildung 2.9: Eine Auswahl verschiedener Ausgabefunktionen

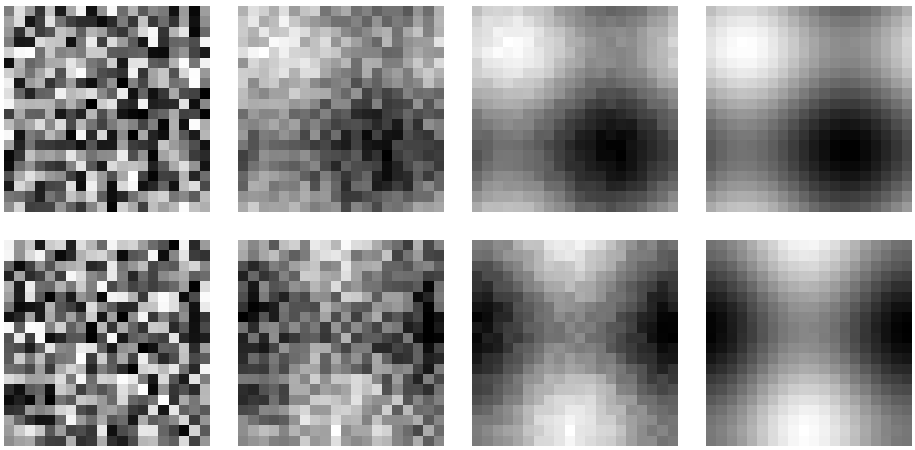

Abbildung 2.10: Jede Zeile zeigt Momentaufnahmen aus der Blobentstehung. Dunkle Zellen stehen für hohe Aktivität. Bei den Bildern ist zu beachten, daß zyklische Randbedingungen gelten, aus diesem Grund stellt auch das Bild rechts unten einen zulässigen Blob dar.

Für die Berechnung der neuronalen Aktivität der Schicht Y wird ebenfalls eine Differentialgleichung benutzt:

$$\dot{y}_b = -\alpha y_b + \sum_{b'} g(b-b')\sigma(y_{b'}) + \varepsilon \sum_a J_{ba} T_{ba} \sigma(x_a) \ . \tag{2.22}$$

Das Aktivitätsleck und der Interaktionskern haben die gleiche Funktion wie in der Differentialgleichung 2.21. Bei dieser Differentialgleichung wurde jedoch der Rauschterm durch einen Kopplungsterm $\varepsilon \sum_a J_{ba} T_{ba} \sigma(x_a)$ ersetzt. Dieser bewirkt, daß die Aktivität von Schicht X auf die Schicht Y übertragen wird. Die Stärke der Kopplung wird durch ε bestimmt. Die Aktivitätsübertragung ist proportional zum Verbindungsgewicht J_{ba} und der Ähnlichkeit der Merkmale an den Punkten a und b, diese wird durch T_{ba} repräsentiert.

In der Schicht Y läuft mittels der Differentialgleichung 2.22 prinzipiell der gleiche Prozeß ab, wie in der Schicht X. Die Aktivität konzentriert sich in einer Gruppe aktiver Neurone. Die Form und Position dieses Blobs wird entscheidend durch die Aktivitätsverteilung in der Schicht X mitbestimmt, da diese durch den Kopplungsterm Einfluß auf den Ablauf des Blobgenerierungsprozesses der Schicht Y nimmt. Aus diesem Grund werden die Blobs der Schicht Y auch als induzierte Blobs bezeichnet. Ebenso wie die Gleichung 2.21 strebt auch die Differentialgleichung 2.22 einem stabilen Gleichgewichtszustand entgegen.

Lernschritt

Sobald die Aktivität der beiden Schichten gegen ihren stabilen Gleichgewichtszustand konvergiert ist, werden die Verbindungsstärken der dynamischen Links verändert. Dabei wird folgende Lernregel verwendet:

$$\Delta J_{ba} = \lambda (J_{ba} + \tau) T_{ba} \sigma(y_b) \sigma(x_a) \ . \tag{2.23}$$

Eine Verbindung wird proportional zu den Aktivitäten der Zellen, die sie verbindet, sowie der Ähnlichkeit der Merkmale dieser Zellen, verändert. Die Größe τ hat die Bedeutung eines konstanten Wachstumsparameters und ermöglicht die Erhöhung von Verbindungsstärken, die bislang den Wert Null hatten. Um die lokalen Vieldeutigkeiten zu eliminieren, stehen die Links, die an einer Zelle $b \in Y$ ankommen, miteinander im Wettbewerb. Dies wird durch die sogenannte postsynaptische Normierung erreicht, bei der alle Links, die an einer Zelle der Schicht Y konvergieren auf den Wert eins normiert werden. Mathematisch stellt diese Normierung eine Randbedingung für die Differentialgleichung dar

$$\sum_a J_{ba} = 1 \qquad \forall b \in Y \ . \tag{2.24}$$

Nachdem die Verbindungen aktualisiert wurden, werden die Aktivitäten der beiden Schichten wieder auf Null gesetzt und ein neuer Berechnungszyklus gestartet, in dem neue Blobs generiert und die Links entsprechend geändert werden. Durch den Rauschterm in der Gleichung 2.21 wird dafür gesorgt, daß die Blobs an immer neuen Positionen entstehen.

Der große Vorteil dieses Modells liegt in der Verwendung von Blobs, die dafür sorgen, daß zu einem Zeitpunkt immer nur eine Untermenge der Zellen aktiv ist und so die zahlreichen Vieldeutigkeiten beim Matchen reduzieren. Außerdem wird durch

die Blobs eine topologische Randbedingung für die Abbildung eingebracht. Da immer nur benachbarte Neurone gleichzeitig aktiv sind, können solche Verbindungen gut wachsen, die benachbarte Neurone in der Schicht X auf benachbarte Neurone in der Schicht Y abbilden. Weiterhin wird die Entstehung neuer Blobs durch Verbindungen begünstigt, die durch die vorausgegangenen Blobpaare gestärkt wurden, diese Tatsache kann als eine Art Kurzzeitgedächtnis interpretiert werden.

2.4.2 Potentialbezogener Algorithmus

Das neurobiologische Modell hat den Nachteil, daß seine Simulation auf einem einzelnen sequentiellen Rechner sehr zeitaufwendig ist. Es wäre sehr viel besser auf einem massiv parallen System zu verarbeiten, was bei der Nähe des Modells zum biologischen Vorbild nicht zu verwundern mag. Da die simulierten Differentialgleichungen einen stabilen Gleichgewichtszustand besitzen, dem sie entgegenstreben, drängt sich die Vermutung auf, daß es möglich sein muß, das System so zu vereinfachen, daß es diese Lösung gut approximiert.

Energiefunktion

Es kann gezeigt werden, daß ein System von Differentialgleichungen der Form 2.21 bzw. 2.22 eine Energiefunktion besitzt, siehe [KMvdM94]. Eine stationäre Lösung dieser Gleichungen entspricht einem lokalen Minimum der entsprechenden Energiefunktion, wobei stationär für zeitlich unverändert steht. Für die Gleichungen 2.22 und 2.21 läßt sich folgende Energiefunktion angeben:

$$H = -\frac{1}{2}\sum_{b}\sum_{b'} g(b-b')\sigma(y_b)\sigma(y_{b'}) + \alpha\sum_{b}\int_{0}^{\sigma(y_b)} \sigma^{-1}(Z)\,dZ - \sum_{b}\varrho_b\sigma(y_b) \ . \quad (2.25)$$

Der Term ϱ_b in dieser Gleichung steht für den Rausch- bzw. Kopplungsterm aus der Differentialgleichung 2.21 bzw. 2.22. Um zeigen zu können, daß H nach unten beschränkt ist, muß nur noch das Integral genauer betrachtet werden. Wird beispielsweise eine sigmoide Ausgabefunktion der Form

$$\sigma(x) = \frac{1}{1+e^{-\lambda x}} \quad (2.26)$$

verwendet, so kann gezeigt werden, daß das Integral immer wohldefiniert ist. Für den Grenzfall $\lambda \to \infty$ verschwindet es sogar. Die Sigmoidfunktion approximiert für diesen Grenzfall die Stufenfunktion und kann umgekehrt beliebig flach gemacht werden. Wenn sich die neuronale Aktivität in den Schichten nach den beiden Differentialgleichungen 2.21 und 2.22 entwickelt, so nimmt die entsprechende Enegiefunktion H monoton ab und erreicht schließlich ein Minimum.

Wird ferner angenommen, daß die enstehenden Blobs immer die gleiche Form haben, was der Realität relativ nahe kommt, so sind die ersten beiden Terme der Gleichung 2.25 für alle Blobs gleich. Es reicht demnach aus, die Funktion

$$V = -\sum_{b}\varrho_b\sigma(y_b) \quad (2.27)$$

zu minimieren, um ein Minimum der Energiefunktion H zu finden. Diese Funktion wird im folgenden als Potentialfunktion bezeichnet werden.

Es ist ferner möglich, die Gleichgewichtslösung der Differentialgleichung, die den Blob repräsentiert, durch eine Fensterfunktion zu ersetzen, wodurch das Verfahren nochmals beschleunigt wird.

Ablauf des Algorithmus

Die schnelle Variante des Dynamic Link Matching läßt sich nun wie folgt beschreiben:

1) Initialisierung

   1.1) Auswahl einer eindeutig festgelegten Blobfunktion, beispielsweise die stationäre Lösung einer Differentialgleichung oder eine Fensterfunktion.

   1.2) Vorbelegung der Links mit der normierten Merkmalsähnlichkeit:

$$J_{ba} = \frac{T_{ba}}{\sum_{a'} T_{ba'}} \ . \tag{2.28}$$

2) Iteration

   2.1) Zufälliges Setzen des Blobs in der Schicht X und Berechnung des Inputs ϱ_b für die Schicht Y nach dem Kopplungsterm:

$$\varrho_b = \sum_a J_{ba} T_{ba} \sigma(x_a) \ . \tag{2.29}$$

   2.2) Berechnung des Minimums der Potentialfunktion V mit ϱ_b für $b \in Y$. Dies kann durch Ausprobieren aller möglichen Blobpositionen geschehen.

   2.3) Veränderung der Verbindungsstärken zwischen den aktiven Zellen und anschließende Normierung der Links.

In dieser Form erinnert der Algorithmus stark an den Kohonen-Algorithmus. Ein wesentlicher Unterschied zu diesem besteht darin, daß jede Zelle beim Kohonenalgorithmus nur einen einzigen Gewichtsvektor besitzt, der die Dimensionalität des Eingaberaums hat, während beim Dynamic Link Matching jede Zelle der Schicht X mit allen Zellen der Schicht Y verbunden ist und keine eindeutige Zuordnung durch den Algorithmus erzwungen wird. Dies ermöglich eine parallele Suche über viele mögliche Lösungen.

2.4.3 Matching mit dem Dynamic Link Algorithmus

In diesem Abschnitt soll das Dynamic Link Matching so beschrieben werden, wie es am Institut für Neuroinformatik zur Gesichtserkennung eingesetzt wird, siehe [KV93]. Diese Aufgabe stellt ein Anwendungsbeispiel des Dynamic Link Matching dar. Der Algorithmus kann prinzipiell auch für Zwecke der Spracherkennung eingesetzt werden und natürlich überall dort, wo nachbarschaftserhaltende Abbildungen einer Gruppe eventuell merkmalsbehafteter Elemente auf eine andere benötigt werden.

Vorverarbeitung

Für die Vorverarbeitung werden Wavelets eingesetzt, eine Einführung in die Wavelettransformation befindet sich im Anhang B. In jedem Bildpunkt wird die Wavelettransformation für mehrere Skalen und verschiedene Orientierungen berechnet. Die jeweiligen Merkmale werden zu einem Vektor zusammengefaßt, dieser Merkmalsvektor wird *Jet* genannt.

Die Wavelets können so gewählt werden, daß sie in ihrer Funktion den einfachen Zellen im primären visuellen Cortex ähneln. Diese Zellen sind besonders gut erregbar durch einen Lichtstreifen bestimmter Orientierung, der auf das rezeptive Feld der Zelle fällt. Auch die Wavelets haben eine Art rezeptives Feld. Die aus ihnen gewonnenen Merkmale haben besonders gute Information über die nähere Umgebung des Ortes, an dem sie lokalisiert sind, und relativ gutes Wissen über eine weitere Umgebung dieses Punktes. Für die Modellierung ist es ausreichend, wenn aus der Vielzahl der möglichen Orientierungen acht herausgegriffen werden und die Skalierungen auf fünf Werte beschränkt werden. Damit ergeben sich Jets mit jeweils 40 Komponenten.

Aus den 128×128 Jets wird eine Untermenge herausgegriffen, die fortan das Bild repräsentiert. Dies ist sinnvoll, da die Jets Information über die Umgebung besitzen und somit die gesamten Jets eine Menge mit großer Redundanz bilden. Andererseits ist die Auswahl einer Teilmenge auch notwendig, um das Dynamic Link Matching, so wie es hier vorgestellt wurde, mit verfügbaren Ressourcen bewältigen zu können. Bei 128×128 Bildpunkten gibt es $128^4 = 2^{28}$ mögliche Verbindungen, was bereits über den üblichen Größen für den Hauptspeicher von Workstations liegt und auch ein entsprechendes Rechenzeitproblem mit sich bringen würde. In bisherigen Simulationen wurden ungefähr 80 Jets verwendet, was einerseits ausreicht, um das Bild in angemessener Weise zu repräsentieren und andererseits, unter dem Einsatz paralleler Programmierung auf entsprechender Hardware, relativ schnell berechnet werden kann.

Die ausgewählten Jets werden mit ihrem nächsten Nachbarn verbunden, wobei Jets am Rand entsprechend weniger Nachbarn besitzen. Beim weiteren Vorgehen wird die absolute Position der Jets im Bild vernachlässigt und die Jets werden relativ zueinander, durch ihre Abstände, in Beziehung gesetzt. Dadurch wird die Translationsinvarianz erreicht.

Mittels der Vorverarbeitung wurde aus dem Bild, das durch eine Menge von Grauwerten repräsentiert war, ein Graph, dessen Knoten mit Jets und dessen Kanten mit Abstandsmaßen markiert sind.

Erkennung

Für die Erkennung wird eine Abbildung mit dem Dynamic Link Matching aufgebaut. Auf der Basis dieser Abbildung wird eine Gesamtähnlichkeit berechnet, beispielsweise als die Summe der Produkte von Merkmalsähnlichkeit und zugehöriger Verbindungsstärke. Liegt dieser Wert über einer festgelegten Schwelle und ist die Ähnlichkeit der anderen Kandidaten um einen ebenfalls zu wählenden Prozentsatz geringer, so gilt das Objekt als erkannt. Andernfalls zeigt das System an, daß das Objekt „unbekannt“ ist.

Kapitel 3

Vorverarbeitung der Daten

Die Kantenbilder werden mit der Mallatschen Wavelettransformation gewonnen, siehe [MZ91], [Lyr93]. Die mathematischen Grundlagen der Transformation werden im Anhang B erläutert. Hier sei noch einmal angemerkt, daß die dyadische Wavelettransformation nach Mallat dem Gradienten der Funktion $(f * \theta_{2^j})(x,y)$ auf der jeweiligen Auflösung entspricht, wobei θ_{2^j} durch Skalierung aus der Glättungsfunktion θ hervorgeht

$$\theta_{2^j}(x,y) = \frac{1}{2^j}\theta(\frac{x}{2^j},\frac{y}{2^j}) \tag{3.1}$$

$$\begin{pmatrix} (W^{\psi^1}_{2^j} f)(x,y) \\ (W^{\psi^2}_{2^j} f)(x,y) \end{pmatrix} = 2^j \vec{\nabla}(f * \theta_{2^j})(x,y) \ . \tag{3.2}$$

Die Länge (*modulus*) des Gradientenvektors ergibt sich nach der Gleichung

$$(M^{\psi}_{2^j} f)(x,y) = \sqrt{|(W^{\psi^1}_{2^j} f)(x,y)|^2 + |(W^{\psi^2}_{2^j} f)(x,y)|^2} \ . \tag{3.3}$$

Die Richtung des Gradienten, also sein Winkel gegenüber der Horizontalen, ist gegeben durch

$$(A^{\psi}_{2^j} f)(x,y) = \arctan \frac{(W^{\psi^2}_{2^j} f)(x,y)}{(W^{\psi^2}_{2^j} f)(x,y)} \ . \tag{3.4}$$

Die folgende Definition stammt von Mallat:

> Man nennt eine Stelle x_0 ein *Modulus Maximum* von $f(x)$, wenn die Funktion $|f(x)|$, auf der Geraden durch den Punkt x_0 in Richtung des Gradienten, ein lokales Maximum in einer Umgebung von x_0 ist und ein absolutes Maximum in einer einseitigen Umgebung von x_0.

Aus der diskreten Wavelettransformation werden für jede Skalierung 2^j die Modulus Maxima bestimmt, indem die Punkte ermittelt werden, für die $(M^{\psi}_{2^j} f)(x,y)$ größer oder gleich seinen beiden nächsten Nachbarn und zumindest echt größer als einer von beiden ist. Welche Punkte hierzu im mehrdimensionalen betrachtet werden, wird der Richtung des Gradienten entnommen. Für die Modulus Maxima

wird der Wert aus $(M^{\psi}_{2^j} f)(x, y)$ übernommen, die anderen Stellen werden auf Null gesetzt.

Die Abbildung 3.1 zeigt die verschiedenen Ergebnisse einer dyadischen Wavelettransformation nach Mallat für die Skalen $0 \leq j \leq 6$. Die beiden linken Spalten zeigen $W^{\psi^1}_{2^j} f$ und $W^{\psi^2}_{2^j} f$. In den folgenden beiden Spalten ist $M^{\psi}_{2^j} f$ und $A^{\psi}_{2^j} f$ zu sehen. Die ganz rechte Spalte zeigt die Modulus Maxima.

Die Information der Modulus Maxima ist im strengen mathematischen Sinne nicht vollständig. Es jedoch möglich, eine Approximation des Signal aus den Modulus Maxima zu gewinnen, die in mehren Iterationen verbessert werden kann. Diese Fehler dieser Approximation sind im Anwendungsgebiet der neuronalen Bildverarbeitung so gering, daß sie vernachlässigt werden können.

Die Rekonstruktionsmöglichkeit zeigt, daß die Modulus Maxima durchaus als Datenformat in Frage kommen. Weiterhin zeigen die Modulus Maxima bei Beleuchtungsänderungen geringere Variationen als die entsprechenden Bilder. Translationen, Rotationen und Skalierungen des Bildes ergeben eine entsprechende Transformation der Modulus Maxima. Dabei kann es zwar zu leichten Artefakten kommen, diese stellen aber für ein Matching-Verfahren, das auch mit Verzerrungen korrekt umgehen kann, kein Problem dar. Die Modulus Maxima bieten sich als Datenformat an, wenn Bilder mit Beleuchtungsvariationen und Texturunterschieden, wie sie beispielsweise bei Frisuren, Hintergründen und Kleidungsstücken auftreten, behandelt werden sollen.

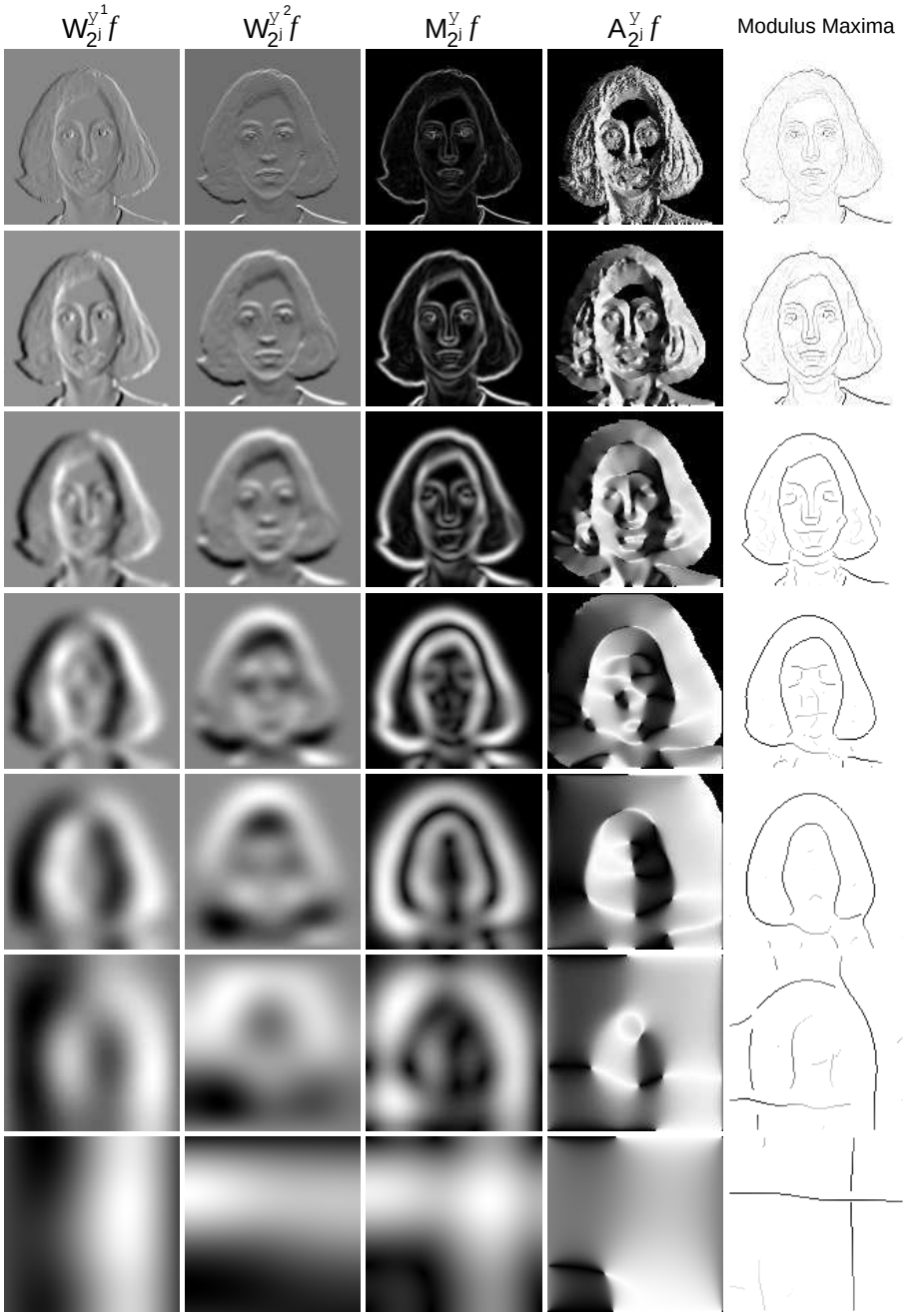

Abbildung 3.1: Die Bilder einer Spalte zeigen den Betrag der Gradienten in x- und y-Richtung ($W^{\psi^1}_{2^j} f$ bzw. $W^{\psi^2}_{2^j} f$), die Länge und Richtung der Gradienten ($M^{\psi}_{2^j} f$ bzw. $A^{\psi}_{2^j} f$) und die Modulus Maxima.

Kapitel 4

Das Matching-Verfahren

In diesem Kapitel wird der Matching-Algorithmus beschrieben. Für spezielle Fragen, die die Implementierung betreffen, sei auf das Kapitel 5 verwiesen. Da das Matching-Verfahren auf dem Dynamic Link Matching mit laufenden Blobs basiert, wird hier zunächst diese Variante des Dynamic Link Matching beschrieben, siehe [WvdM96]. Eine Einführung in das Dynamic Link Matching findet sich im Abschnitt 2.4.

4.1 Dynamic Link Matching mit laufenden Blobs

Die Dynamik der Differentialgleichungsvariante des Dynamic Link Matching hat den Nachteil, daß die Blobs bei jeder Iteration neu erzeugt werden müssen. Diesen Nachteil beseitigt der Potentialansatz zwar, aber dafür hat er immer noch mit relativ großen Blobs zu kämpfen, optimale Blobs überdecken etwa die Hälfte der Schicht. Dadurch gibt es innerhalb der Blobs wieder viele mögliche Zuordnungen, was die Qualität der gefundenen Abbildung nicht gerade positiv beeinflußt.

Hier soll nun ein Ansatz vorgestellt werden, der es erlaubt, die Blobs wiederzuverwenden und zudem die Vieldeutigkeit der Zuordnung vermindert. Außerdem wird das Verfahren symmetrisch, das heißt, es wird nicht mehr zwischen gesetzten Blobs und induzierten Blobs unterschieden. Die neuronale Aktivität der Zellen in Schicht X und Schicht Y wird nach der gleichen Differentialgleichung berechnet:

$$\dot{x}_a^n = -x_a^n + \gamma \left(\sum_{a'} g(a-a')\sigma(x_{a'}^n) - \beta \sum_{a'} \sigma(x_{a'}^n) \right) - \alpha s_a^n + \varepsilon \sum_{a'} J_{aa'}^{nm} \sigma(x_{a'}^m) + p + \varrho_a^n \ , \tag{4.1}$$

hierbei geben die tiefgestellten Indizes a, a' an, welche Zellen der Neuronenschicht betroffen sind. Die hochgestellten Indexe n, m stehen für die jeweilige Schicht. Die einzelnen Terme haben folgende Bedeutung:

$\dot{x}_a^n$ Zeitliche Ableitung der neuronalen Aktivität des Elements a der Schicht n.

$\sigma(x_a^n)$ Ausgabe des Neurons x_a^n. Die Funktion σ steht für eine Ausgabefunktion, wie sie beispielsweise Abbildung 2.9 zu finden ist.

$-x_a^n$ ist das Aktivitätsleck, das für sich alleine einen exponentiellen Abfall der Aktivität bewirkt.

$\gamma \sum_{a'} g(a-a')\sigma(x^n_{a'})$ beschreibt die lokale Kooperation der Neuronen einer Schicht. Der Faltungskern g ist durch die Formel:

$$g(x) = \frac{1}{2\pi\sigma^2} e^{\frac{-x^2}{2\sigma^2}} \tag{4.2}$$

bestimmt.

$-\beta \sum_{a'} \sigma(x^n_{a'})$ bewirkt den Wettbewerb innerhalb einer Schicht. Zusammen mit der lokalen Kooperation wird dies als Interaktionsterm bezeichnet, vergleiche Abbildung 2.8.

$-\alpha s^n_a$ steht für die Selbstinhibition. Sie entwickelt sich ebenfalls nach einer Differentialgleichung:

$$\dot{s}_a = \mu_\pm (x_a - s_a) \ , \tag{4.3}$$

mit

$$\mu_\pm = \begin{cases} \mu_- & : \quad x_a - s_a < 0 \\ 0 & : \quad x_a - s_a = 0 \\ \mu_+ & : \quad x_a - s_a > 0 \end{cases} \ .$$

$\varepsilon \sum_{a'} J^{nm}_{aa'} \sigma(x^m_{a'})$ entspricht einem Kopplungsterm mit der Kopplungsstärke ε.

p ist ein freier Parameter, der es ermöglicht das Verhalten der Differentialgleichung zu steuern.

ϱ^n_a ein Rauschterm zur Symmetriebrechung.

4.1.1 Aktivitätsdynamik der Neuronenschicht

Wird der Kopplungsterm weggelassen oder die Kopplungsstärke ε auf Null gesetzt und die Selbstinhibition s^n_a vernachlässigt, so bewirkt diese Gleichung das gleiche Verhalten der neuronalen Aktivität, wie die Differentialgleichung 2.21, es entstehen Blobs. Der Kompetitionsterm $-\beta \sum_{a'} \sigma(x^n_{a'})$ verhindert ein zu starkes Anwachsen der Neuronenerregung. Über den Parameter p kann zusätzlich das Verhalten der Differentialgleichung gesteuert werden, was für die praktische Simulation auf einem Computer sehr nützlich sein kann.

Betrachten wir nun die Selbstinhibition. Sie wächst für die aktiven Zellen der Schicht und fällt für die inaktiven Zellen. Nach einigen Iterationen wird die Selbstinhibition schließlich so groß, daß sie den Blob zwingt, seine Lokalisation zu wechseln. Da der Blob die Tendenz hat, benachbarte Neurone zu aktivieren, weicht er auf die nächstgelegenen Zellen mit geringerer Stärke der Selbstinhibition aus. Der Blob verweilt dort, bis auch an diesem Ort die Selbstinhibition zu stark wird, um abermals weiterzuziehen. In der Simulation ist dies ein kontinuierlicher Prozeß, ein stetiges Kommen und Gehen. Die Blobs wandern langsam über die ganze Schicht. Wenn sie die Schicht einmal komplett abgedeckt haben, hängt es von der Parameterwahl ab, ob sie kontinuierlich weiterlaufen oder verschwinden, um sofort an einem anderen Ort wieder neu zu entstehen. Abbildung 4.1 zeigt einige Ausschnitte aus der Aktivitätsentwicklung für die Dynamik der Wanderblobs.

In Abbildung 4.1 ist sehr schön zu sehen, wie die Selbstinhibition dem Blob folgt. Die zweite Sequenz zeigt deutlich den langen Schweif der Selbstinhibition,

dieser stellt eine Art Gedächtnis dar, weil er die Bahn festhält, auf der sich der Blob in einem zurückliegenden Zeitintervall bewegt hat. Durch dieses Gedächtnis wird der Blob sehr elegant an immer neue Positionen geleitet. In der unteren Sequenz ist auch zu sehen, wie der Blob sprunghaft die Lokalisation wechselt. Die Parameter für die Dynamik der beiden Sequenzen unterscheiden sich hauptsächlich in der Sigmoidfunktion. Für die obere Zeitreihe wurde eine Sigmoidfunktion verwendet, wie sie in Abbildung 2.9 rechts oben zu sehen ist, für die untere Sequenz wurde die Funktion rechts unten in der gleichen Abbildung verwendet.

4.1.2 Randbedingungen

Besondere Beachtung haben auch die Randbedingungen verdient. In den beiden vorausgegangen Versionen des Dynamic Link Matching hatten die Schichten die Form eines Torus, die Randbedingungen waren zyklisch. In dieser Variante ist nun eine Neuronenschicht ein zweidimensionales Feld von Neuronen und es gibt keine Verbindung zwischen Neuronen, die sich beispielsweise an dem rechten Rand befinden und solchen, die am linken Rand der Schicht lokalisiert sind. Diese Art von Randbedingungen ist neurobiologisch wesentlich plausibler als die Form eines Torus. Für die Blobs haben diese Randbedingungen zur Folge, daß die Randneuronen nur in Ausnahmefällen im Zentrum eines Blobs liegen, vielmehr werden sie von den seitlichen Ausläufern des Blobs überstrichen. Diese Tatsachen decken sich hervorragend mit unseren Bedürfnissen für die Bildverarbeitung. Beispielsweise ist bei dem Abbild eines Menschen der Fuß nicht benachbart mit dem Kopf, bloß weil beide Teile des Körpers auf dem unteren bzw. oberen Rand des Bildes lokalisiert sind. Ebenso können wir mit Recht erwarten, daß vernünftige Mechanismen die Aufmerksamkeit des Betrachters bzw. die Aufnahmerichtung einer Videokamera so gerichtet haben, das die interessanten bzw. informationstragenden Bereiche in der Mitte des entstandenen Abbildes angeordnet sind und nicht am Rand. Beim Auge ist eine derartige Ausrichtung besonders wichtig, da die Sehschärfe mit zunehmender Entfernung von der Fovea, dem Ort des schärfsten Sehens, sehr stark abnimmt. Die Abbildung 4.2 zeigt eindrucksvoll diesen Zusammenhang.

4.1.3 Lernregel

Die Lernregel wird bei der Wanderblobdynamik ebenfall durch eine Differentialgleichung beschrieben:

$$\dot{J}^{nm}_{aa'} = \lambda(J^{nm}_{aa'} + \tau)T^{nm}_{aa'}\frac{d}{dt}\sigma(x^n_a)\frac{d}{dt}\sigma(x^m_{a'}) \ . \tag{4.4}$$

Folglich ist die Änderung der Verbindungsstärke $\dot{J}^{nm}_{aa'}$ proportional zur Korrelation der zeitlichen Ableitung der sigmoiden neuronalen Aktivität zwischen den Zellen a und a', sowie der Ähnlichkeit der Merkmale, mit denen diese beiden Zellen behaftet sind. Der Parameter λ drückt eine Lernkonstante aus und τ hat einen kleinen positiven Wert, um das Wachstum von Verbindungen zu ermöglichen, die bislang den Wert Null hatten.

Im Unterschied zu den beiden vorausgegangenen Versionen des Dynamic Link Matching wird hier im Anschluß an einen Lernschritt keine Normierung vorgenommen, sondern lediglich die negativen Verbindungsstärken auf Null gesetzt. Die Links,

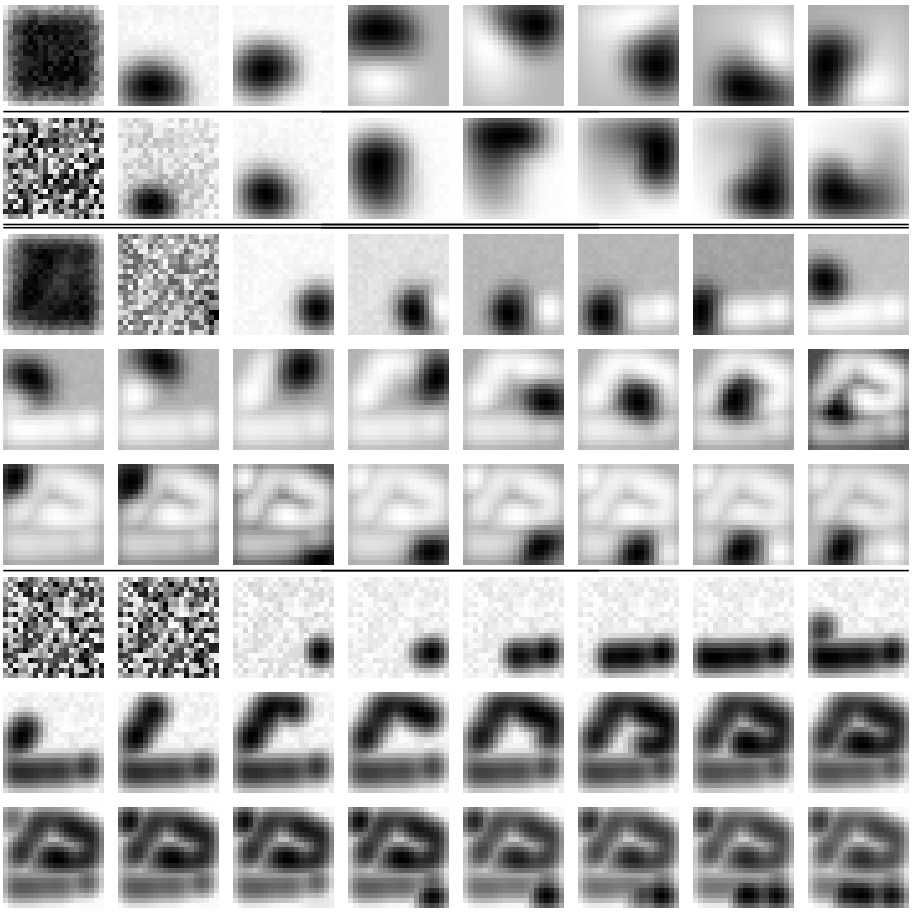

Abbildung 4.1: Jedes Teilbild (über und unter dem Doppelstrich) zeigt eine Aktivitätssequenz der Wanderblob-Dynamik für zwei verschiedene Parametersätze:
1. Teilbild: $\gamma = 1$, $\sigma = 2$, $\alpha = 1$, $\beta = 0.03$, $\varepsilon = 0$, $p = 0$, $\varrho = 0.01$, $\mu_+ = 0.2$, $\mu_- = 0.05$, Ausgabefunktion $\sigma(x) = \begin{cases} 0 & : \quad x \leq 0 \\ x & : \quad 0 < x < 1 \\ 1 & : \quad x \geq 0 \end{cases}$
2. Teilbild: $\gamma = 1$, $\sigma = 3$, $\alpha = 1$, $\beta = 0.03$, $\varepsilon = 0$, $p = 0$, $\varrho = 0.01$, $\mu_+ = 0.2$, $\mu_- = 0.005$, Ausgabefunktion $\sigma(x) = \begin{cases} 0 & : \quad x \leq 0 \\ \sqrt{x} & : \quad 0 < x < 1 \\ 1 & : \quad x \geq 0 \end{cases}$
Bei jeder Sequenz ist oberhalb der einfachen Linie die Aktivität der Neuronenschicht und unterhalb die Selbstinhibition dargestellt. Zwischen den aufeinanderfolgenden Bildern einer Sequenz liegen jeweils zehn Iterationen der Differentialgleichung 4.1.

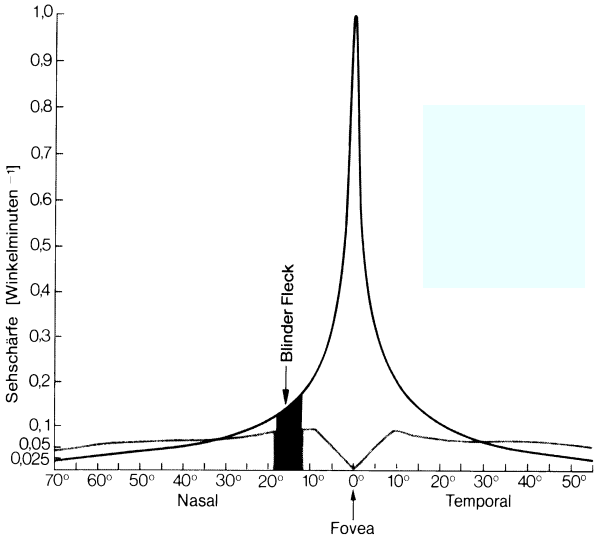

Abbildung 4.2: Abhängigkeit der Sehschärfe (Y-Achse) vom Ort im Gesichtsfeld (X-Achse), aus [ST80]. Die Kurve, die in der Fovea ihr Maximum hat, zeigt die Sehschärfe für das Farbsehen (Zapfen), während die Kurve mit dem Minimum am Ort der Fovea die Sehschärfe für das Sehen bei sehr wenig Licht (Stäbchen) zeigt.

die an einer Zelle ankommen, werden erst in der Summe auf eins normiert, wenn der Input $\sum_{a'} J^{nm}_{aa'} \sigma(x^m_{a'})$ für diese Zelle einen Wert liefert, der größer als eins ist.

4.1.4 Korrelation der Ausgabeänderung

Wie bereits erwähnt, steigt mit der Blobgröße auch die Anzahl der mehrdeutigen Zuordnungen. Aus diesem Grund sind möglichst kleine Bereiche, auf den sich die gesamte Schichtaktivität konzentriert, wünschenswert. Die Wanderblobs sind bereits klein im Vergleich zu ihren stationären Kollegen. Durch Verwendung der Korrelation zwischen den Ausgabeänderungen anstatt den neuronalen Aktivitäten selbst kann der Algorithmus nochmals an Genauigkeit gewinnen. Durch die Bewegung des Blobs ist es möglich, relativ zur Bewegung zwei Teile zu unterscheiden. Der Blob besitzt nun einen vorderen Teil und einen hinteren. Mittels der Korrelation bezüglich der Veränderung der Neuronenausgabe sind nur Abbildungen erlaubt, die zwischen den gleichen Teilen stattfinden. Die Abbildung 4.3 veranschaulicht diesen Zusammenhang.

In Abbildung 4.3 ist zu sehen, daß die zeitliche Korrelation auch negative Werte annehmen kann. Dadurch ist es dem Algorithmus möglich, falsch verbundene Links aktiv zu verlernen. Bei den ersten beiden Varianten des Dynamic Link Matching können Verbindungen nur passiv mittels Wettbewerb verlernt werden, also nur wenn neue Links wachsen, werden die falschen langsam abgebaut.

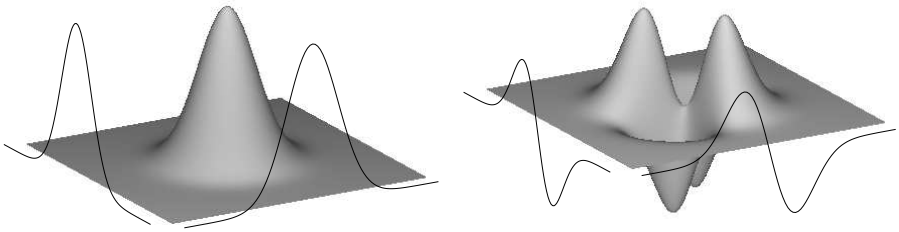

Abbildung 4.3: Zusammenhang zwischen Korrelation der Ausgabeänderung und Zuordnungsgenauigkeit. Im Bild links ist die direkte Korrelation zwischen zwei Blobs dargestellt, rechts befindet sich die Graphik für die Korrelation bezüglich der Veränderung der Neuronenausgabe. Die Funktionen im Vordergrund zeigen einen Schnitt durch die Aktivitätsverteilung bzw. deren zeitliche Ableitung, wobei dieser Schnitt durch einen Blob geht.

4.1.5 Ablauf des Dynamic Link Matching mit wandernden Blobs

Der Algorithmus ist zunächst sehr einfach:

1) Initialisierung

   1.1) Die Links der Verbindungsmatrix können auf Null gesetzt oder mit den Ähnlichkeitswerten der von ihnen verbunden Zellen belegt werden.

2) Iteration

   2.1) Die Wanderblob-Dynamik wird nach Differentialgleichung 4.1 simuliert und eventuelle Normierungen der betroffenen Links in der Verbindungsmatrix werden vorgenommen.

   2.2) Anwendung der Lernregel 4.4.

Eine Verbesserung für das Laufzeitverhalten läßt sich noch leicht einbringen. Da durch die Dynamik der Wanderblobs sichergestellt ist, daß der Blob in aufeinanderfolgenden Iterationen immer an neuen Positionen erscheint, ist es möglich, die Korrelation der Ausgabeänderung über ein bestimmtes Zeitintervall zu sammeln und anschließend einen Lernschritt mit diesem Konzentrat zu speisen. Die Korrelation ist außerhalb der Blobs nahe bei Null. Solange beide Blobs immer neue Positionen annehmen, kann die Korrelation bezüglich der Veränderung der Neuronenausgabe ohne Informationsverlust aufsummiert werden. Die optimale Anzahl der Iterationen, über die diese Summation erfolgt liegt knapp unter der Anzahl, die ein Blob benötigt, bis er wieder an den gleichen Positionen lokalisert ist. Die Iterationsanzahl für einen derartigen Blobzyklus ist von der Schichtgröße unabhängig, wird aber stark bestimmt durch die Parameter α und $\mu_{\pm}$. Aus diesem Grund sollte die Anzahl der Iterationen, über die die Korrelation der Ausgabeänderung summiert wird, durch einen neuen Parameter z bestimmt werden. Dadurch wird der Algorithmus wie folgt formuliert:

1) Initialisierung

   1.1) Die Links der Verbindungsmatrix können auf Null gesetzt oder mit den Ähnlichkeitswerten der von ihnen verbunden Zellen belegt werden.

2) Iteration

2.1) z Iterationen

2.1.1) Die Wanderblob-Dynamik wird nach Differentialgleichung 4.1 simuliert und eventuelle Normierungen der betroffenen Links in der Verbindungsmatrix werden vorgenommen.

2.1.2) Berechnung der Korrelation der Ausgabeänderung und Addition dieser zu einer Summe $C_{aa'}^{nm}$ für alle Zellen a in Schicht n und alle Zellen a' in Schicht m.

2.2) Anwendung der modifizierten Lernregel:

$$\dot{J}_{aa'}^{nm} = \lambda(J_{aa'}^{nm} + \tau)T_{aa'}^{nm}C_{aa'}^{nm} \ . \tag{4.5}$$

2.3) Zurücksetzen der Summe $C_{aa'}^{nm}$ auf den Wert Null.

4.2 Matching von Kantenbildern

Wollte man das bisherige Verfahren für das Matching von Grauwertbildern mittels des Dynamic Link Matching auf Kantenbilder übertragen, so müßte aus der Menge der Kanten eine Teilmenge herausgegriffen werden. Diese Menge müßte genügend Information über das Bild enthalten, damit beurteilt werden kann, ob ein abgebildetes Objekt mit einem anderen übereinstimmt. Anderseits muß die Menge so handlich sein, daß sie bei Anwendung des Dynamic Link Matching keine Komplexitätsprobleme hervorruft. Diese Methode bereitet einige Probleme. So war es bei Grauwertbildern unbedeutend, ob sich ein Jet genau auf der Nasenspitze befand oder leicht daneben. Bei den Modulus Maxima macht ein Pixel aber bereits den Unterschied zwischen Kante oder Nicht-Kante aus, dies bedeutet in Werten Betrag des Gradienten oder Null. Durch diese dramatischen Verhältnisse ist man gezwungen die Knoten explizit auf die Kanten zu setzen, was die Freiheit des Systems bereits stark einschränkt. Nehmen wir trotz der Probleme an, daß der Graph korrekt plaziert ist, so ergibt sich das Problem einer geeigneten Kodierung. Für den Punkt, auf dem der Knoten plaziert ist könnte die Richtung und der Betrag des Gradienten gespeichert werden. Damit fehlt aber noch das Umgebungswissen. Der Betrag und die Richtung des Gradienten kann in einem einzelnen Punkt zu ganz verschiedenen Merkmalen gehören, siehe beispielsweise Abbildung 4.4. Weiterhin muß das Merkmal Information über andere Kanten, die in nächster Nähe verlaufen, enthalten.

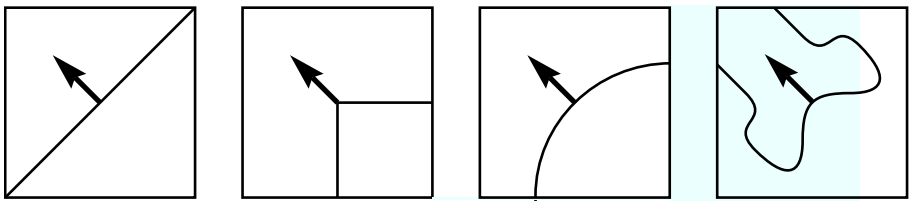

Abbildung 4.4: Beispiele für verschiedene Kantenbilder, zu denen der Gradient in der Mitte gehören kann.

Sollen Mehrskalen-Kanten verwendet werden, da sie eine genauere Wiedergabe des Bildes ermöglichen, so kommt noch das Problem hinzu, daß sich die korrespon-

dierenden Kanten auf den verschiedenen Skalen an unterschiedlichen Positionen befinden können, vergleiche Abbildung 3.1.

Die geschilderten Probleme verdeutlichen, warum die Bildung eines Graphen aus einer Teilmenge der Kanten kein vielversprechender Ansatz ist. Es müßten bei der Generierung des Graphen zuviel Entscheidungen bezüglich dessen Struktur getroffen werden, ohne daß Wissen über die im Bild repräsentierten Objekte vorhanden wäre.

Aus den genannten Gründen wird in dieser Arbeit nicht nur eine Teilmenge der Kanten für das Matching verwendet, sondern dem Algorithmus stehen alle im Bild vorhandenen Kanten zur Verfügung.

Da Bilder mit einer Auflösung von 128×128 Punkten eine Verbindungsmatrix mit $128^4 = 2^{28}$ Links besitzen, die bei Verwendung einer einfachen Gleitkommadarstellung jeweils vier Byte benötigen, liegt der gesamte Speicherbedarf für eine derartige Verbindungsmatrix bei einem Gigabyte. Wird das Dynamic Link Matching mit laufenden Blobs verwendet, so werden zwei Verbindungsmatrizes, eine Korrelationsmatrix und eine Ähnlichkeitsmatrix gebraucht, die somit zusammen vier Gigabyte Speicher belegen. Eine Möglichkeit bestände darin dem Mathematiker und Komplexitätstheoretiker A. M. Turing und seinem Modell der sogenannten Turing-Maschine zu huldigen und die anfallenden Daten auf ein Band zu schreiben. Ernsthafte Überlegungen zu diesem Problem müssen zu einer Verringerung der Anzahl der Matrixelemente, für die weitere Berechnungen durchgeführt werden, führen. Es ist zwar möglich die Ähnlichkeiten der Merkmale bei jeder Verwendung neu zu berechnen und so die Ähnlichkeitsmatrix einzusparen. Ebenfalls ist es denkbar, eine Verbindungsmatrix zu opfern, da sich die Verbindungen für die beiden Richtungen zwar prinzipiell individuell entwickeln können, sich aber in der Regel nur geringfügig unterscheiden. Dadurch wäre der Speicherbedarf zwar halbiert (andere benötigte Variablen unberücksichtigt), aber auch zwei Gigabyte sind entschieden zu viel.

4.2.1 Reduktion des Speicherbedarfs

Wie die obigen Ausführungen zeigen, verbietet es sich von selbst direkt 128×128 Punkte auf die gleiche Anzahl matchen zu wollen. Auch ist keinerlei Vorwissen vorhanden, das es ermöglicht, aus den 2^{28} Links die benötigten herauszugreifen. Es muß eine Möglichkeit gefunden werden, das Problem langsam zu approximieren.

Superpixel

Für eine Approximation des gesamten Matchingproblem ist es sinnvoll, zunächst Approximationen des Bildes zu betrachten. Indem die Auflösung vergröbert wird, kann man das Bild unterabtasten(*subsample*). Es können beispielsweise je 2×2 Pixel zu einem neuen Pixel zusammengefaßt werden, dessen Wert dem Mittelwert der vier Pixel entspricht. Jedes Pixel, das aus der Zusammenfassung mehrer Pixel entstanden ist, wird fortan als *Superpixel* bezeichnet.

Werden je 16×16 Pixel zu einem Superpixel zusammengefaßt, so kann ein Bild mit 128×128 Pixeln durch 8×8 Superpixel dargestellt werden. Verfahren wir so mit beiden Bildern und ordnen jedem Superpixel eine Zelle zu, so können wir die Links dieser Zellen als *Superlinks* auffassen. Ein Matching von 8×8 auf 8×8 Neuronen ist für das Dynamic Link Matching kein Problem, auch die Blobdynamik läßt sich bei dieser Schichtgröße schnell simulieren. Nach einigen Iterationen erhalten wir eine Abbildung zwischen den beiden stark vergröberten Bildern, die als eine erste Näherung der gewünschten Abbildung angesehen werden kann. Es stellt sich nun

das Problem von dieser ersten Näherung auf die Struktur der vollen Abbildung zu schließen.

Verfeinerung

Aufgrund des Wettbewerbs und der Nachbarschaftserhaltung sind die gewonnenen Abbildungen stark strukturiert. In der Regel gibt es für jedes Pixel ein Link mit relativ großer Verbindungsstärke, während die restlichen Links, die dieses Pixel besitzt, sehr schwach sind. In einer Reihe von Simulatioen wurde die Annahme bestätigt, daß es möglich ist, auf einen großen Teil dieser schwachen Verbindungen zu verzichten und sie bei den weiteren Berechnungen nicht zu berücksichtigen. Auf die Dynamik hat dies keinen wesentlichen Einfluß, da sich ihr Fehlen nur in einem sehr kleinen Unterschied des Inputs für jede Zelle bemerkbar macht. Es ist demnach möglich, aus der ersten Näherung der Abbildung einen Großteil der Verbindungen zu entfernen, ohne die Qualität der Approximation bemerkenswert zu verschlechtern.

Wenn es möglich ist, genügend Links zu vernachlässigen, so kann die volle Abbildung langsam approximiert werden, indem aus einer groben Näherung die schwachen Links entfernt werden und die verbleibenden starken Links in geeigneter Weise verfeinert werden.

Bisher haben wir einzelne Pixel zu Superpixeln zusammengefaßt, diesen eine Zelle der Schicht zugeordnet und deren Links als Superlink interpretiert. Die Verfeinerung muß nun die Superlinks wieder in kleinere Einheiten aufgeteilen. Dazu können wir die beiden Superpixel, die das Superlink verbindet in je 2×2 Pixel(eventuell sind dies wieder Superpixel) aufteilen und das Superlink durch 16 neue Links ersetzten, die jedes der 2×2 Pixel des einen Superpixel mit jedem Pixel des anderen verbinden und die gleiche Verbindungsstärke besitzen wie das Superlink. Die Abbildung 4.5 veranschaulicht dieses Verfahren zur Verfeinerung.

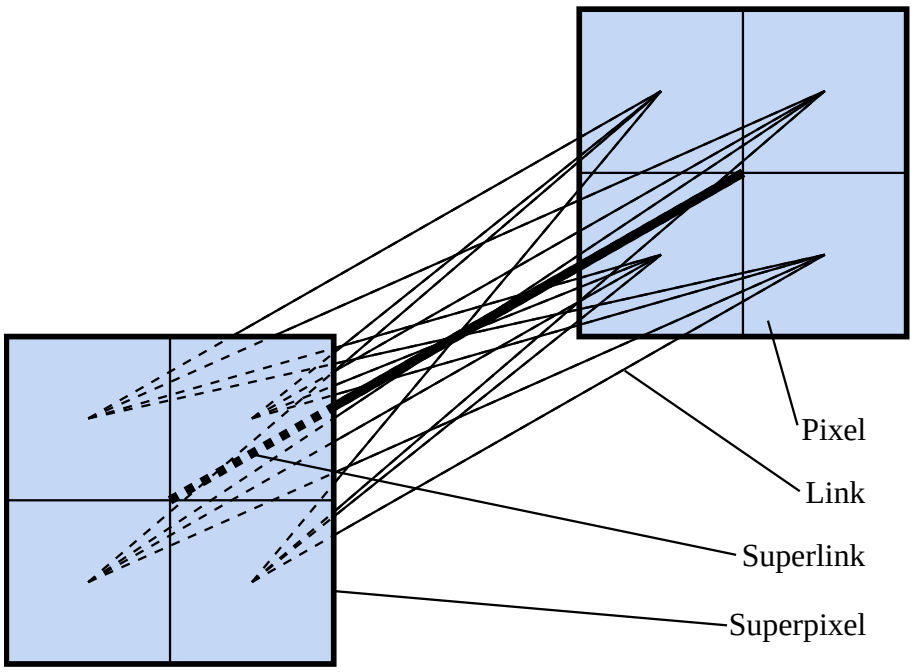

Abbildung 4.5: Zerlegung eines Superlinks

Um eventuelle Probleme mit sehr hohen Inputwerten für einzelne Zellen, im direkten Anschluß an die Verfeinerung zu vermeiden, wird vor der Verfeinerung eine Normierung der Links erzwungen. Dabei werden wieder nur die Links auf den Wert eins normiert, die auf eine Zelle konvergieren und gemeinsam eine Verbindungsstärke größer eins ergeben.

Selektion

Vor jeder Verfeinerung der Verbindungsmatrix müssen die Links ausgewählt werden, die anschließend zerlegt werden sollen. Für die Auswahl gibt es grundsätzlich zwei konträre Ansätze:

- Die Selektion wird unter Berücksichtigung topologischer Gesichtspunkte vorgenommen. So können beispielsweise immer nur Links ausgewählt werden, die in der vierdimensionalen Verbindungsmatrix benachbart sind.

- Es werden nur die stärksten Links ausgewählt, ungeachtet der Punkte, die sie verbinden.

Bei der Auswahl nach topologischen Kriterien besteht stets die Gefahr, daß Vorwissen über gewünschte Abbildungen eingebracht wird, das dem Matchingprozeß ein bestimmtes Verhalten aufzwingt. So ist es möglich, bestimmte Abbildungen völlig zu unterdrücken und dadurch die Allgemeingültigkeit des Verfahrens zu zerstören. Bei dem Versuch, eine Abbildung zwischen zwei völlig unterschiedlichen Bildern zu finden, würde das Dynamic Link Matching eine schwächer strukturierte Näherung für die Abbildung liefern, eine Auswahl nach topologischen Gesichtspunkten würde künstlich diese Struktur einbringen, indem es beispielsweise die Standardabbildungen favorisiert. Dies kann dazu führen, daß der Algorithmus dem Benutzer eine weitaus höhere Güte der Abbildung vorgaukelt, als tatsächlich gegeben ist.

Die Selektion der Links unter alleiniger Berücksichtigung der Verbindungsstärke würde die Abbildung nicht künstlich strukturieren, sondern nur etwas schwächer ausgeprägte Strukturen vernichten. Dies gibt zu der Hoffnung Anlaß, daß der Matchingprozeß zwei ähnliche Bilder in annähernd gleicher Weise auf ein drittes abbilden würde. So könnten leicht unterschiedliche Abbildungen eines Objekts, durch das Matching auf einen Prototypen für dies Objekt, als verschiedene Ansichten des gleichen Objekts erkannt werden. Bei dieser Methode kann es theoretisch passieren, daß die Verbindungsmatrix „entartet“, das heißt, sie verstößt grob gegen die Nachbarschaftserhaltung. Eine derartige Abbildung kann erwünscht sein, wenn zwei unähnliche Bilder gematcht werden. So ist es möglich, aufgrund der schlechten Qualität der Abbildung die Bilder als unähnlich zu klassifizieren.

Weiterhin muß entschieden werden, wieviele Links bei der Verfeinerung bearbeitet werden sollen. Hierzu gibt es die folgenden Möglichkeiten:

- Eine genau festgelegte Anzahl von Verbindungen wird ausgewählt.

- Es werden solange Links ausgewählt bzw. eliminiert, bis eine gegebene Gewinnfunktion maximiert bzw. eine Kostenfunktion minimiert ist. Eine einfache Kostenfunktion könnte die Elimination einer möglichst großen Anzahl von Links fordern, wobei die Summe ihrer Verbindungsstärken unter einer gegebenen Schranke bleiben muß.

Beide Verfahren erfordern genaue Voruntersuchungen darüber, wieviele Verbindungen selektiert werden können, bis die vorhanden Ressourcen, insbesondere Speicherplatz, ausgeschöpft sind. Die Vorteile des ersten Verfahrens sind darin zu sehen, daß es diese Beschränkungen exakt einhält, wohingegen die zweite Methode zwar mehr Flexibilität bietet, dafür aber auch die Gefahr birgt, den gegebenen Rahmen zu sprengen.

Ist die Abbildung auf einer Auflösungsstufe eine optimale Approximation für die gesuchte Abbildung, so kann leicht eine untere Schranke für die Anzahl der Links, die mindestens ausgewählt werden müssen, gegeben werden. Nehmen wir an, die Abbildung sei die identische, so entspricht dies in der zweidimensionalen Projektion aus Anhang A der Hauptdiagonalen. Wird nun beispielsweise die Schichtgröße verdoppelt, so ist es offensichtlich, daß sich auch die Anzahl der Links verdoppeln muß, damit die Diagonale vollständig abgedeckt werden kann. Die Anzahl der Links muß folglich linear mit der Schichtgröße wachsen.

Nun ist die Verbindungsmatrix zu Anfang aber keine optimale Lösung, sondern besitzt viel zu viele Links ($n^4 - n^2$ sind überflüssig zur Bildung einer Diagonale). Allerdings ist diese Redundanz notwendig, um dem Algorithmus genügend Spielraum zu geben, die richtige Näherung an die Abbildung zu finden. Würden wir den nachfolgenden höheren Auflösungsstufen nicht ebenfalls eine gewisse Freiheit lassen, so könnte der Algorithmus direkt nach der ersten Näherung abgebrochen werden, da sich die Approximation in den nachfolgenden Stufen nicht mehr verbessern kann. Andererseits sollte der Spielraum für die höher auflösenden Stufen nicht mehr ganz so freizügig bemessen sein wie für die erste Näherung, um den Algorithmus dazu zu bringen, mit den bisherigen Ergebnissen zu arbeiten und sich nicht auf jeder Ebene für eine völlig andere Abbildung zu entscheiden. Weiterhin ist natürlich stets die Speicherkapazität des Rechners im Auge zu behalten.

Faktor	Schichtgröße						
	4	8	16	32	64	128	256
2	256	512	1024	2048	4096	8192	16384
	-	4096	8192	16384	32768	65536	131072
3	256	768	2304	6912	20736	62208	186624
	-	4096	12288	36864	110592	331776	995328
4	256	1024	4096	16384	65536	262144	1048576
	-	4096	16384	65536	262144	1048576	4194304
5	256	1280	6400	32000	160000	800000	4000000
	-	4096	20480	102400	512000	2560000	12800000

Tabelle 4.1: Anzahl der Links in Abhängigkeit von der Schichtgröße, der Verfeinerung und der Startgröße. In der ersten Spalte sind einige Beispiele möglicher Faktoren für die Zunahme der Links, bei Auswahl einer festen Anzahl vorgegeben. In den jeweiligen Zeilen ist die Anzahl der Links in Abhängigkeit von der Schichtgröße eingetragen. Die obere Hälfte einer Zeile bezieht sich auf eine anfängliche Schichtgröße von 4×4 Neuronen, die untere geht von 8×8 Zellen aus.

Die Tabelle 4.1 zeigt deutlich, daß die Wahl einer zu hochauflösenden Schicht für den Anfang und die Beibehaltung zu vieler Links es schnell unmöglich machen kann auch noch Bilder mit 256×256 Pixeln zu matchen und den Matchingprozeß als ganzes sehr speicher- und rechenintensiv macht.

4.2.2 Ähnlichkeiten für Superpixel

Zwar haben wir bisher in recht eleganter Weise erklärt, wie man aus den hochauflösenden Bildern gröbere Darstellungen durch Verwendung von sogenannten Superpixeln gewinnt, allerdings wurde bislang nicht auf das Problem der Beurteilung der Ähnlichkeit zweier Superpixel eingegangen.

Für normale Grauwertbilder kommt man sofort auf die Idee, den Superpixeln einfach einen Grauwert zuzuweisen, der dem Mittelwert der Einzelpixel entspricht und auf der Basis dieser Mittelwerte eine Ähnlichkeitsberechnung durchzuführen.

Spätestens für sehr große Superpixel besitzt ein solcher Mittelwert jedoch keinerlei Aussagekraft, und das Vorgehen erscheint auch schon für die Größenverhältnisse fragwürdig, mit denen das hier beschriebene Matchingverfahren arbeiten soll. Eine Verbesserung ist denkbar, indem man die zur Verfügung stehende Information ausnutzt und zunächst für jeden Pixel in einem Superpixel die Ähnlichkeit zu jedem Pixel in dem anderen Superpixel berechnet. Für jeden Pixel wird das Maximum dieser Ähnlichkeitswerte zu den Pixeln des anderen Superpixels ermittelt. Das Superpixel erhält als Ähnlichkeitswert den Mittelwert dieser Maxima. Man beachte, daß dieses Verfahren gerichtet ist und somit für ein Superpixelpaar zwei Ähnlichkeitswerte liefert. Dies verdoppelt sofort den Speicherbedarf für die Ähnlichkeitswerte, wenn man nicht gewillt ist, bei jedem Lernschritt neuerlich die Ähnlichkeiten zu berechnen. Da der Rechenaufwand ebenfalls sehr hoch ist, dürfte man kaum dazu neigen, das Ganze mehrmals zu rechnen. Bei einem Ausgangsbild von 128×128 Pixeln muß für die vollständig verknüpfte Startschicht die Ähnlichkeit zwischen 2^{28} Merkmalen berechnet werden, dies benötigte, auf einer Sun Workstation vom Typ Sparc 10 ungefähr 20 Minuten.

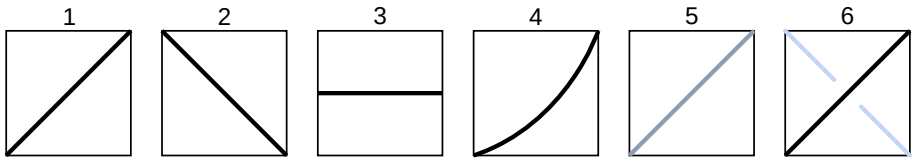

Abbildung 4.6: Superpixel von Kantenbildern, wie ähnlich sind sie?

Betrachten wir nun die Ähnlichkeitsbestimmung für Superpixel von Kantenbildern, so ergeben sich neue Probleme. Wie soll beispielsweise die Ähnlichkeit der in Abbildung 4.6 dargestellten Superpixel beurteilt werden? Alle diese Bilder könnten die gleiche Kante zeigen:

1:2 Horizontale oder vertikale Spiegelung bzw. Rotation um 90° oder 270°.

1:3 Drehung um 45°.

1:4 Leichte Verzerrung.

1:5 Leicht veränderte Perspektive, dadurch erscheint die Kante schwächer.

1:6 Leicht veränderte Perspektive, eine neue schwache Kante wird sichtbar.

Ist man zu freizügig, so ist fast alles ähnlich und der Informationsgehalt gering. Wird sehr restriktiv vorgegangen, so sind auch Merkmale unähnlich, die die gleiche

Kante zeigen. Einen guten Mittelweg zu finden ist schwer und der Algorithmus, der dies leistet, dürfte bereits sehr komplex werden. Überhaupt scheint das gesamte Matching-Verfahren von dem Algorithmus zur Bestimmung der Ähnlichkeiten abzuhängen, das Problem wäre somit nur verlagert.

In einer Versuchsreihe wurden, mit einem Gesamtrechenaufwand mehrerer Tage einer Sparc 10, die beiden oben vorgestellten Verfahren zur Ähnlichkeitsberechnung sowie einige Varianten getestet. Die Versuchsanordnung war dabei so gewählt, daß ein Bild eines Gesichtes mit der vertikalen Spiegelung des selben Bildes gematcht werden sollte. Aus beiden Bildern wurde mittels der Mallatschen Wavelettransformation jeweils eine Mehrskalen-Kantendarstellung gewonnen und diese als Input für das Matchingverfahren verwendet. Das Ergebnis war niederschmetternd. Mit allen Funktionen zur Ähnlichkeitsberechnung wurde auf allen Skalen der Wavelettransformation stets eine identische Abbildung gefunden. Für das erste Verfahren, das die Ähnlichkeitsberechnung auf der Basis von Mittelwerten durchführte, war dies zu erwarten, da fast jeder Superpixel zu jedem anderen ähnlich ist. Das zweite Verfahren, welches das Maximum der Ähnlichkeit einzelner Pixel verwendet, liefert zwar wesentlich weniger sehr ähnliche Superpixelpaare, doch konnte auch hier die signifikante Struktur nicht herauskommen. Die probierten Varianten scheiterten ebenfalls deutlich.

Das endgültige Aus für den Versuch eine Ähnlichkeitsfunktion für das Matchingverfahren zu finden, war gekommen, als die Ähnlichkeitsmatrix Punkt zu Punkt unter Vorgabe der Lösung, also vertikal gespiegelt, gerechnet wurde. Da die Modulus Maxima für das gespiegelte Bild nicht exakt an den gleichen Orten lokalisiert waren, wie die gespiegelten Modulus Maxima des ursprünglichen Bildes, lieferte die Funktion auch hier zu schwache Ähnlichkeiten, wodurch das Ergebnis wieder eine identische Abbildung wurde.

4.2.3 Matching ohne explizite Ähnlichkeitsfunktion

Die Dynamik mit der Ähnlichkeitsmatrix läuft vereinfacht folgendermaßen ab. Die Verbindungen können dort am besten wachsen, wo die Ähnlichkeitmatrix Werte nahe bei eins enthält. Dies hat zur Folge, daß die jeweilige Schicht über diese starken Links einen besonders guten Input bekommt und so die Blobs synchronisiert werden. Besitzt die Ähnlichkeitsmatrix beispielsweise für eine vertikale Spiegelung gute Werte, so werden bevorzugt Links verstärkt, die zu dieser vertikalen Spiegelung gehören. Befindet sich nun in Schicht eins ein Blob links oben, so erhält die zweite Schicht einen besonders guten Input links unten und umgekehrt. Befinden sich die Blobs an den korrespondieren Punkten, so wird die Abbildung verbessert und dadurch werden die Blobs noch stärker synchronisiert. Die Wechselwirkungen zwischen neuronaler Aktivität in den jeweiligen Schichten und Veränderung der Verbindungsstärken wird von der Ähnlichkeitsmatrix bei der Änderung der Links direkt und bei der Aktivitätsänderung indirekt beeinflußt.

Die Ähnlichkeitsmatrix geht als ein individueller Faktor für die Änderung der Verbindungsstärke von jedem Link in den Matchingprozeß ein. Da die Symmetriebrechung durch Rauschen bei der Verwendung von Merkmalen überflüssig ist, kann der Rauschterm ϱ_a^n in Gleichung 4.1 weggelassen werden. Dafür wird das Merkmal f_a^n, das für die Verwendung von Kantenbildern den Modulus Maxima (bzw. dem Mittelwert der Modulus Maxima in einem Superpixel) entspricht, für jedes Neuron additiv in die Schichtdynamik eingebracht werden. Dies ergibt im Endeffekt wieder einen Faktor in der Link-Dynamik 4.4, der diesmal allerdings eng mit der neuronalen Aktivität x_a^n und der Ausgabefunktion σ verknüpft ist.

Die Differentialgleichung 4.1 lautet wie folgt:

$$\dot{x}_a^n = -x_a^n + \gamma \left(\sum_{a'} g(a-a')\sigma(x_{a'}^n) - \beta \sum_{a'} \sigma(x_{a'}^n) \right) - \alpha s_a^n + \varepsilon \sum_{a'} J_{aa'}^{nm} \sigma(x_{a'}^m) + p f_a^n \ . \tag{4.6}$$

Ohne Ähnlichkeitsmatrix werden die Wechselwirkungen bei der Änderung der neuronalen Aktivität direkt durch die Merkmale und indirekt bei der Link-Dynamik beeinflußt. Der Anteil der Merkmale an der Dynamik für die neuronale Aktivität macht sich dadurch bemerkbar, daß die Blobs bevorzugt an den Orten lokalisiert sind, wo sie eine gute Unterstützung durch die Merkmale erhalten. Dadurch ergibt sich eine Bahn, die die Blobs entlanglaufen, wenn es ihnen der Input aus der anderen Schicht erlaubt. Sie werden sich so synchronisieren, daß jeder eine möglichst große Strecke auf seiner Lieblingsbahn zurücklegen kann. Dieses Verhalten prägt die Abbildung.

Für Kantenbilder, wie sie hier verwendet werden, kann man sich gut vorstellen, wie die Blobs die stark ausgeprägten Kanten entlanglaufen und so die Abbildung im bezug auf die relative Anordnung der Kanten in den beiden Schichten aufgebaut wird.

Das Matchen ohne Ähnlichkeitsmatrix macht natürlich auch deren Berechnung und Speicherung überflüssig. Für die Verwendung der Merkmale in der Neuronenaktivität muß lediglich das Bild so durch Superpixel dargestellt werden, daß für jedes Neuron ein entsprechendes Superpixel vorhanden ist. Als Merkmal für das Superpixel wird hier einfach der Mittelwert der einzelnen Pixel verwendet.

4.2.4 Der Matching-Algorithmus

Der Matching-Algorithmus soll mit der vollen Dynamik der laufenden Blobs arbeiten, es werden also zwei Verbindungsmatrizes benötigt. Da sich diese Matrizes individuell entwickeln können, sollten auch zwei Korrelationsmatrizes vorgesehen werden, die der Struktur der Verbindungsmatrizes angepaßt werden. Dies ist notwendig, um nicht zuviel Zeit durch die Suche nach bestimmten Elementen zu verlieren. Die Blobdynamik soll nach einer Verfeinerung nicht wieder von Anfang an berechnet werden, sondern es soll mit den Aktivitäten weitergerechnet werden, die die Neuronen vor der Verfeinerung besaßen. Hierzu werden die Neuronenschichten ebenfalls verfeinert, indem die einzelnen Neuronen zerlegt werden und ihre Aktivität von jeder neuen Zelle übernommen wird.

Zusammenfassend läßt sich der Matching-Algorithmus folgendermaßen beschreiben:

1) Initialisierung
   - 1.1) Initialisierung der Neuronenschichten für eine vorgegebene Startgröße.
   - 1.2) Die beiden Schichten werden vollständig durch Superlinks miteinander verbunden.
   - 1.3) Die Links der Verbindungsmatrix werden auf Null gesetzt.
2) Iteration, bis die Schichtgröße der Bildauflösung entspricht.
   - 2.1) Die Bilder werden, durch Mittelwertbildung, so vergröbert, daß je ein Superpixel mit einer Zelle der Schicht korrespondiert.

2.2) z_1 Iterationen

2.2.1) z_2 Iterationen

2.2.1.1) Die Wanderblob-Dynamik wird nach Differentialgleichung 4.6 simuliert und eventuelle Normierungen der betroffenen Links in der Verbindungsmatrix werden vorgenommen.

2.2.1.2) Berechnung der Korrelation der Ausgabeänderung und Addition dieser zu der jeweiligen Summe $C_{aa'}^{nm}$ für alle Zellen a in Schicht n und alle Zellen a' in Schicht m.

2.2.2) Anwendung der modifizierten Lernregel:

$$\dot{J}_{aa'}^{nm} = \lambda(J_{aa'}^{nm} + \tau)C_{aa'}^{nm} \ . \tag{4.7}$$

2.2.3) Zurücksetzen der Summe $C_{aa'}^{nm}$ auf den Wert Null.

2.3) Die Normierung der Verbindungsmatrizes wird erzwungen.

2.4) Die Superlinks für die Verfeinerung werden ausgewählt.

2.5) Die Superlinks werden verfeinert.

2.6) Die Matrizes für die Korrelation der Ausgabeänderung werden der Struktur der jeweiligen Verbindungsmatrix angepaßt.

2.7) Verfeinerung der Neuronenschichten.

Kapitel 5

Implementationen

In diesem Kapitel sollen die verschiedenen Implementationen beschrieben werden, wie sie im Rahmen dieser Arbeit verwirklicht wurden. Als Plattform wurde zunächst IDL (Interactive Data Language) gewählt, um möglichst schnell einen Prototyp für nähere Untersuchungen und Verbesserungen zu erhalten. Im weiteren Verlauf vollzog sich ein Wechsel von IDL zu **C++** und damit von den starren Matrizes hin zu dynamischen Datentypen, die eine angemessene Behandlung der spärlich besetzten Matrizes erlauben.

An dieser Stelle soll nur auf die programmiertechnischen Problemstellungen eingegangen werden. Die Erläuterung des Matching-Algorithmus, sowie die Behandlung der mathematischen Gesichtspunkte der zugrundeliegenden Dynamiken befindet sich in 4.

Der Schwerpunkt soll hier auf der Reduktion des Speicherbedarfs für große Matrizes liegen, die sich mit fortschreitender Iteration ausdünnen und so zu spärlich besetzten Matrizes werden. Weiterhin sollen Möglichkeiten beleuchtet werden, die es erlauben, das Problem der lokalen Minima in feinauflösenden Optimierungsproblemen zu entschärfen.

5.1 Simulation der Differentialgleichungen

Für die Implementation der Simulation einer Differentialgleichung wurde in dieser Arbeit stets das Euler-Verfahren verwendet. Dies beruht auf der folgenden Äquivalenz:

$$\begin{aligned} \dot{x}(t) &= f(x(t)) \\ \frac{\Delta x(t)}{\Delta t} &= f(x(t)) \\ \frac{x(t+\Delta t) - x(t)}{\Delta t} &= f(x(t)) \\ x(t+\Delta t) &= x(t) + \Delta t f(x(t)) \ . \end{aligned}$$

Dies Verfahren ergibt für genügend kleine $\Delta t < 1$ eine gute Näherung der verwendeten Differentialgleichungen und ist leicht zu berechnen.

5.2 Prototyp in IDL

Interactive **D**ata **L**anguage ist ein Paket für die interaktive Analyse und Visualisierung wissenschaftlicher oder technischer Daten. Diese Paket vereint eine vektor- bzw. matrixorientierte Programmiersprache mit numerischen Hilfmitteln und graphischen Visualisierungswerkzeugen. IDL ist eine komplette, strukturierte Programmiersprache, die sowohl die interaktive Verarbeitung von Daten, wie auch die Erzeugung komplexer Funktionen, Prozeduren und Anwendungen unterstützt. Im Quelltext vorhandene Module werden vor dem ersten Gebrauch in jeder Sitzung übersetzt und können fortan mit Parameterübergabe aufgerufen werden. Die Operatoren und Funktionen von IDL arbeiten auf ganzen Matrizes, wodurch der Entwurf von Programmen, die auf Vektoren und Matrizes arbeiten, sehr vereinfacht wird. Gerade wegen dieser Matrixverarbeitung und den sehr umfangreichen und leicht zu handhabenen Werkzeugen zur Visualisierung von Daten wurde IDL für den Entwurf des Prototyps und der sich anschließenden Untersuchung und Verfeinerung dieses Programmes ausgewählt.

5.2.1 Erste Implementation

Als erster Schritt wurde zunächst nur die Dynamik der Wanderblobs realisiert. Dies entspricht der Gleichung 4.1, wobei $\varepsilon = p = \varrho_a^n = 0$ gilt. Das Programm erlaubt Untersuchungen der Abhängigkeiten zwischen Parameterwahl und Verhalten der Aktivitätsflecken in Wachstum und Fortbewegung. Darüber hinaus bietet es die Möglichkeit, sich an der entstehenden Ordnung zu erfreuen, die bereits durch so relativ einfache Gleichungen erzeugt werden kann, wie sie in 4.1 beschrieben werden.

Dieses Programm wurde so erweitert, daß es zwei unabhängige Schichten mit bewegten Konzentrationszentren der Aktivität erlaubt. Für die Dynamik der beiden Schichten wurde jeweils ein Kopplungsterm eingeführt, der beschreibt, in welcher Form Aktivität von der einen Schicht in die andere propagiert werden kann. Weiterhin wurden zwei Verbindungsmatrizes verwendet, die nach der Gleichung 4.4 gelernt wurden.

Korrelation der Ausgabeänderung und Lernschritte

Um die Korrelation der neuronalen Ausgabeänderung zu bestimmen, wurde die Differenz als leicht zu berechnende Näherung für die Ableitung verwendet. Dies ist vertretbar, da es auf den exakten Wert der Ableitung nicht ankommt Da die Korrelation aus der Änderung der Aktivität in den beiden Schichten berechnet wird, ist leicht einzusehen, daß die Korrelation eines einzelnen Zeitschrittes recht wenig Information trägt, weil die Aktivität an den meisten Stellen fast konstant geblieben ist. Aus diesem Grund wird die Korrelation der Ausgabeänderung über mehrere Zeitschritte aufsummiert und erst nach der Iteration der Blobdynamik über diese Zeitschritte die Lernregel einmal benutzt mit der Summe der Korrelationen und einer mit die Anzahl der verstrichenen Zeitschritte multiplizierten Lernkonstanten. Diese Verfahrensweise ermöglicht die Einsparung vieler Lernschritte, die einen Aufwand von n^2 haben, wenn n für die Anzahl der Elemente in einer Schicht steht. Direkt nach einem Lernschritt wird die Matrix für die Korrelationssumme der Ausgabeänderung wieder auf Null gesetzt.

Normierung

Direkt im Anschluß an einen Lernschritt werden die negativen Verbindungsgewichte wieder auf Null gesetzt, da negative Verbindungsstärken für diesen Algorithmus keinen besonderen Sinn ergeben und seinen Ablauf behindern. In der Startphase des Dynamik Link Matching sind die Blobs noch nicht miteinander synchronisiert, so daß es zu negativen Korrelationen der Ausgabeänderung und somit zu negativen Verbindungsgewichten kommen könnte, die dann bei anschließender Synchronisation der Aktivitätshügel erst mühsam abgebaut werden müßten, bevor der positive Charakter dieser Verbindungen in Erscheinung treten würde.

Wie in Abschnitt 4.1.3 beschrieben, gibt es keine automatische „postsynaptische“ Normierung. Die Normierung wird erst vorgenommen, wenn der Input, den eine Zelle aus der anderen Schicht erhält, den Wert eins überschreitet. In diesem Fall wird sofort normiert und der Input der Zelle auf eins gesetzt. Durch diese Vorgehensweise kann es dazu kommen, daß es Verbindungen gibt, deren Gewichte vorübergehend größer als eins sind. Die Dynamik der laufenden Blobs ist so beschaffen, daß nur Links zwischen Zellen, die gleichzeitig aktiv sind, verstärkt werden. Bei einer derartigen Konstellation würde eine starke positive Verbindung zu einem Input größer eins und somit zu einer Normierung führen. Somit ist ein unbegrenztes Wachstum der Verbindungen ausgeschlossen.

5.2.2 Berechnung der Merkmalsähnlichkeiten

In früheren Arbeiten wurden die Merkmale auf zwei verschiedene Arten in die Dynamik eingebracht. Für beide müssen die Ähnlichkeiten der Merkmale mit einer geeigneten Funktion bestimmt werden.

- Mit diesen Ähnlichkeitswerten wird die Verbindungsmatrix initialisiert und in der weiteren Berechnung werden diese Ähnlichkeiten nicht mehr benutzt.
- Die Lernregel enthält einen Faktor, der dem Ähnlichkeitswert zwischen den beiden zugehörigen Punkten entspricht.

Im Rahmen dieser Arbeit wurden die Merkmale auf eine andere Art an die Dynamik angebunden. Wie in Abschnitt 4.2.3 beschrieben, ist es möglich, anstelle eines Ähnlichkeitswertes für die Verbindungen die Merkmale als additive Komponente direkt an die Dynamik der Neuronen anzukoppeln.

Für den Prototyp wurde zunächst eine Ähnlichkeitsfunktion benutzt, deren Werte als Faktor in die Lernregel eingehen. Die Anbindung der Merkmale an die Dynamik der Neurone wird in der **C++**-Implementation verwendet, siehe Abschnitt 4.2.3.

5.2.3 Versuche zur Reduktion des Speicherbedarfs

Besondere Bemühungen wurden unternommen, um den Speicherbedarf für die vierdimensionalen Abbildungsmatrizes zu reduzieren. Wie in Abschnitt 4.2.1 beschrieben, würde der Speicherbedarf einer Verbindungsmatrix, bei einer Bildgröße von 128×128 Pixeln, ein Gigabyte betragen. Realistische Bildgrößen beginnen allerdings bei ungefähr 512×512 Punkten.

Erste Versuche mit dem IDL-Prototyp ergaben, daß sich die Verbindungsmatrix relativ schnell derart vorstrukturiert, daß es möglich ist, viele Werte in den Berechnungen auszusparen. Bereits nach wenigen Lernschritten lassen sich Verbindungen identifizieren, die zu der entstehenden Abbildung keinen wesentlichen Beitrag leisten und so nicht länger berücksichtigt werden müssen. Bei dieser Methode lassen sich recht schnell ungefähr 75% der Verbindungen aus den Matrizes eliminieren, ohne daß die Qualität der gefundenen Abbildung wesentlich verschlechtert wird. Probleme bereitet dieses Verfahren jedoch, da es keine Möglichkeit gibt, die Verbindungsmatrix vom Beginn der Berechnungen an zu dezimieren, ohne die Flexibilität der Abbildungen sehr wesentlich zu beeinflussen. Dennoch zeigten diese Untersuchungen sehr eindrucksvoll, daß die großen Verbindungsmatrizes und die mit ihnen verbundenen Strukturen stark reduzierbar sind. Bei den Versuchen wurde auch deutlich, daß das bisherige Programm auf der Basis IDL nur ein Prototyp sein konnte, denn soweit dynamische Strukturen in IDL überhaupt realisierbar sind, werden sie auf jeden Fall unerträglich langsam.

Ein ganz anderer Ansatz das Problem zu lösen, setzte bei der speziellen Aufgabenstellung Kantenbilder aufeinander abzubilden, an. Hierbei werden Kante-zu-Nicht-Kante-Verbindungen als unähnlich angesehen und somit nicht oder nur sehr schwer gelernt. Dies ist selbstverständlich symmetrisch, so daß auch Nicht-Kante-zu-Kante-Verbindungen unerwünscht sind. Die Kanten sind in natürlichen Bildern in sehr viel geringerer Zahl vorhanden als die Punkte, an denen keine Kanten lokalisiert sind. Aufgrund dieser Tatsache wird klar, daß die Links, die Nicht-Kante-zu-Nicht-Kante-Verbindungen repräsentieren, nur sehr geringe Information tragen, da es nicht möglich ist, zu sagen, welches Nicht-Kante-Element auf welches Nicht-Kante-Element abgebildet werden muß, ohne die endgültige Abbildung zu kennen. Aufgrund dieser Überlegung wurde der Prototyp so umgeschrieben, daß es möglich war, die Berechnungen der Korrelation der neuronalen Ausgabeänderung und das Lernen nur noch für die Verbindungen vorzunehmen, die Kante-zu-Kante-Verbindungen repräsentieren. Mit dem so entstandenen Programm können Kantenbilder erfolgreich gematcht werden, allerdings sehen die entstandenen Abbildungen „löchrig“ aus, was natürlich zu erwarten war, da bei jeder vollständigen Abbildung auch die Nicht-Kante-zu-Nicht-Kante-Verbindungen berücksichtigt werden müssen.

5.2.4 Matchen mit Mehrskalen-Kanten

Bei der Verwendung mehrer Ebenen der Wavelettransformation ergeben sich Probleme mit den vielen lokalen Maxima des Ähnlichkeitsgebirges für die hochauflösenden Stufen. Ein erster Versuch, dies zu umgehen, bestand darin, die Abbildung zunächst auf einer niederfrequenten Ebene zu berechnen und mit ihr die Verbindungsmatrix der nächsthöheren Stufe zu initialisieren. Die ersten Experimente hierzu scheiterten an der Robustheit des Dynamik Link Matching mit der Dynamik der laufenden Blobs. So war es nicht möglich, das Auffinden der Abbildung durch diese Vorinitialisierung der Verbindungsmatrizes wesentlich zu beeinflussen. Allerdings muß berücksichtigt werden, daß bei diesem Test nur eine Bildgröße von 16×16 Punkten benutzt wurde. Für größere Bilder wäre eins der oben erläuterten Verfahren zur Reduktion des Speicherbedarfs für die großen Verbindungsmatrizes notwendig. Die einzige zu diesem Zeitpunkt lauffähige Methode, das Matchen unter ausschließlicher Verwendung der Kante-zu-Kante-Verbindungen, hat den bereits erwähnten Nachteil, daß die Abbildungen unvollständig sind. Da die Lokalisation der Kante-zu-Kante-Verbindungen höherer Ebenen nicht exakt mit der tieferer Ebenen übereinstimmt, wird die Verbindungsmatrix ausgedünnt. Zwar wäre es möglich gewesen, die Verbindungsmatrix beispielsweise mit einem Gaußkern zu glätten, um so einen

Teil der Löcher zu beseitigen und damit die Ausgangssituation für den nächsten Matching-Prozeß zu verbessern. Jedoch darf nicht vergessen werden, daß dies Programm nur als ein Prototyp dienen sollte, der gerade dazu bestimmt war, solche Probleme aufzudecken.

5.3 Neuimplementation in C++

Aufgrund der in Abschnitt 4 beschrieben Überlegungen und den mit dem Prototyp gemachen Erfahrungen wurden die Anforderungen an eine C++-Implementation festgelegt. Zusammenfassend lassen sich hierbei die folgenden Entwurfskriterien festhalten:

- Die volle Abbildung soll schrittweise approximiert werden. Hierzu werden Hilfmittel wie Superpixel und Superlinks verwendet.
- Für die spärlich besetzten Matrizes muß eine geeignete Kodierung gefunden werden, um sie effizient im Speicher ablegen zu können.
- Der Selektionsprozeß, der entscheidet, welche Links verfeinert werden, muß in der Lage sein die wichtigen Strukturen der Abbildung zu erhalten und dabei genügend Links zu eliminieren, um die Matrizes handhabbar zu machen.
- Die Merkmale werden direkt in die neuronale Aktivität eingebracht, um auch Kantenbilder erfolgreich matchen zu können.

5.3.1 Kodierung der spärlich besetzten Matrizes

Wie bereits erläutert, muß der Speicherbedarf für die Verbindungs- und Korrelationsmatrizes reduziert werden. Hierbei reicht es nicht aus, allein die Anzahl der Elemente in diesen Strukturen zu verringern, sondern es muß eine geeignete Kodierung gefunden werden, die es erlaubt, mehrere Verbindungen so zu gruppieren, daß sie ohne großen Overhead repräsentiert werden können.

Datenstruktur

Bei der Betrachtung des Verfeinerungsverfahrens ergibt sich eine mögliche Repräsentation der Verbindungsgruppen von selbst. Es ist äußerst naheliegend, die bei der Zerlegung eines Superlinks neu generierten Links zu einer Gruppe zusammenzufassen und diese dadurch zu repräsentieren, daß die Links geordnet in einem linearen Feld abgelegt und nur der Index der ersten Verbindung sowie die Anzahl der Links, die aus einem Superlink entstanden sind, hinterlegt werden. Die Indizes der anderen Verbindungen lassen sich auf einfache Weise aus dem gespeicherten Index und der Anzahl der neugebildeten Links berechnen. Wird die Zerlegung aller Superlinks nach dem gleichen einheitlichen Prinzip vorgenommen, so ist die Anzahl der bei der Verfeinerung erzeugten Verbindungen immer die gleiche und muß somit nur einmal in einer übergeordneten Struktur abgelegt werden. In der gleichen höheren Datenstruktur werden die vielen einzelnen Verbindungsbündel verwaltet und dadurch die gesamte Verbindungsmatrix repräsentiert.

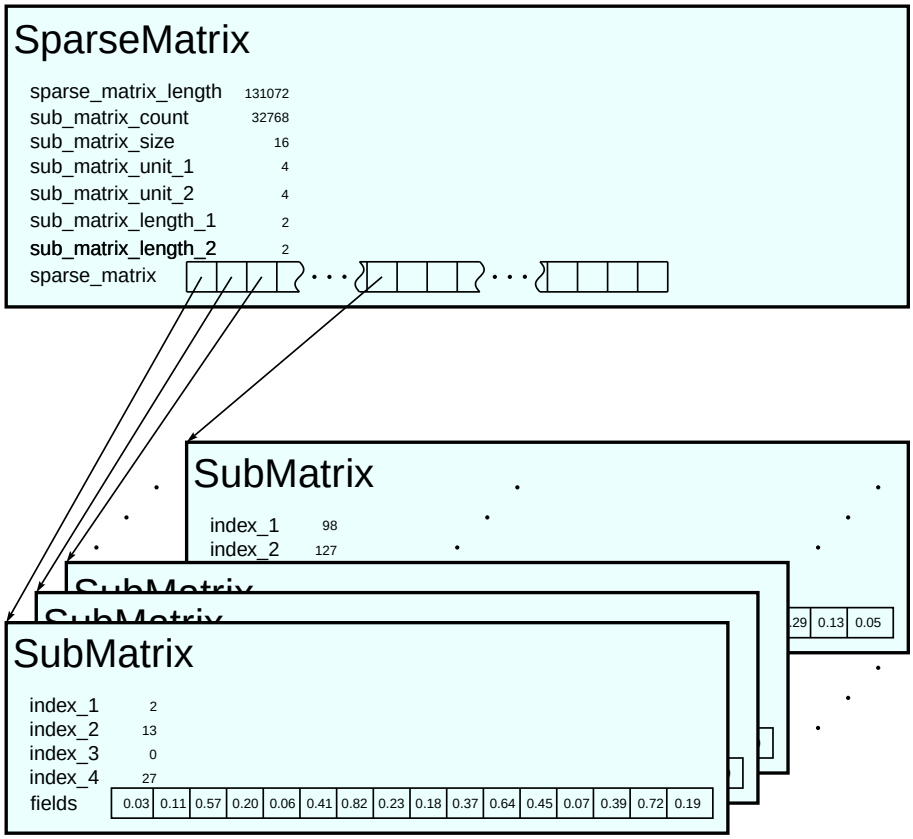

Abbildung 5.1: Repräsentation einer spärlich besetzten Matrix

Zusammenfassend läßt sich die Verfeinerung und Kodierung eines Superlinks wie folgt beschreiben:

1) Zerteile die beiden Superpixel, die das Superlink verbindet in 2×2 Pixel oder eventuell kleinere Superpixel.

2) Verbinde jedes der 2×2 Pixel des Superlinks aus Schicht X mit jedem Pixel des Y-Superpixels, es ergeben sich 2^4 neue Verbindungen.

3) Sei $\vec{i} = (i_1, i_2, i_3, i_4)$ der Index des Superlinks, so besitzt das Link, das die Pixel links oben in den beiden Superpixeln verbindet, den Index $\vec{j} = (2i_1, 2i_2, 2i_3, 2i_4)$, da sich alle Seitenlängen verdoppeln, wenn alle Superlinks in 2×2 Pixel aufgeteilt werden.

4) Die $2^4 = 16$ neuen Links, sowie der Index $\vec{j}$ werden in die entsprechenden Felder einer Instanz der Klasse SubMatrix eingetragen.

5) Die Felder einer Instanz der Klasse SparseMatrix werden folgendermaßen geändert:

   - Dem Element des Feldes sparse_matrix, das den Index sub_matrix_count trägt wird ein Zeiger auf die Instanz aus Punkt 4 zugewiesen. Die Indizes des Feldes sparse_matrix laufen von Null bis sub_matrix_count_1.

- Der Zähler sub_matrix_count wird um eins erhöht.
- Die Elemente sub_matrix_length_1 und sub_matrix_length_2 erhalten den Wert 2, da die Superpixel in 2×2 Pixel zerlegt wurden.
- Den Elemente sub_matrix_unit_1 und sub_matrix_unit_2 wird der Wert 4 zugewiesen, da jedes Superpixel in 4 neue Pixel zerlegt wurde.
- Das Elemente sub_matrix_size erhält den Wert 16, da insgesamt 16 neue Links erzeugt wurden.

Die Elemente sub_matrix_unit_1, sub_matrix_unit_2 und sub_matrix_size können zwar aus den Werten von sub_matrix_length_1 und sub_matrix_length_2 berechnet werden, da sie aber oft benötigt werden und nur einmal pro spärlicher Matrix gespeichert werden müssen, wird das Hinterlegen bevorzugt.

Wie Abbildung 5.1 zeigt, werden bei der im Rahmen dieser Arbeit vorgenommenen Implementation die Bündel von Links in einzelnen Instanzen der Klasse **SubMatrix** abgelegt. Zu der Klasse SubMatrix gehören die vier Komponenten des Index der ersten Verbindung und ein Feld, das die Werte der einzelnen Links aufnimmt. Die ersten beiden Komponenten des Index der Verbindung bestehen aus dem Index des Superpixels in der Bildebene, der Schicht X. Die letzten Komponenten werden aus dem Index in der Objektebene, also Schicht Y, gebildet. Die Klasse **SparseMatrix** enthält ein Array, das eine feste Länge besitzt, die in `sparse_matrix_length` abgelegt ist. In diesem Array werden Zeiger auf Instanzen der Klasse Submatrix abgelegt. Der Zähler `sub_matrix_count` speichert die genaue Anzahl dieser SubMatrizes. Die Anzahl der Links, die in einer SubMatrix abgelegt sind wird in `sub_matrix_size` hinterlegt. Die Elemente `sub_matrix_unit_1`, `sub_matrix_unit_2`, `sub_matrix_length_1` und `sub_matrix_length_2` enthalten die Grundlängen des zweidimensionalen Feldes, in das ein Superpixel zerlegt wird, bzw. die Anzahl der Elemente, die ein solches Feld enthält.

Struktur der Matrix für die Korrelation der Ausgabeänderung

Bisher wurde nur die Kodierung der Verbindungsmatrix behandelt, aber auch bei der Berechnung der Korrelation der neuronalen Ausgabeänderung wird eine vierdimensionale Matrix verwendet, die den gleichen Speicherbedarf wie eine Verbindungsmatrix besitzt. Da die Korrelation der Änderung der Neuronenausgabe in der Lernregel verwendet wird, ist es zweckmäßig, daß sie die gleiche Struktur hat wie die Verbindungsmatrix. Anderfalls müßte zu jedem Link der entsprechende Eintrag in der Matrix für die Korrelation gesucht werden, was mit sehr viel Aufwand verbunden wäre und so die Bearbeitung der Lernregel verlangsamen würde. Wird die Struktur der Korrelationsmatrix an die Struktur der Verbindungsmatrix angepaßt, so erfordert dies die Verwendung zweier Matrizes, um sicherzustellen, daß bei unterschiedlicher Entwicklung der beiden Verbindungsmatrizes jeweils eine Korrelationsmatrix für die Anpassung zur Verfügung steht. Es werden somit nicht nur zwei Verbindungsmatrizes unterschieden, eine von Schicht X nach Schicht Y und eine für die umgekehrte Richtung, sondern für jede Richtung wird auch eine eigene Matrix für die Korrelation der neuronalen Ausgabeänderung verwendet.

Bei jeder Verfeinerung der Verbindungsmatrizes müssen auch die Elemente der Korrelationsmatrizes zerlegt werden. Dies wird erreicht, indem nach der Verfeinerung einer Verbindungsmatrix ihre neue Struktur auf die entsprechende Korrelationsmatrix übertragen wird. Hierzu wird mit dem Index der ersten Verbindung, die bei der Verfeinerung eines Superlinks neu generiert wurde, nicht nur eine Instanz der Klasse SubMatrix für die Verbindungsmatrix, sondern darüberhinaus eine

zweite Instanz für die zugehörige Korrelationsmatrix gebildet. Werden die Verfeinerungen ausschließlich nach einem Lernschritt durchgeführt, so können die Feldelemente der Instanz für die Korrelationsmatrix einfach auf Null gesetzt werden, wie dies vom Matching-Algorithmus gefordert wird.

5.3.2 Selektion

Für die hier beschriebene Implementation wurde die Auswahl einer festgelegten Anzahl der stärksten Links ohne Berücksichtigung topologischer Gesichtspunkte verwendet. Besonderes Augenmerk muß dabei auf die große Anzahl von auszuwählenden Links gerichtet werden. Um die Selektion möglichst schnell zu machen wird ein Heap verwendet, der die besten Verbindungen aufnimmt und so geordnet ist, daß an seiner Spitze stets das kleinste Element steht. Bei den weiteren Vergleichen muß nur geprüft werden, ob das Element größer als das Minimum der bisher besten Links ist. Die Aufnahme einer neuen Verbindung in diese Bestenliste geschieht einfach durch Austausch mit dem bisherigen Minimum und anschließenden Absenken des Elementes an die richtige Position, nach dem bekannten Sift-Algorithmus, siehe [Knu73].

5.3.3 Initialisierung der spärlichen Matrizes

Die Verbindungsmatrizes werden zunächst so initialisiert, daß für jede Zelle der einen Schicht ein Link zu jeder Zelle der anderen Schicht vorhanden ist. Diese Verbindungsmatrix muß in der Datenstruktur der spärlich besetzten Matrizes korrekt abgelegt werden. Die Repräsentation der Links läßt sich folgendermaßen veranschaulichen:

1) Die Zellen jeder Schicht werden zu Superzellen zusammengefaßt, ganz genau so wie aus Pixeln Superpixel werden.

2) Alle Superzellen der einen Schicht werden mit allen Superzellen der anderen Schicht durch je ein Superlink verbunden.

3) Jedes Superlink wird zerlegt und die dabei entstehenden Links werden in einer Instanz der Klasse SubMatrix abgelegt.

4) Alle Links werden mit dem Wert Null initialisiert.

5) Die einzelnen SubMatrix-Instanzen werden in einer Instanz der Klasse SparseMatrix verwaltet.

Nachdem auf diese Weise die Verbindungsmatrizes erzeugt wurden, wird ihre Struktur auf die entsprechende Korrelationsmatrix übertragen. Die Elemente der Korrelationsmatrix erhalten ebenfalls den Wert Null.

Kapitel 6

Resultate

6.1 Untersuchung der Invarianz durch Leistungsspektrum

Wie in Abschnitt 2.1.2 geschildert, ist es möglich, die Translations-, Rotations- und Skalierungsinvarianz mittels des Leistungsspektrums zu erreichen. Im Rahmen dieser Arbeit wurde untersucht, wie sich diese Transformation für die Objekterkennung eignet. Bei der Objekterkennung ist es wichtig, daß beurteilt werden kann, ob ein abgebildetes Objekt mit einem bekannten Objekt übereinstimmt. Die Erkennung soll unabhängig von Ort, Perspektive, Beleuchtung und teilweiser Verdeckung möglich sein.

Zunächst wurde am Beispiel der Rotation untersucht, welche Probleme die für diskrete Bilder unvermeidliche Interpolation bereitet. Zu diesem Zweck wurde ein segmentiertes Gesicht um den Bildmittelpunkt gedreht. Die Rotation erfolgte in einzelnen Gradschritten von 0° bis 90°. Hierbei tauchte zunächst das Problem auf, daß 128×128 Bilder kein Pixel im Mittelpunkt besitzen, die Fouriertransformation aber so berechnet wird, daß ihr Mittelpunkt auf einen Bildpunkt fällt. Dadurch unterscheiden sich bereits eine Drehung um den Winkel α mit anschliessender Fouriertransformation und eine Fouriertransformation mit anschliessender Drehung um den Winkel $-\alpha$. Dies Problem wurde umgangen, indem die Eingabebilder auf 127×127 Pixel reduziert wurden. Nun ergab sich bei einer Umordnung in Polarkoordinaten auf 127×127 Bildpunkte das Problem, daß die Standardwinkel wie beispielsweise 90° zu einer Verschiebung führten, die nicht durch eine einfache Shift-Operation rückgängig gemacht werden konnten. Aus diesem Grund wurde bei der Umordnung das Bild wieder auf 128×128 Pixel abgebildet. Die Bilder wurden weiterhin so segmentiert, daß das Objekt nur in der Mitte des Bildes präsent war und so bei der Rotation keine zusätzlichen Fehler durch verschwindende bzw. neu erscheinende Bildbereiche entstanden. Die Segmentationsmaske wurde mit einer Gaußfunktion geglättet, um Artefakte bei der Fouriertransformation (eine senkrechte Kante enthält alle Frequenzen) zu verhindern. Durch diese Vorgehenweise erreichte die Transformation für die Standardwinkel $90°, 180°$ und 270° das optimale Ergebnis, die Transformierten waren mit der Transformierten des nicht gedrehten Bildes identisch. Die Abbildung 6.1 zeigt den Fehlerverlauf für die Winkel zwischen 0° und 90°, dabei wurde der Fehler mittels $\sqrt{\sum_i (x_i - y_i)^2}$ berechnet.

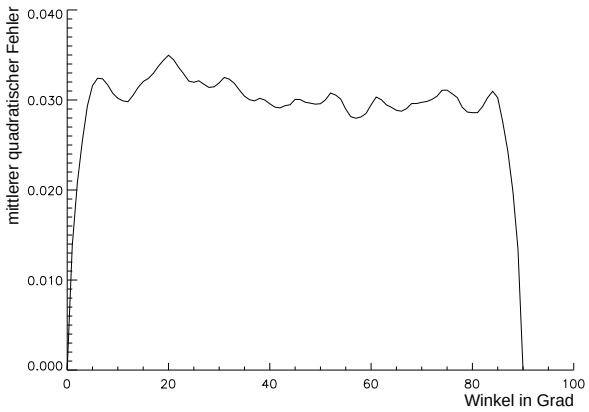

Abbildung 6.1: Fehlerverlauf der Transformationsergebnisse, in Abhängigkeit vom Drehwinkel.

Weiterhin wurden zwei Abbilder von Gesichtern miteinander verglichen, siehe Abbildung 6.2. Die Transformation wurde jeweils für das segmentierte und das unsegmentierte Gesicht berechnet.

Die folgende Tabelle zeigt den Fehler für die verschiedenen Kombinationen:

	Ute seg.	Ute unseg.	Sarah seg.	Sarah unseg.
Ute seg.	-	0.183873	0.131656	0.134528
Ute unseg.	0.183873	-	0.253201	0.125931
Sarah seg.	0.131656	0.253201	-	0.195224
Sarah unseg.	0.134528	0.125931	0.195224	-

Tabelle 6.1: Fehler für den Vergleich einiger Bilder (seg. steht für segmentiert und unseg. für unsegmentiert).

Die Werte in Tabelle 6.1 zeigen deutlich, daß diese Transformation für eine Objekterkennung ungeeignet ist. Dies liegt an dem relativ großem Fehler für den Vergleich zwischen dem segmentierten und unsegmentierten Gesicht. Die Objekterkennung soll unabhängig von dem Hintergrund sein. Die Probleme mit dem Hintergrund waren zu erwarten, da sich bei der Fouriertransformation eine solche lokale Änderung in jeder einzelnen Komponente wiederfindet.

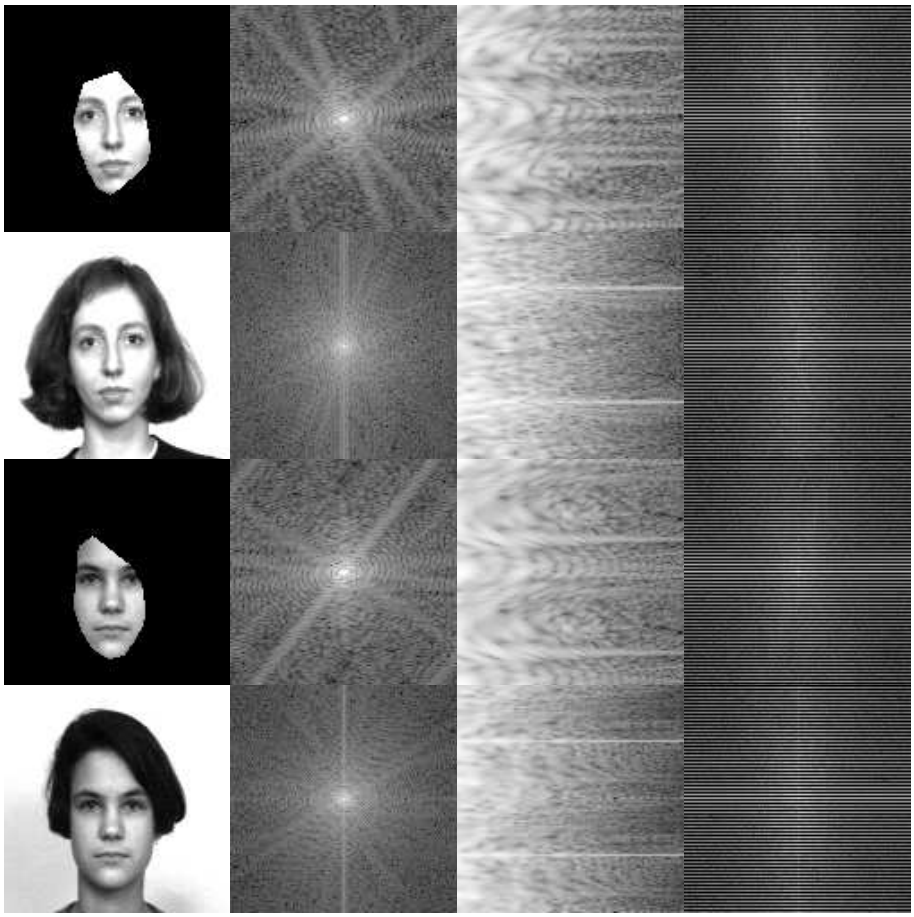

Abbildung 6.2: Die Bilder einer Spalte zeigen (von links nachs rechts) untransformiertes Bild, Betrag der Fouriertransformation, Darstellung des Leistungsspektrums in Polarkoordinaten mit logarithmischer Skala für den Radius, Leistungsspektrum des Bildes der vorherigen Spalte.

6.2 Matching von Kantenbildern

Der im Kapitel 4 beschriebene Matching-Algorithms wurde, wie in 5.3 ausgeführt, implementiert. Nachdem das Matching unter Verwendung einer Ähnlichkeitsfunktion keine brauchbaren Ergebnisse lieferte, sollen hier nun die Ergebnisse für das Matching mit der Kopplung der Merkmale an die Neuronendynamik vorgestellt werden.

Die Abbildungen 6.3 bis 6.14 zeigen die Ergebnisse eines Matchings zwischen zwei identischen Bildern, für die ersten sechs Skalen der Mallatschen Wavelettransformation. Es gehören jeweils zwei aufeinanderfolgende Abbildungen zu einem Matching-Prozeß. Die erste dieser beiden Abbildungen zeigt die Modulus Maxima der Bilder, für die das Matching erstellt wurde, sowie die zweidimensionalen Projektionen der Abbildung auf den fünf Feinheitsgraden (von 8×8 bis 128×128 Bildpunkten). Die verschiedenen Feinheitsgrade sind in einem Bild überlagert, es beginnt oben links

mit 8×8 Bildpunkten, darunter ist das Ergebnis für 16×16 Bildpunkte zu sehen und endet schließlich bei 128×128 Bildpunkten. Die zweite Abbildung zeigt die einzelnen Feinheitsgrade in der Darstellung als Karte. Es beginnt oben links mit 8×8 Bildpunkten, daneben ist das Ergebnis für 16×16 Bildpunkte zu sehen und endet unten links mit 128×128 Bildpunkten.

Die Ergebnisse eines Matchings für die vertikale Spiegelung eines Bildes werden in den Abbildungen 6.16 bis 6.21 gezeigt, diesmal ist nur die Darstellung in Form von Karten abgebildet. Anhand der zweidimensionalen Projektionen wurde für jede Abbildung überprüft, daß der Matching Prozeß jeweils eine vertikale Spiegelung gefunden hat. Die Anordnung der Feinheitsgrade in den Abbildungen ist die gleiche wie bei den entsprechenden Abbildungen für die Ergebnisse des oben beschriebenen Matchings. Die Abbildung 6.15 zeigt die einzelnen Modulus-Maxima-Bilder, die für das Matching verwendet wurden.

Die Abbildungen 6.23 bis 6.28 zeigen die Ergebnisse eines Matchings zwischen zwei verschiedenen Ansichten eines Gesichts. Die Darstellung ist die selbe, wie für die vertikale Spiegelung. Die Kantenbilder in Form der Modulus Maxima sind in Abbildung 6.22 zu sehen.

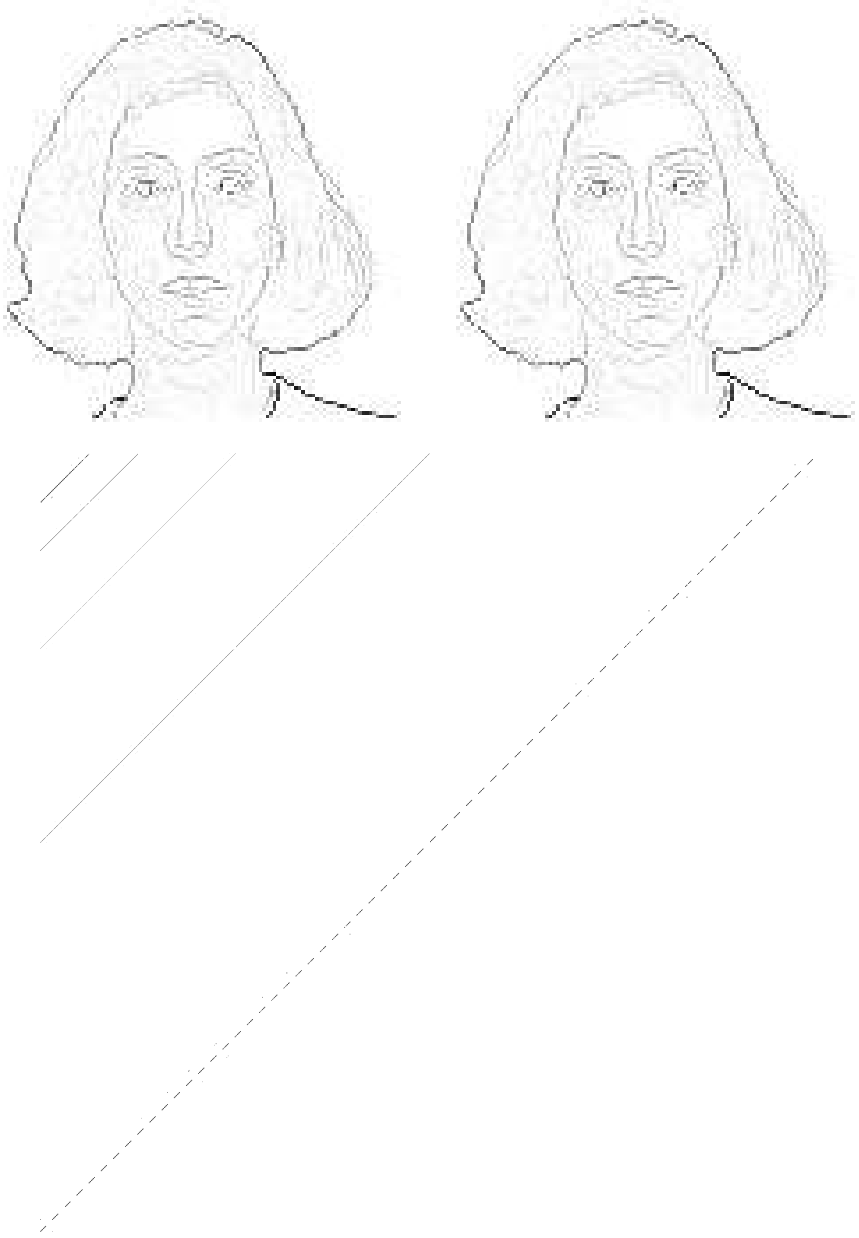

Abbildung 6.3: Projektion der Abbildung Ute0 auf Ute0, für die Skalierung 2^0 der dyadischen Wavelettransformation nach Mallat.

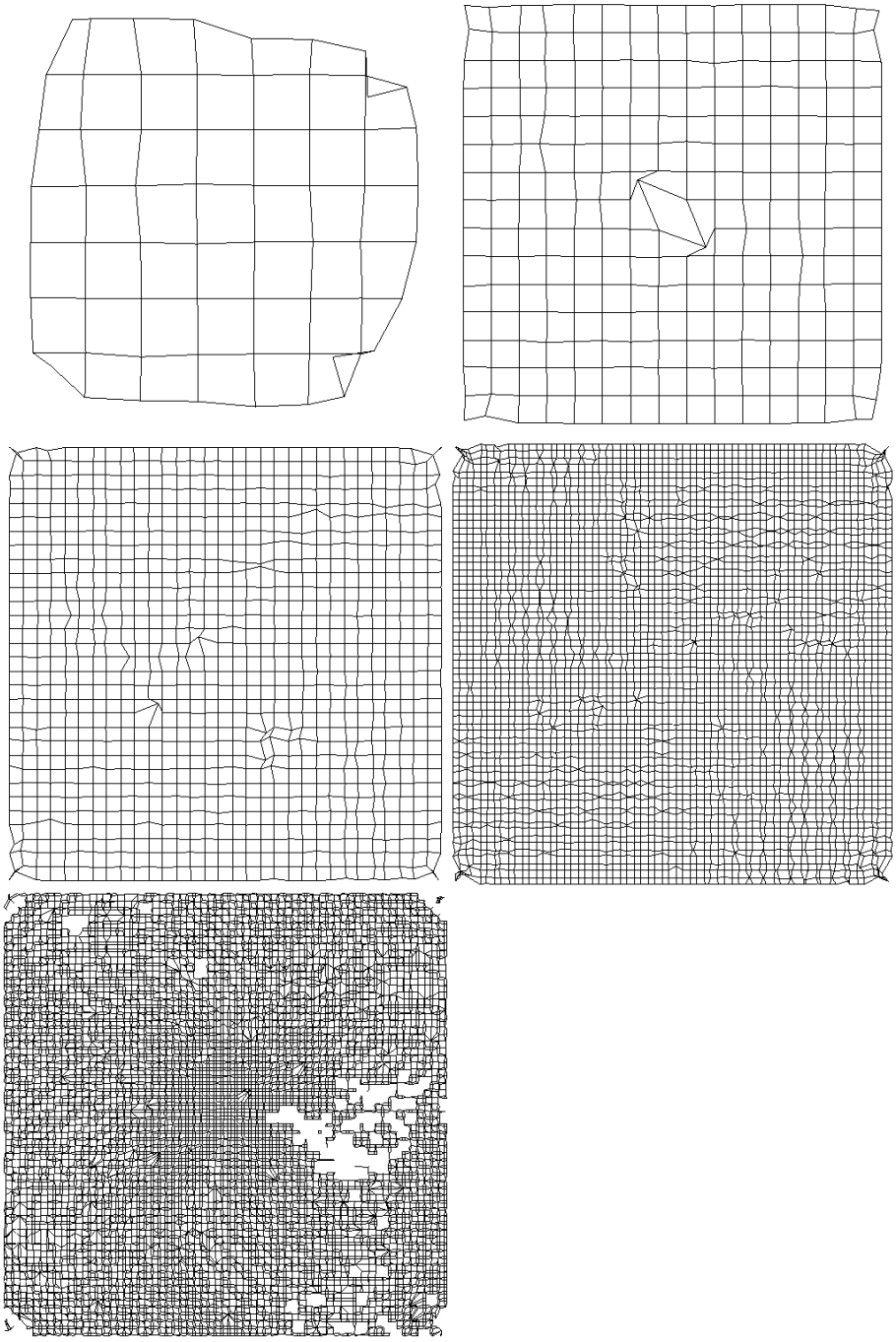

Abbildung 6.4: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^0 der dyadischen Wavelettransformation nach Mallat.

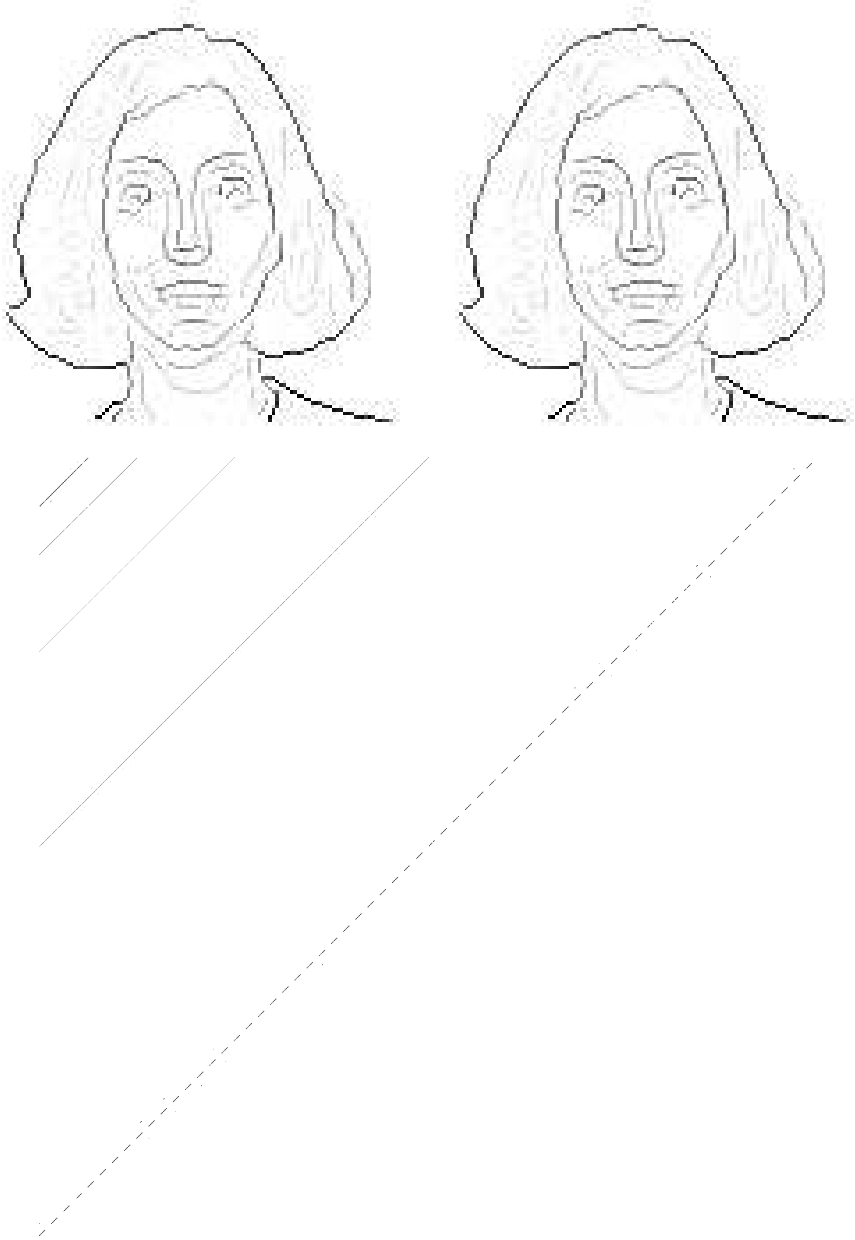

Abbildung 6.5: Projektion der Abbildung Ute0 auf Ute0, für die Skalierung 2^1 der dyadischen Wavelettransformation nach Mallat.

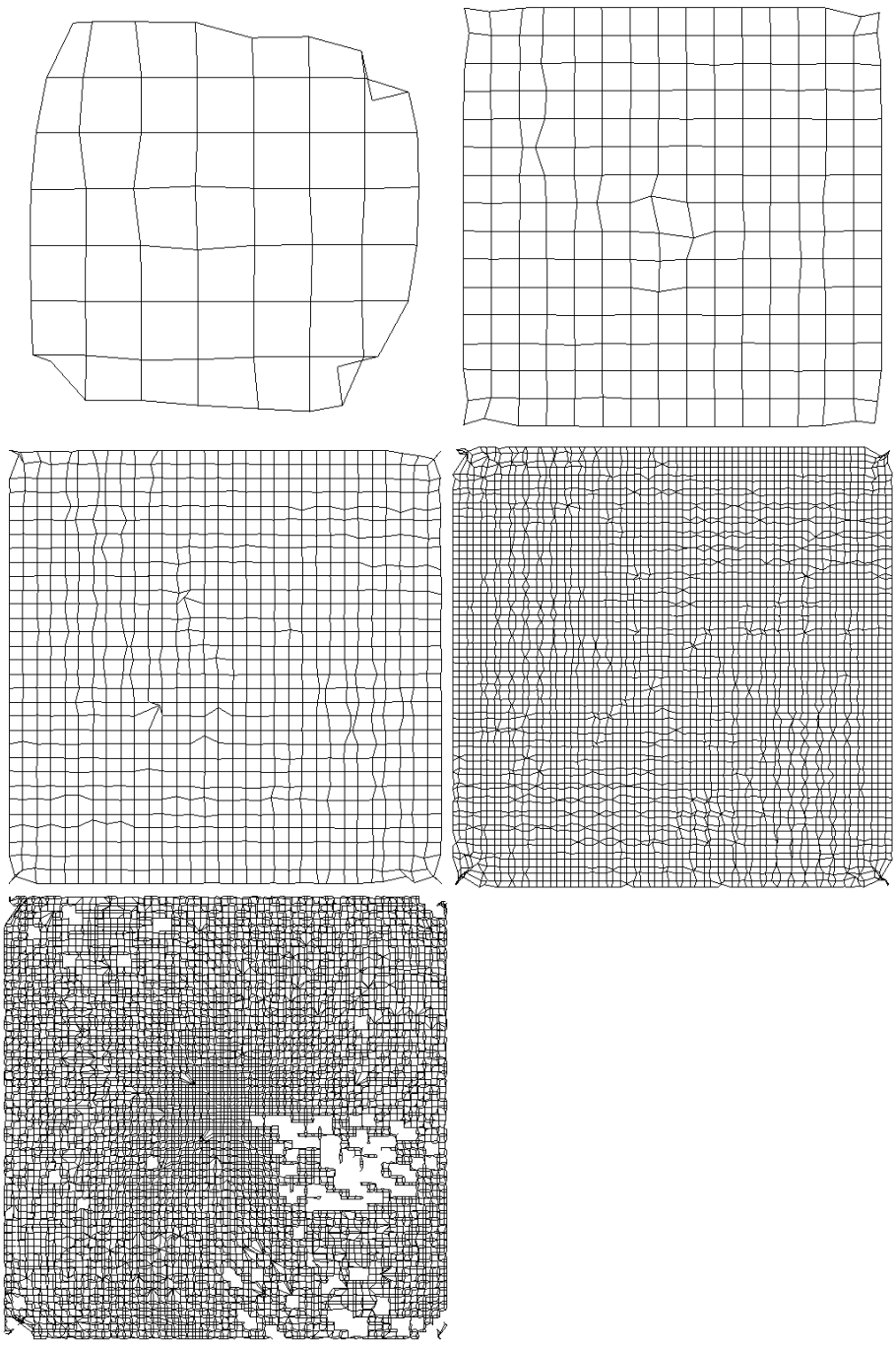

Abbildung 6.6: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^1 der dyadischen Wavelettransformation nach Mallat.

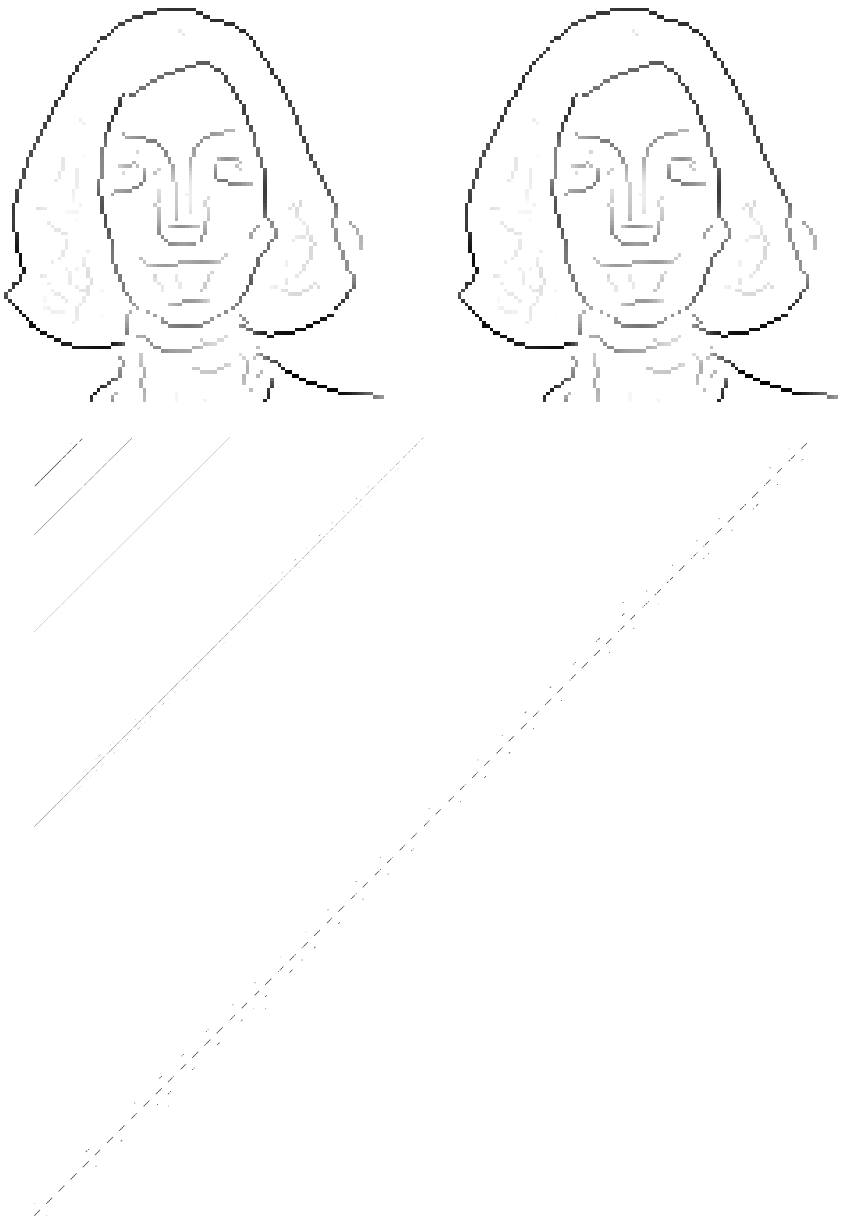

Abbildung 6.7: Projektion der Abbildung Ute0 auf Ute0, für die Skalierung 2^2 der dyadischen Wavelettransformation nach Mallat.

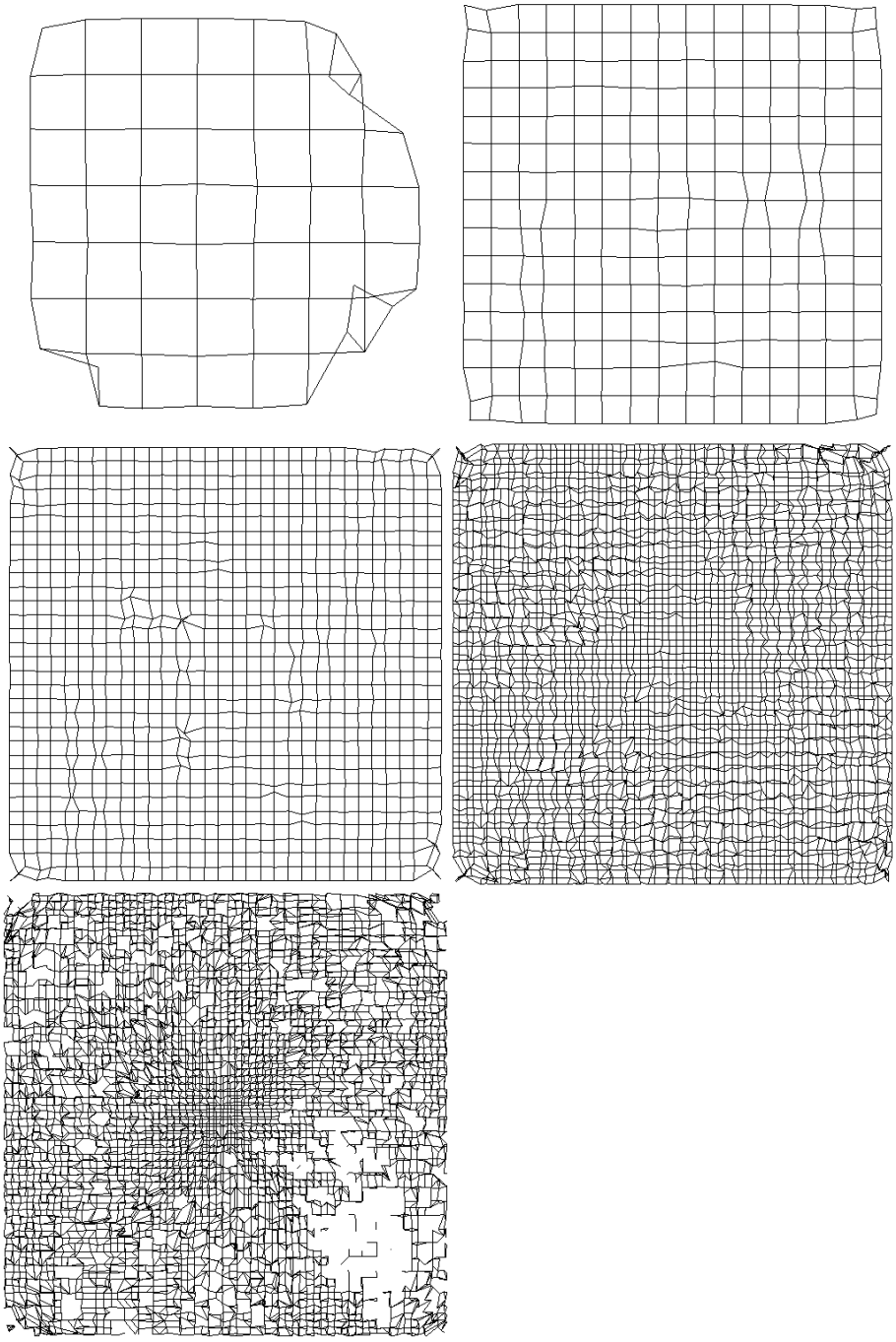

Abbildung 6.8: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^2 der dyadischen Wavelettransformation nach Mallat.

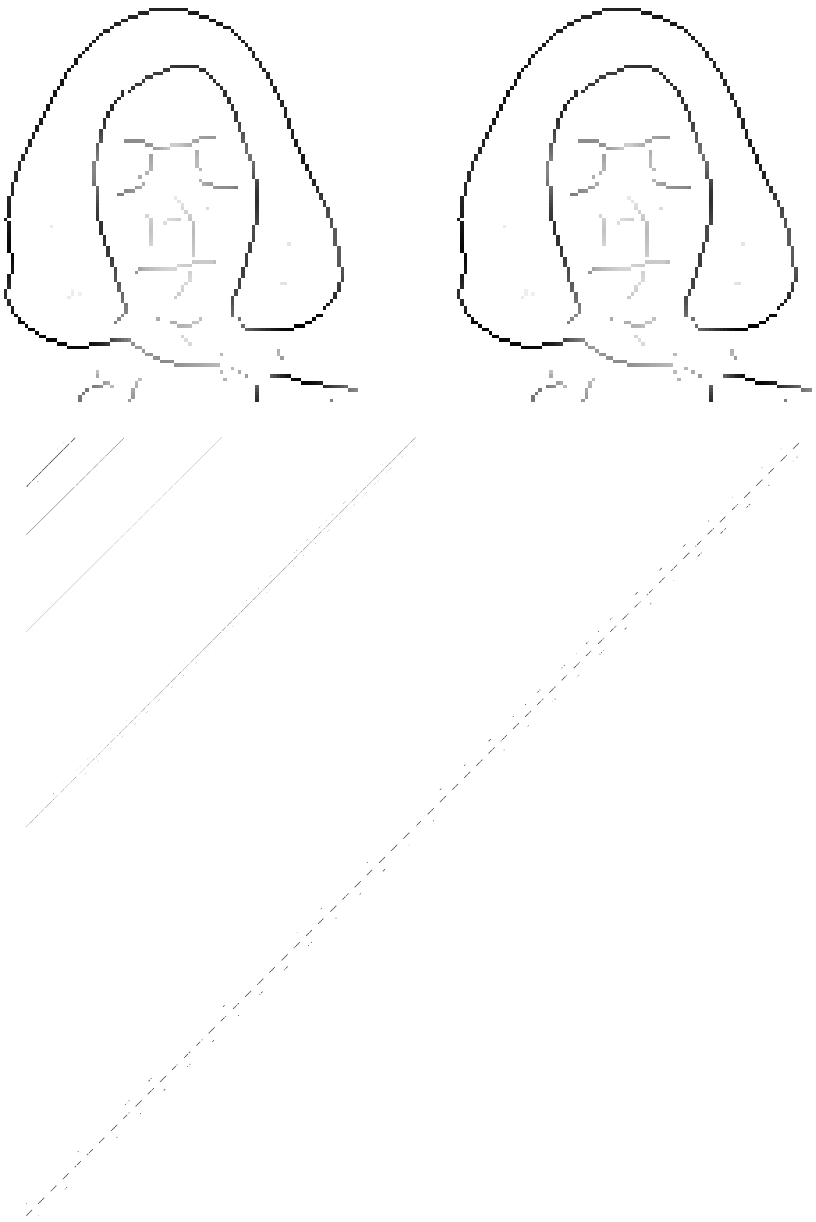

Abbildung 6.9: Projektion der Abbildung Ute0 auf Ute0, für die Skalierung 2^3 der dyadischen Wavelettransformation nach Mallat.

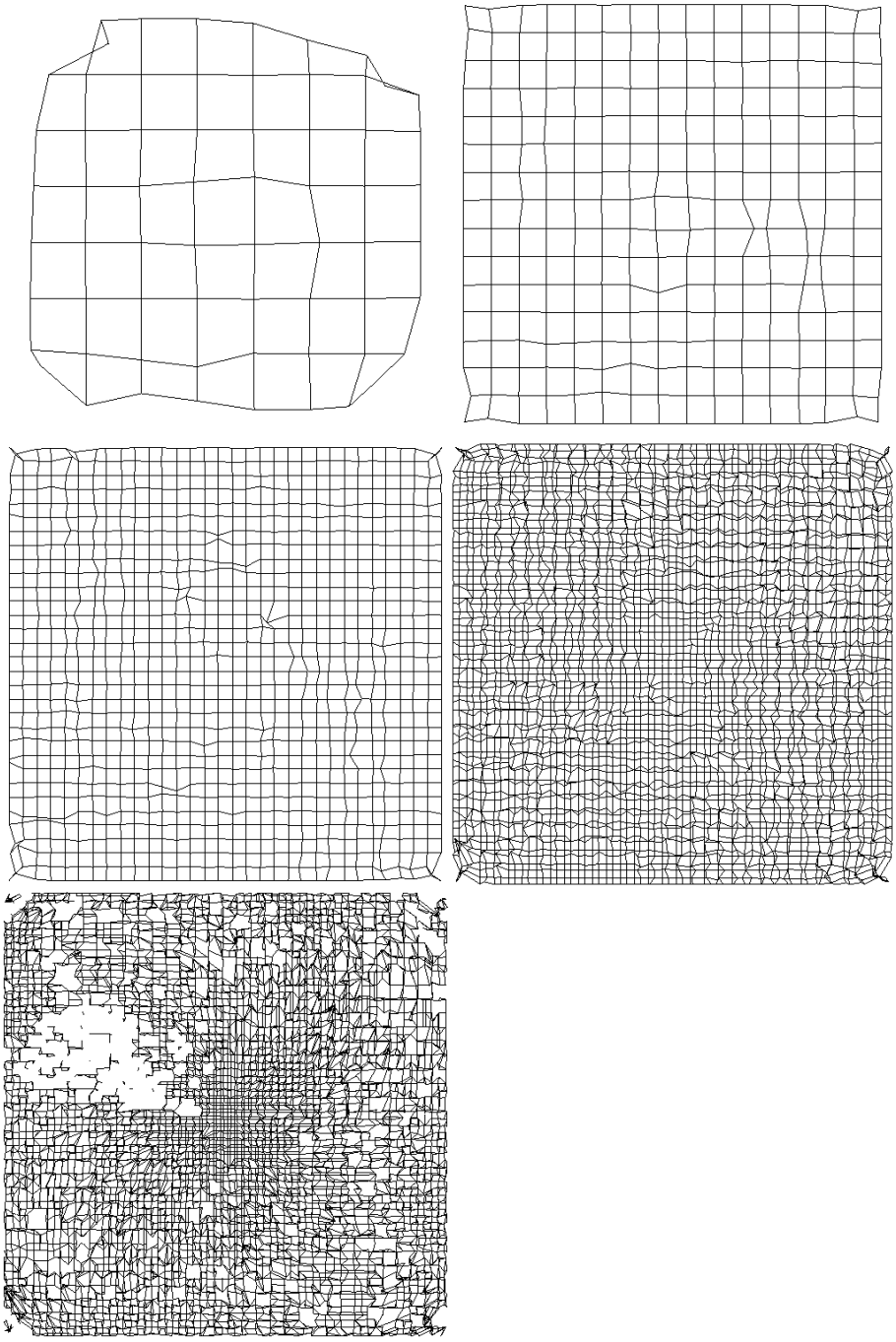

Abbildung 6.10: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^3 der dyadischen Wavelettransformation nach Mallat.

Abbildung 6.11: Projektion der Abbildung Ute0 auf Ute0, für die Skalierung 2^4 der dyadischen Wavelettransformation nach Mallat.

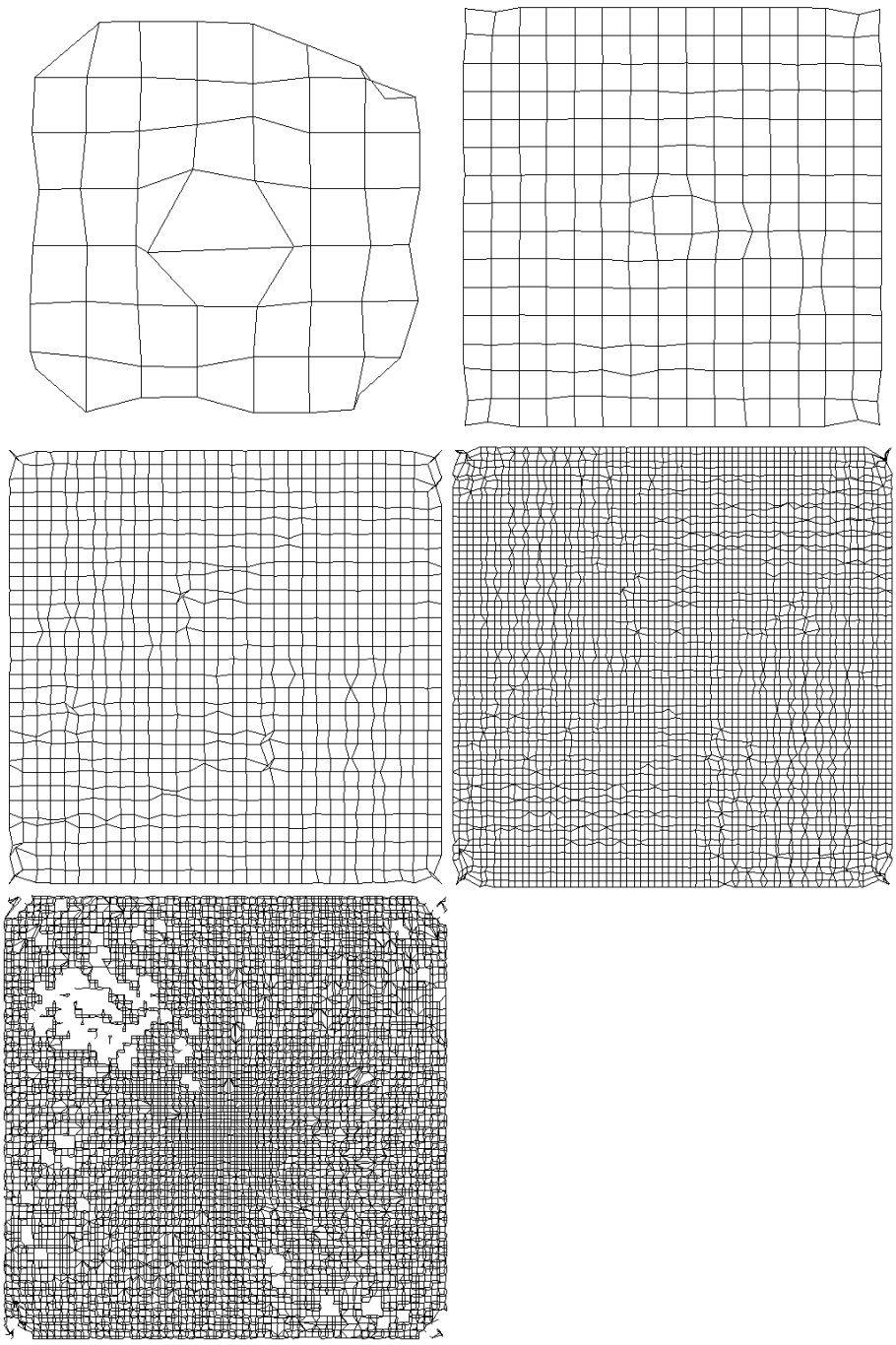

Abbildung 6.12: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^4 der dyadischen Wavelettransformation nach Mallat.

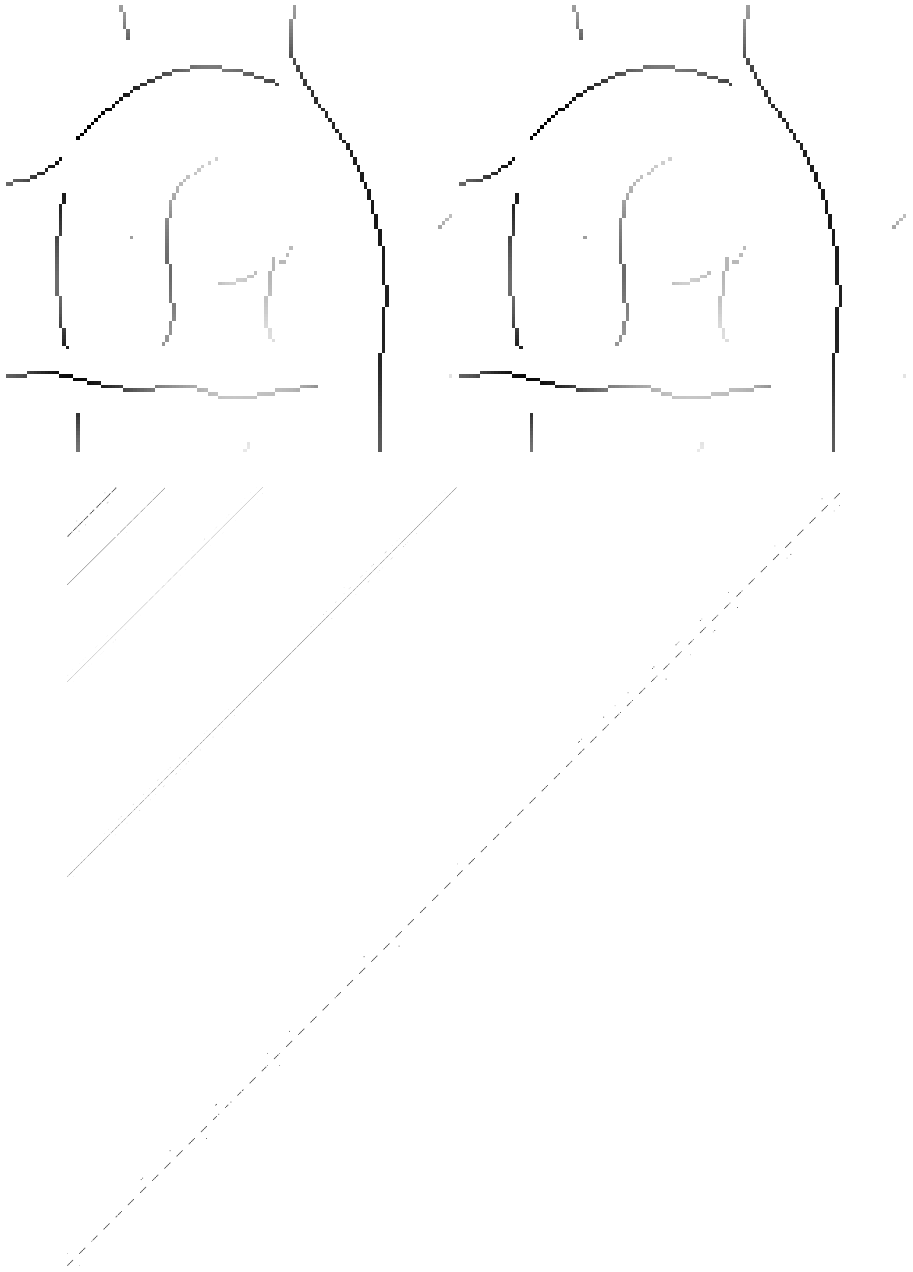

Abbildung 6.13: Projektion der Abbildung Ute0 auf Ute0, für die Skalierung 2^5 der dyadischen Wavelettransformation nach Mallat.

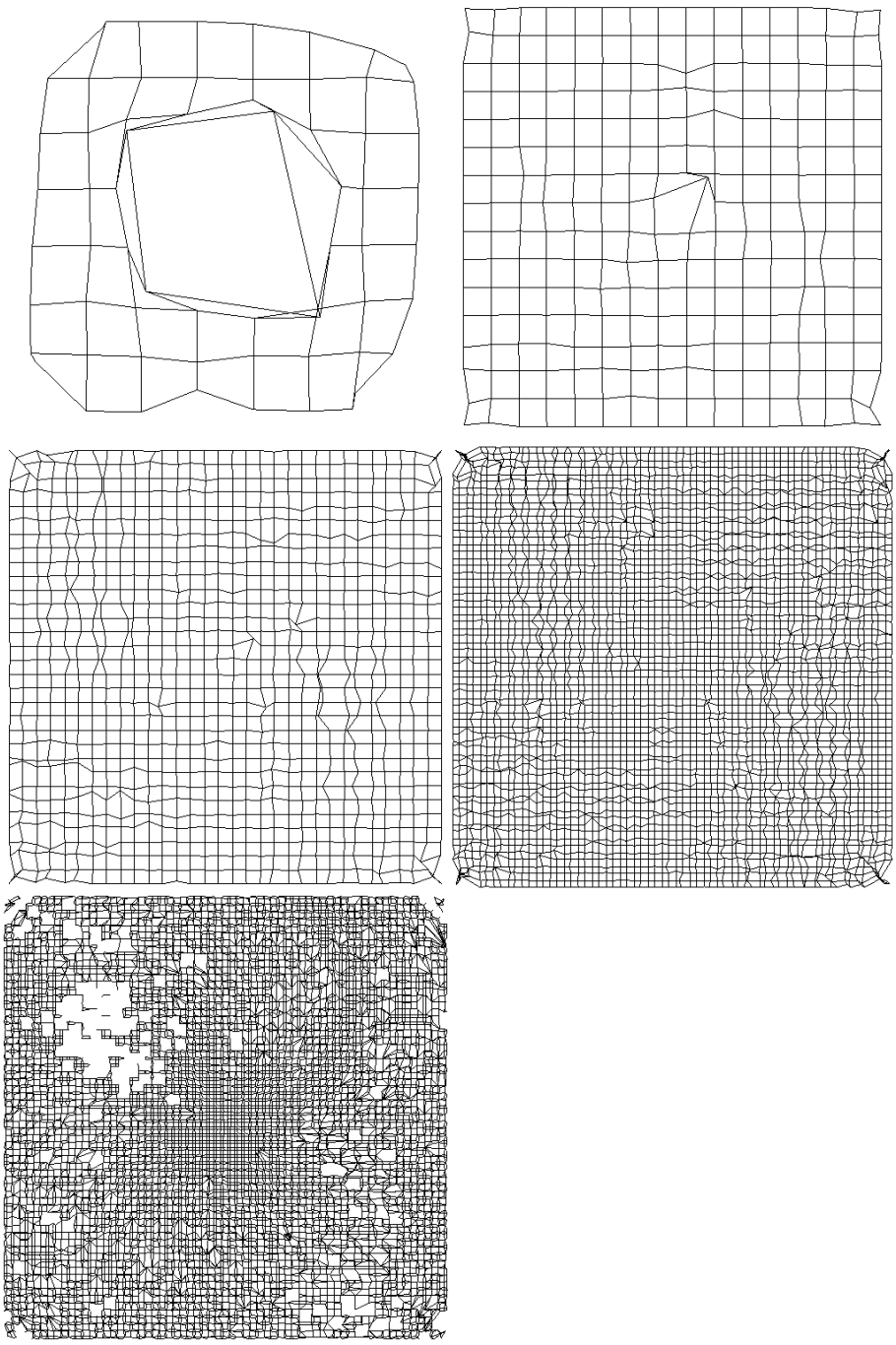

Abbildung 6.14: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^5 der dyadischen Wavelettransformation nach Mallat.

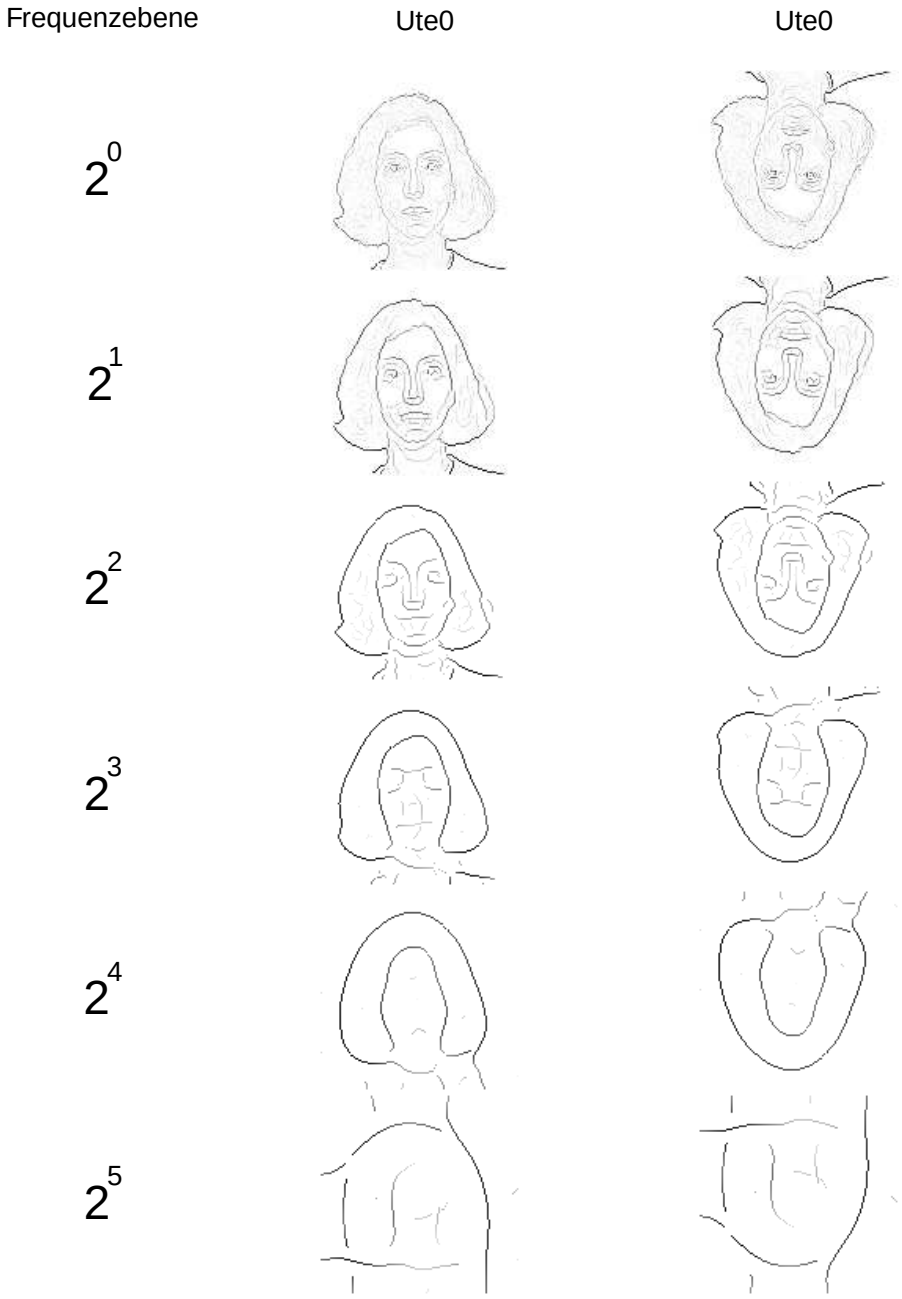

Abbildung 6.15: Übersicht, der Abbildungen 6.16 bis 6.21.

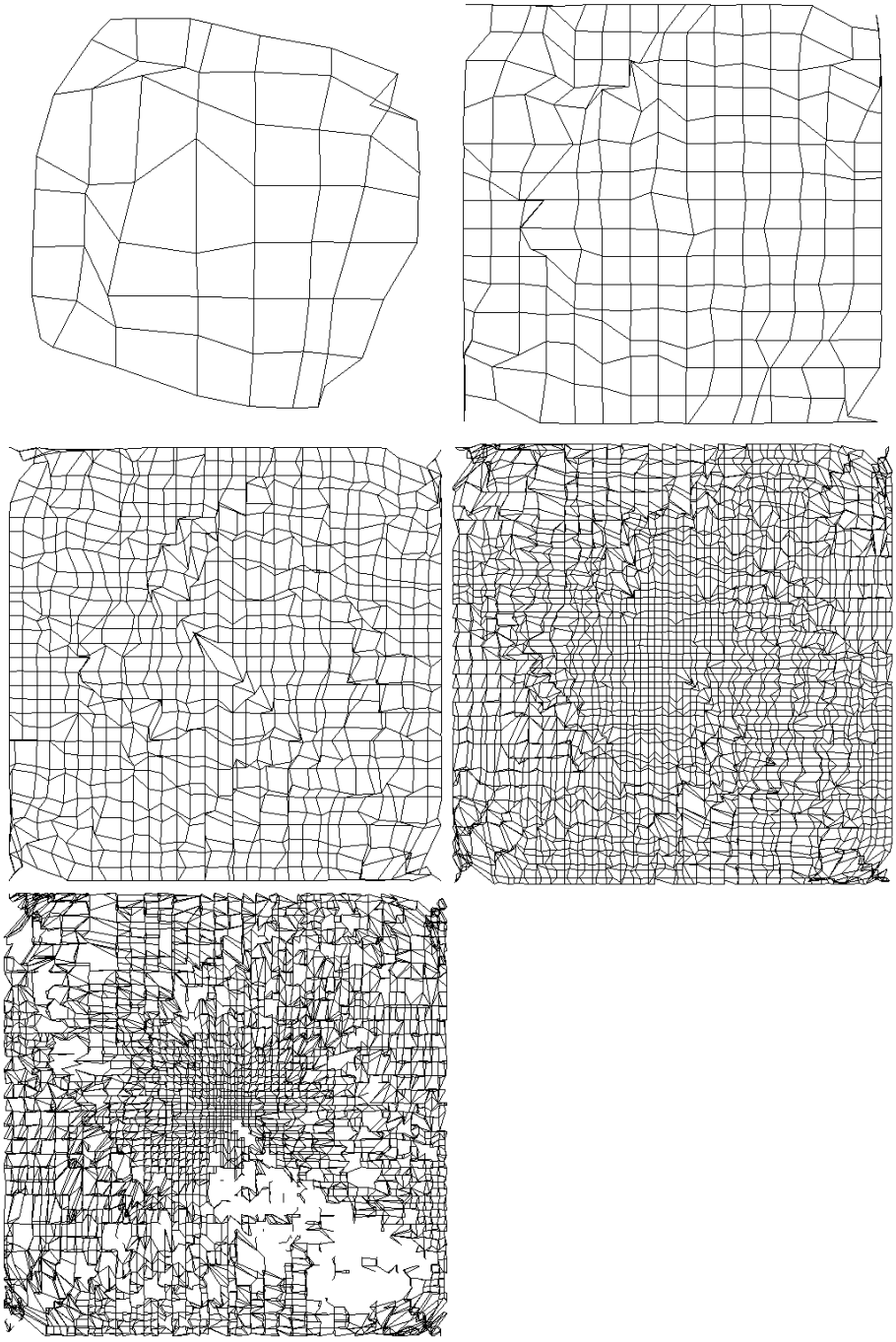

Abbildung 6.16: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^0 der dyadischen Wavelettransformation nach Mallat.

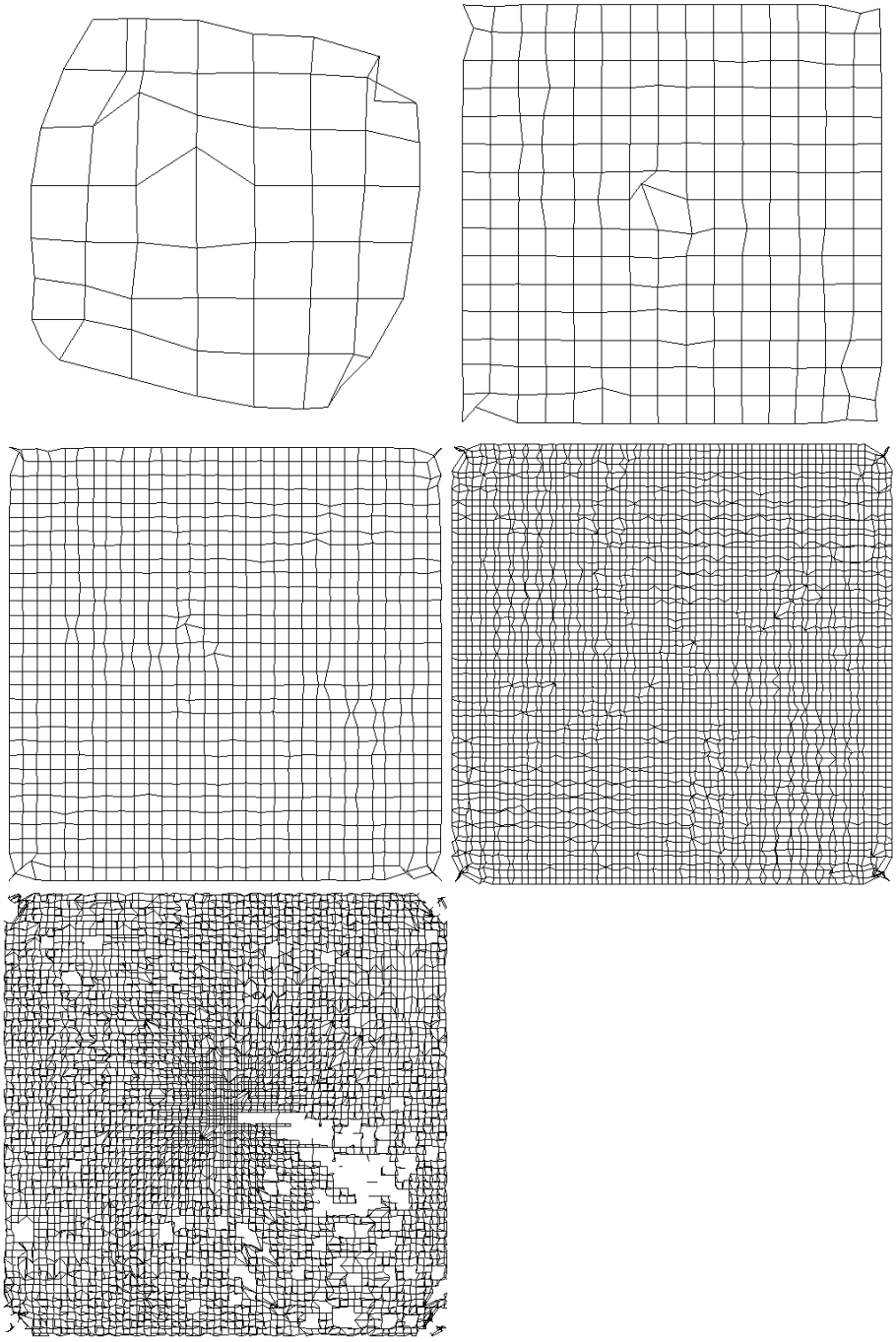

Abbildung 6.17: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^1 der dyadischen Wavelettransformation nach Mallat.

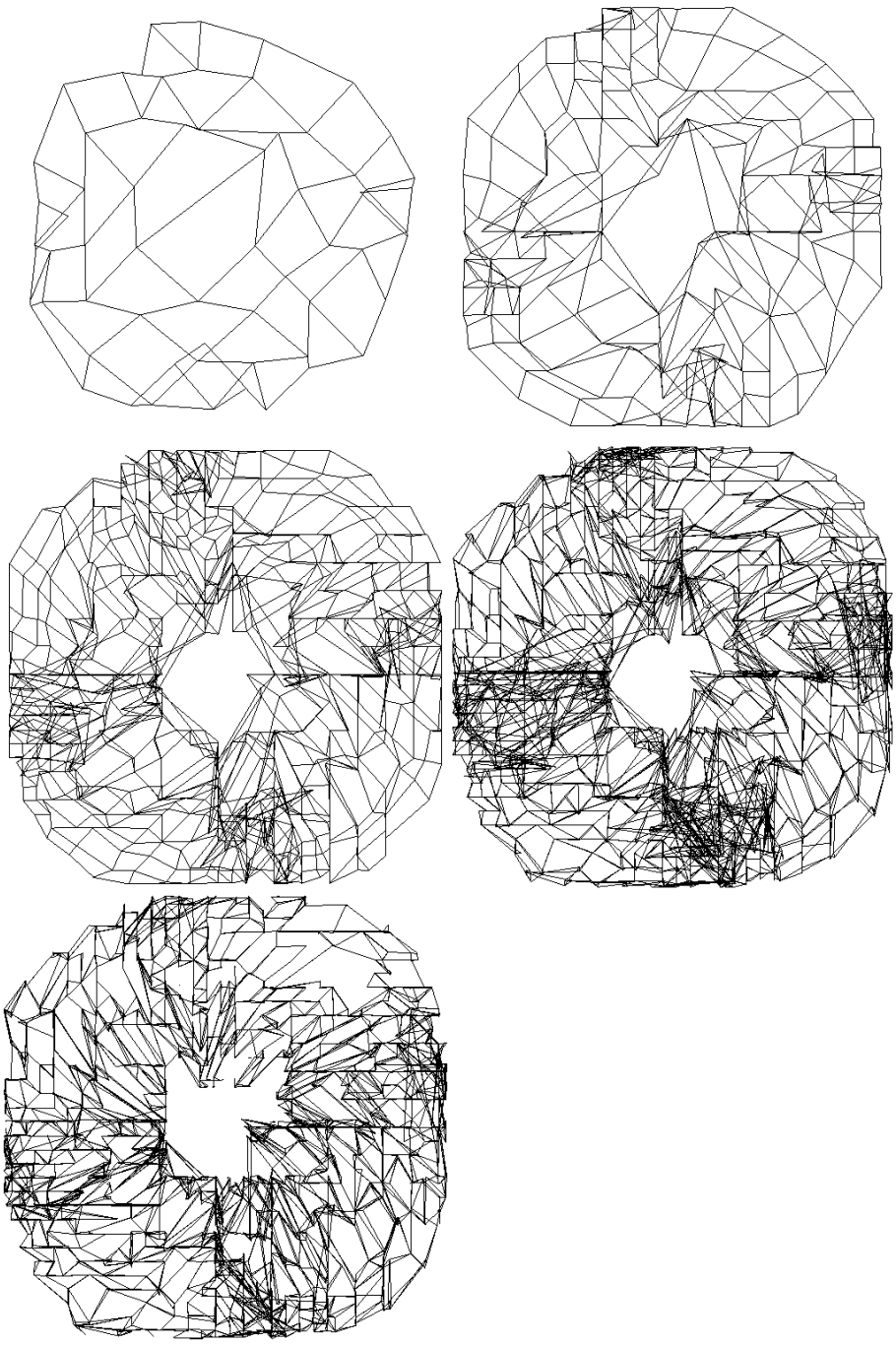

Abbildung 6.18: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^2 der dyadischen Wavelettransformation nach Mallat.

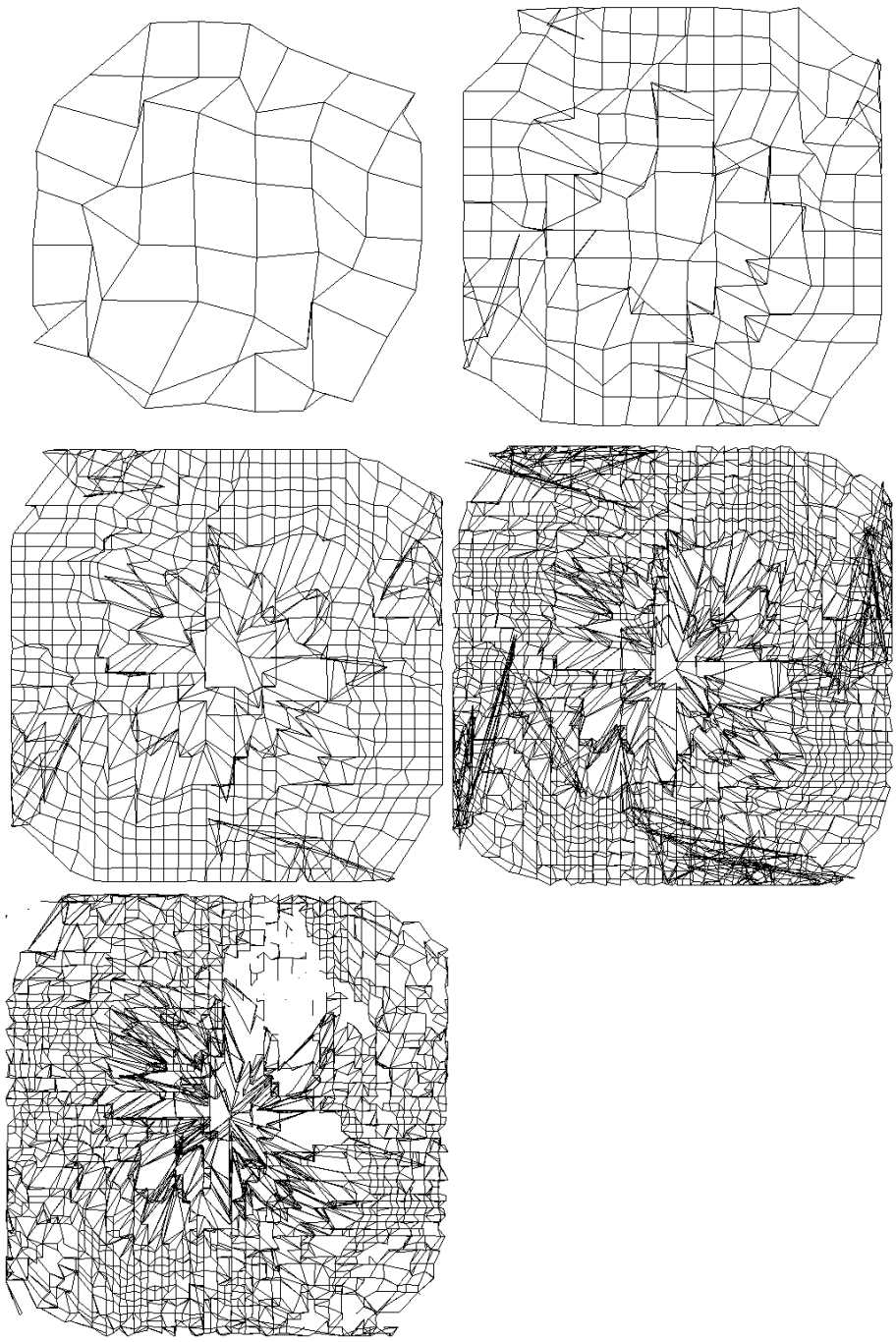

Abbildung 6.19: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^3 der dyadischen Wavelettransformation nach Mallat.

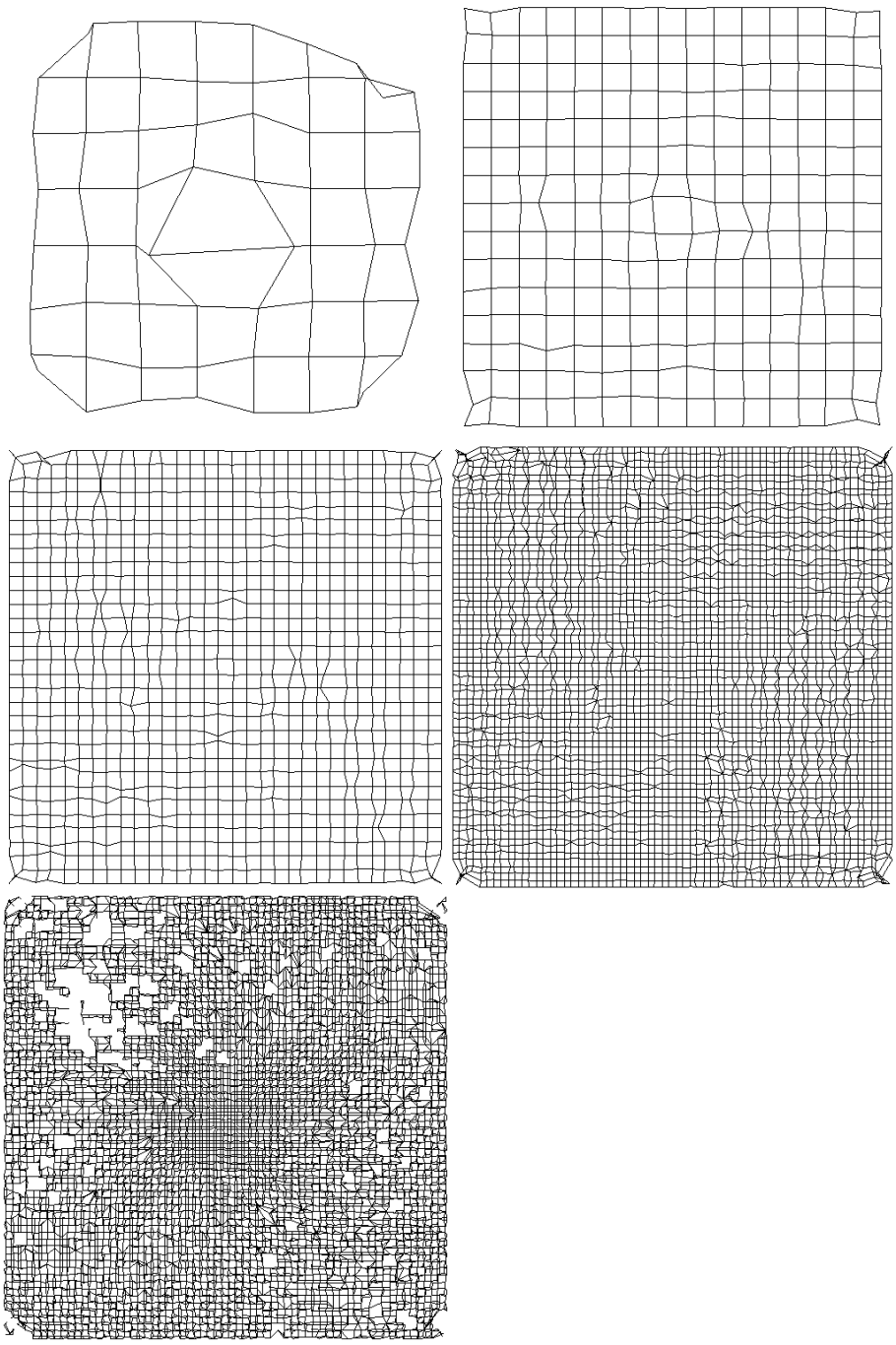

Abbildung 6.20: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^4 der dyadischen Wavelettransformation nach Mallat.

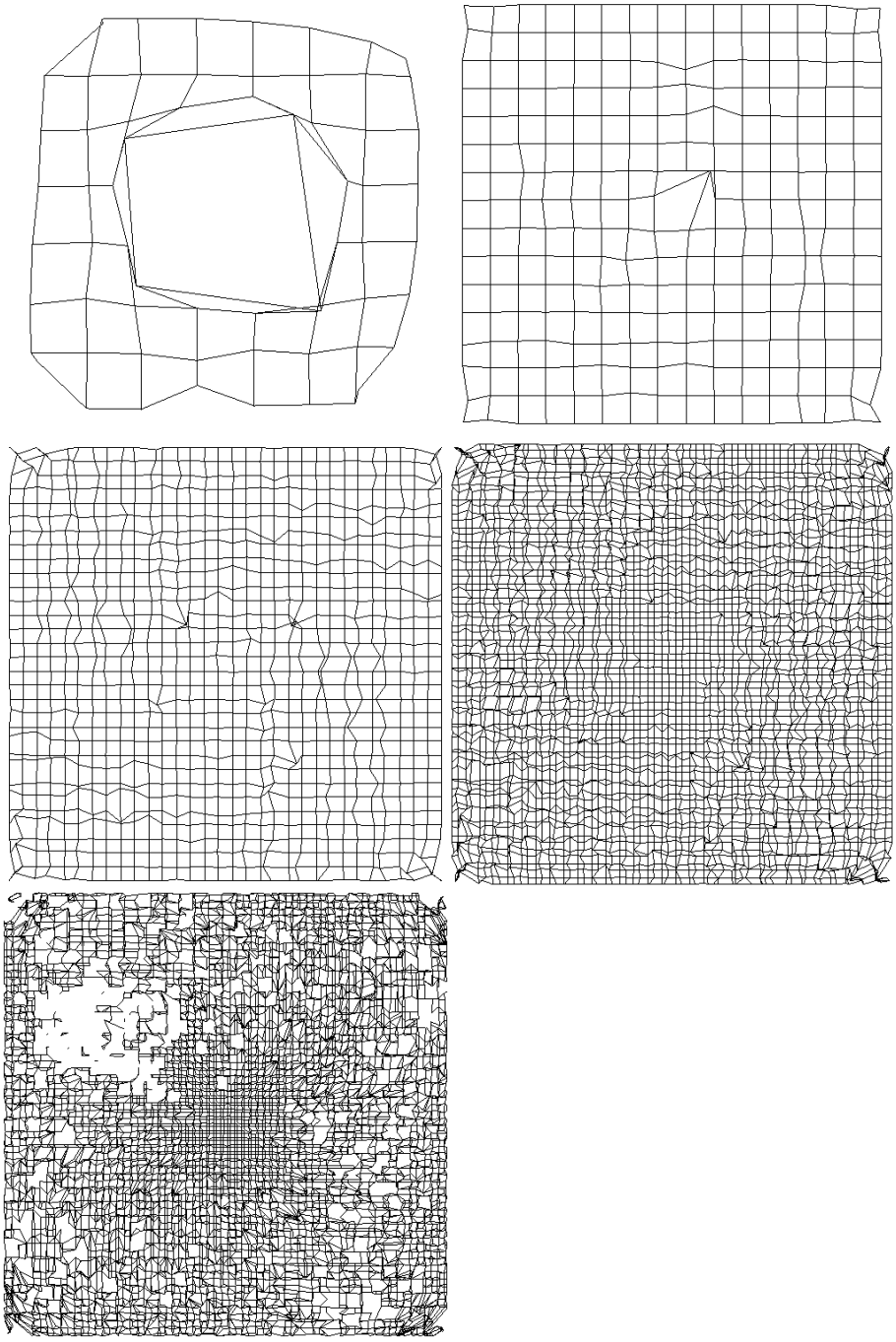

Abbildung 6.21: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^5 der dyadischen Wavelettransformation nach Mallat.

In allen Darstellungen der Verbindungsmatrizes für eine Auflösung von 128 × 128 Bildpunkten ist deutlich zu sehen, daß Löcher entstanden sind. Offensichtlich wurde beim letzten Verfeinerungsschritt zu radikal verfahren, obwohl stets nach dem gleichen Schema vorgegangen wurde.

Die Abbildungen für die vertikale Spiegelung zeigen überwiegend gute Ergebnisse, lediglich für die Skalen zwei und drei der Mallatschen Wavelettransformation ergeben sich schlechtere Abbildungen. Die Aufgabe kann aufgrund dieser Ergebnisse als prinzipiell gelöst betrachtet werden, im Kapitel 8 wird eine Möglichkeit aufgezeigt, wie das Ergebnis noch verbessert werden könnte.

Die Ergebnisse des Matchings zwischen zwei Ansichten eines Gesichtes können nicht zufrieden stellen. Lediglich das Ergebniss der tiefsten Frequenzstufe kann als gut bezeichnet werden. Betrachtet man die Abbildungen 6.23 bis 6.27 genauer, so fällt auf, daß die Verbindungsmatrizes bis zu einer Größe von 16 × 16 Bildpunkten noch ungefähr die Struktur eines regelmäßigen Gitters erkennen lassen. Die starken Verschiebungen auf den größeren Feinheitsgraden werden somit von den vielen kleinen Kanten herrühen, die den Matchingprozeß erschweren, da es sehr viele ähnliche Merkmale gibt. Eventuell könnte auch hier die im Kapitel 8 vorgestellte Verbesserung abhilfe schaffen. Andernfalls könnte eine leichte Änderung der Dynamik für bessere Ergebnisse sorgen. An der Abbildung 6.21 fällt besonders gut auf, daß die hier verwendete Form der Blobdynamik auf den groben Auflösungen, bei unglücklicher Merkmalsverteilung eine Unterrepräsentation der Bildmitte zur Folge haben kann, die in diesem Beispiel auf den feinauflösenden Ebenen wieder korrigiert wird. Auch hier könnte eine leichte Änderung der Blob-Dynamik eine Verbesserung der Ergebnisse bewirken.

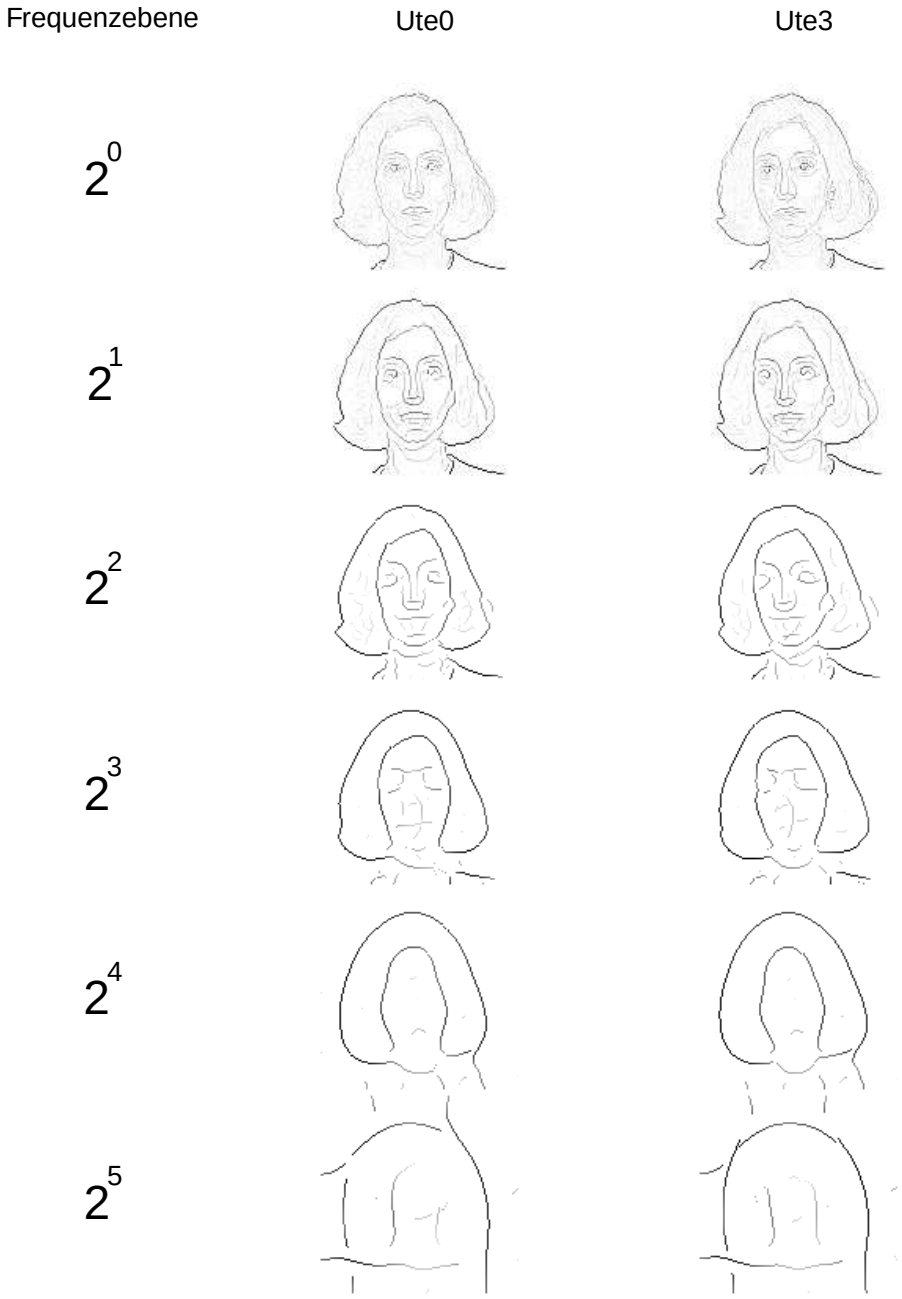

Abbildung 6.22: Übersicht, der Abbildungen 6.23 bis 6.28.

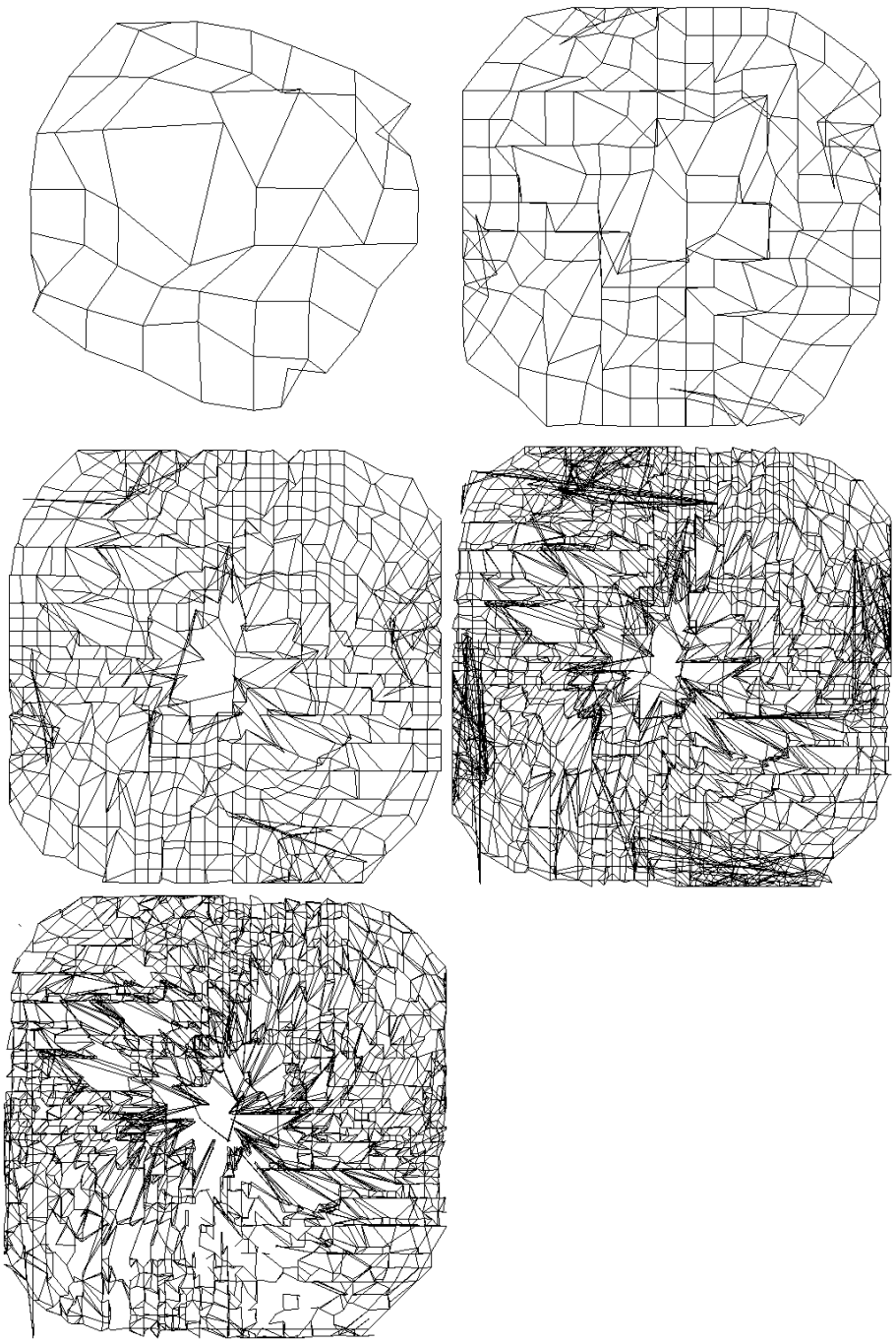

Abbildung 6.23: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^0 der dyadischen Wavelettransformation nach Mallat.

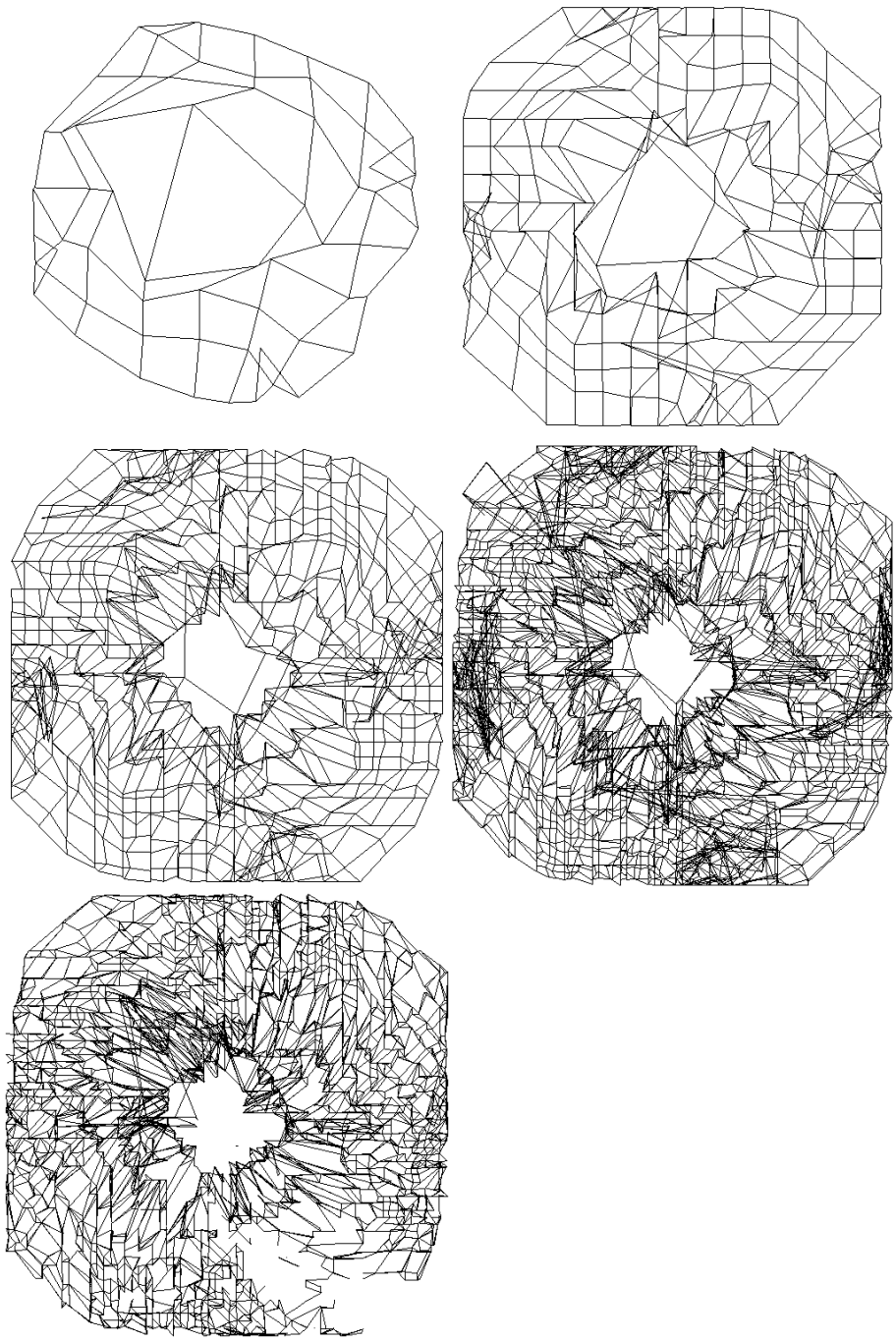

Abbildung 6.24: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^1 der dyadischen Wavelettransformation nach Mallat.

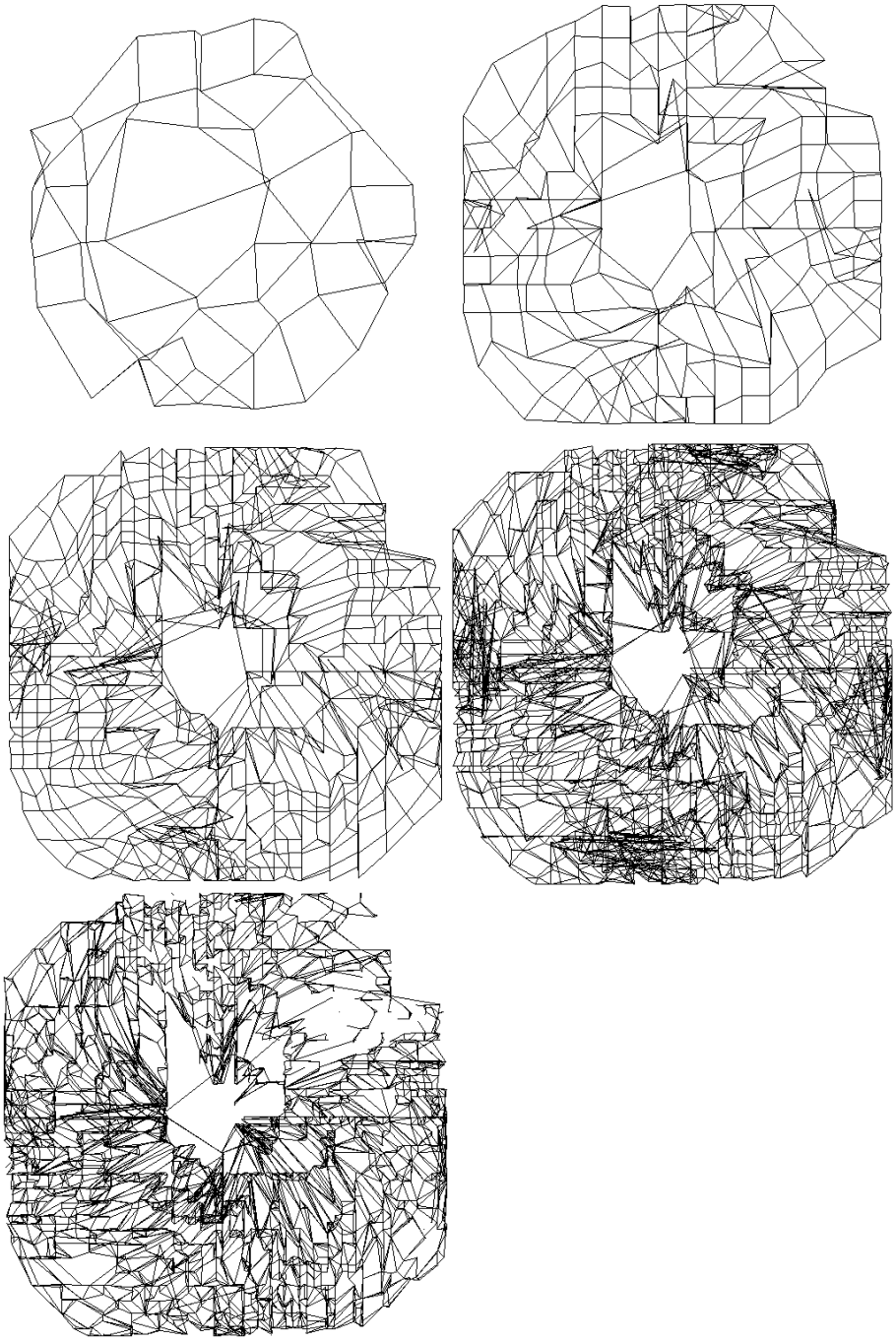

Abbildung 6.25: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^2 der dyadischen Wavelettransformation nach Mallat.

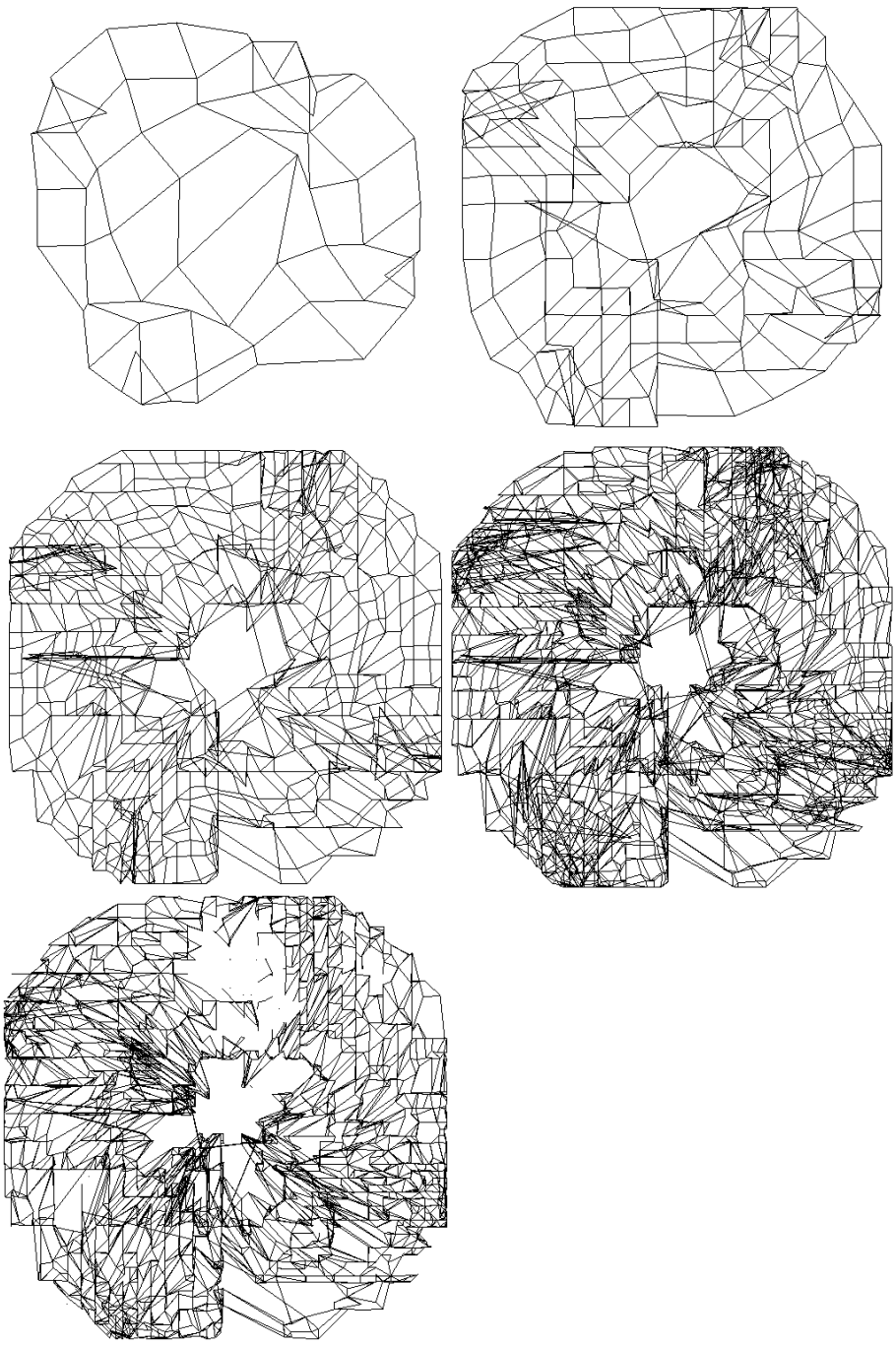

Abbildung 6.26: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^3 der dyadischen Wavelettransformation nach Mallat.

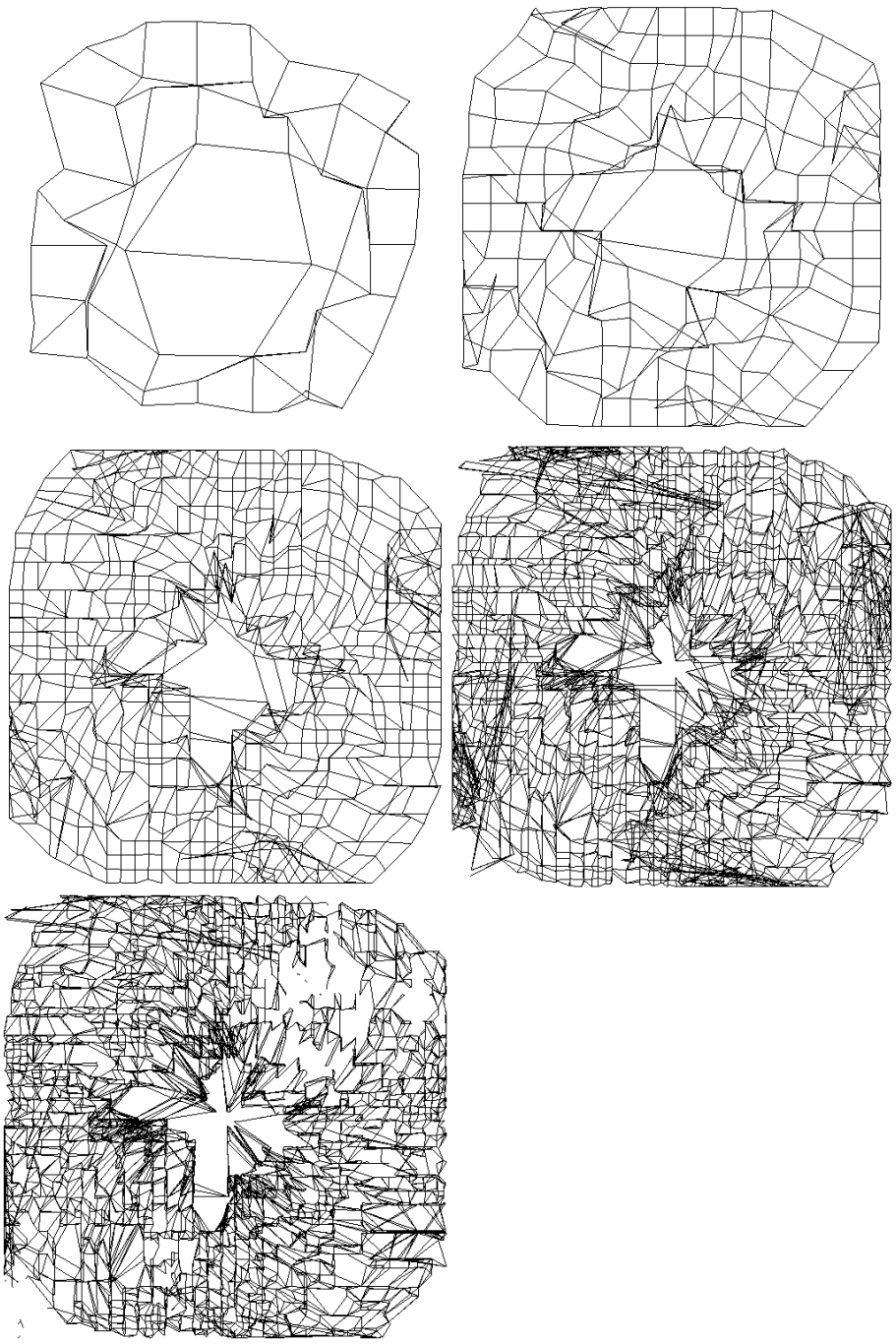

Abbildung 6.27: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^4 der dyadischen Wavelettransformation nach Mallat.

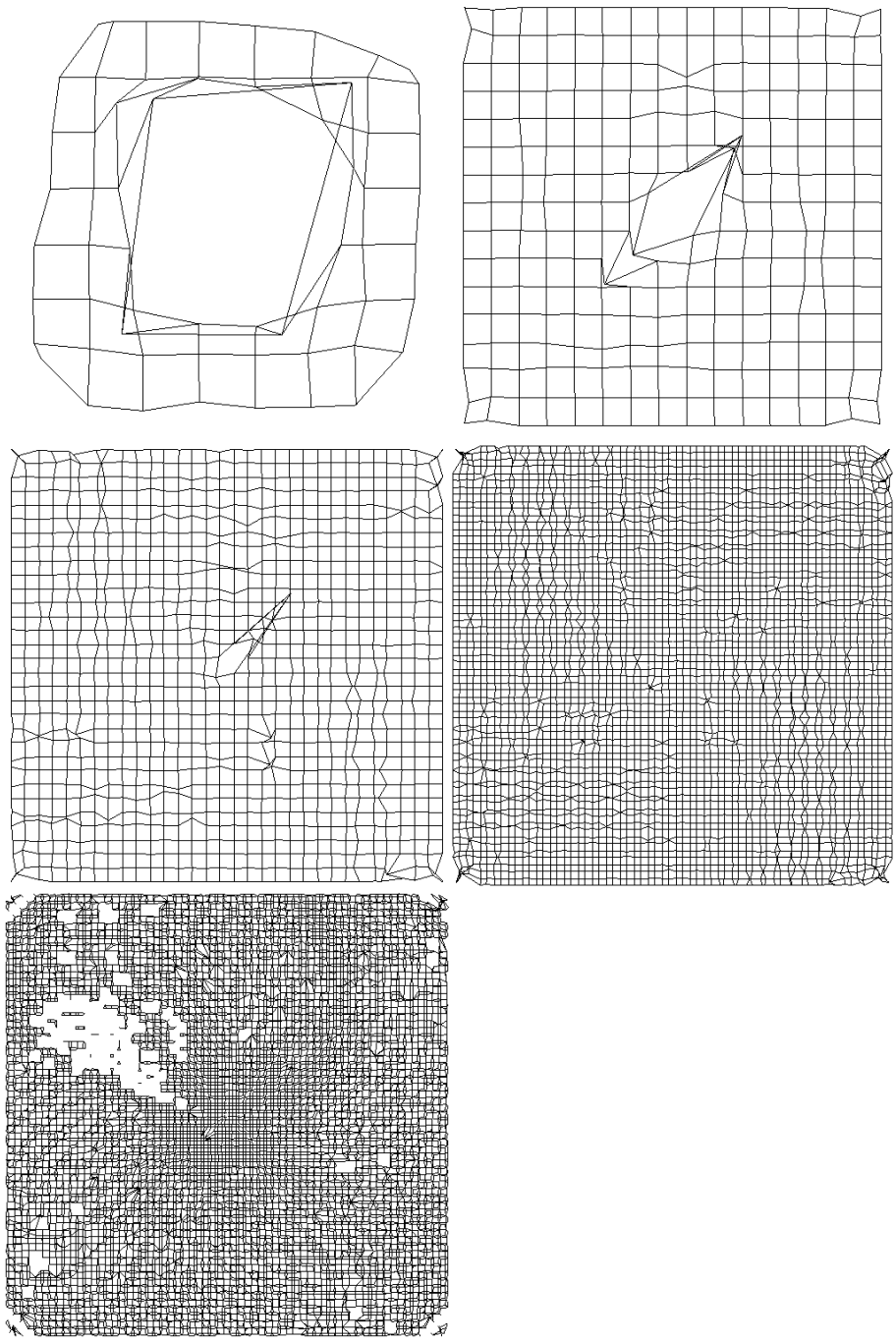

Abbildung 6.28: Die Karten der Abbildung Ute0 auf Ute0, für die Skalierung 2^5 der dyadischen Wavelettransformation nach Mallat.

Kapitel 7

Zusammenfassung

Wie die Ergebnisse im Kapitel 6 zeigen, ist es möglich die volle Abbildung schrittweise durch die Verfeinerung einer groben Näherung zu gewinnen. In dieser Arbeit wurde das Verfeinerungsverfahren für das Matching von Kantenbildern eingesetzt, wo es zufriedenstellende Resultate liefert. Das Prinzip kann ebenfalls für andere Matchingaufgaben benutzt werden. So ist es beispielsweise denkbar, daß aus einer bereits gewonnenen Abbildung zwischen zwei Profilbildern ein hochauflösender Matching-Prozeß zum detailierten Vergleich der Ohren, welche sehr signifikant sind, gestartet wird und der auf die bereits vorhandene globale Information aufbaut.

Darüberhinaus wurde gezeigt, daß es möglich ist, auf eine Ähnlichkeitsfunktion für die Merkmale zu verzichten, wenn die Merkmale direkt in die Neuronendynamik eingebracht werden. Dies ist eine sehr interessante Alternative zur Ähnlichkeitsmatrix, da es zuweilen problematisch ist, eine gute Ähnlichkeitsfunktion für die Merkmale zu finden. Für die bisherige Form des Dynamic Link Matching war die Ähnlichkeitsfunktion von entscheidender Bedeutung. Das Auffinden einer Funktion zur Beurteilung der Merkmalsähnlichkeit ist eine schwierige Aufgabe, die für jeden Merkmalstyp neu zu lösen ist. Mit der Einbeziehung der Merkmale in die Dynamik der Neuronen kann das Dynamic Link Matching stärker auf die Merkmale ausgerichtet werden und so eine Ähnlichkeitsmatrix überflüssig machen. Dies wurde hier für das Matching der Kantenbilder gezeigt.

Kapitel 8

Ausblick

Leider war es in der für diese Arbeit zur Verfügung stehenden Zeit nicht mehr möglich, das Matching der verschiedenen Mallatebenen stärker mit einander zu verknüpfen. Die Idee besteht darin, für jede Skala der Mallattransformation eine Neuronenschicht zu verwenden. Die Dynamik der Schichten wird genauso simuliert, wie dies in der vorhandenen Implementation geschieht. Für die X- und Y-Schicht einer Mallatebene wird die Korrelation der Ausgabeänderung berechnet und in einer Matrix aufsummiert. Für alle Neuronenschichten zusammen gibt es nur eine Verbindungsmatrix, die mit den einzelnen Korrelationsmatrizen gelernt wird. Auf diese Weise müssen sich alle Skalen der Mallatschen Wavelettransformation auf eine Abbildung einigen. Es ist zu erwarten, daß die tieffrequenten Ebenen der Mallattransformation für die globale Ausrichtung sorgen und die höherfrequenten eine lokale Feinabstimmung bewirken.

Eine Parallelisierung des Verfahrens ist in der gegenwärtigen Implementation sehr leicht möglich, indem die einzelnen Matching-Prozesse für die verschiedenen Auflösungsskalen der Mallatschen Wavelettransformation auf verschiedenen Prozessoren gerechnet werden. Mit der oben geschilderten Idee würde jeder Rechenknoten eine komplette Blob-Dynamik simulieren. Für die Berechnung der Korrelationen wäre bereits mehr Kommunikation zwischen den korrespondierenden Knoten notwendig, es könnte daher besser sein, auf den groben Ebenen je ein korrespondierendes Schichtpaar einem Prozessor zuzuordnen. Prinzipiell ist es auch vorstellbar, das für jedes Neuron ein relativ einfacher Prozessor eingesetzt wird und die Dynamik einer Schicht parallel auf einer SIMD-Architektur berechnet wird, dabei steht SIMD für *single instruction, multiple data* und bedeutet, daß die Prozessoren alle den gleichen Befehl ausführen, aber verschiedene Daten bearbeiten. Die Blob-Dynamik eingnet sich sehr gut für eine derartige Umsetzung, dabei wird der Interaktionskern wieder durch die kurzreichweitigen lateralen Verbindungen ersetzt, deren Einfluß auf das System gerade durch den Interaktionskern beschrieben wird.

Anhang A

Darstellung vierdimensionaler Matrizen

Hier werden die beiden Darstellungsformen für die vierdimensionalen Matrizen erklärt. Die Matrizen entstehen bei der Generierung einer Abbildung zwischen zweidimensionalen Bildern.

A.1 Zweidimensionale Projektion

Eine einfache Möglichkeit, die vierdimensionalen Matrizen auf zwei Dimensionen zu reduzieren, besteht darin, daß die Bilder jeweils auf eindimensionale Ketten projiziert werden. Hierzu werden die einzelnen Zeilen der Bilder hintereinander gehängt. Hat ein Bildpunkt eines $n \times m$ Bildes den zweidimensionalen Index (x, y), so geht dieser in den eindimensionalen Index $yn + x$ über. Beschreibt ein Element einer vierdimensionalen Matrix die Verbindung zwischen den Bildpunkten (x_1, y_1) und (x_2, y_2), so erhält es in der zweidimensionalen Projektion den Index $(y_1 n_1 + x_1, y_2 n_2 + x_2)$. Die umgekehrte Berechnung, der Bildkoordinaten aus dem Index (p, q) der zweidimensionalen Projektion, läßt sich mittels der Funktionen div (ganzahlige Division) und mod (Rest der ganzahligen Division) beschreiben:

$$\begin{aligned} x_1 &= p \mod n_1 \\ y_1 &= p \operatorname{div} n_1 \\ x_2 &= q \mod n_2 \\ y_2 &= q \operatorname{div} n_2 \end{aligned}$$

Die Methode läßt sich zwar sehr leicht umsetzen, das „Lesen“ dieser zweidimensionalen Projektionen ist allerdings eine Kunst für sich. Für die identische Abbildung ist das Ergebnis der Projektion die Hauptdiagonale. Die Abbildungen A.1 bis A.3 zeigen etwas kompliziertere Beispiele.

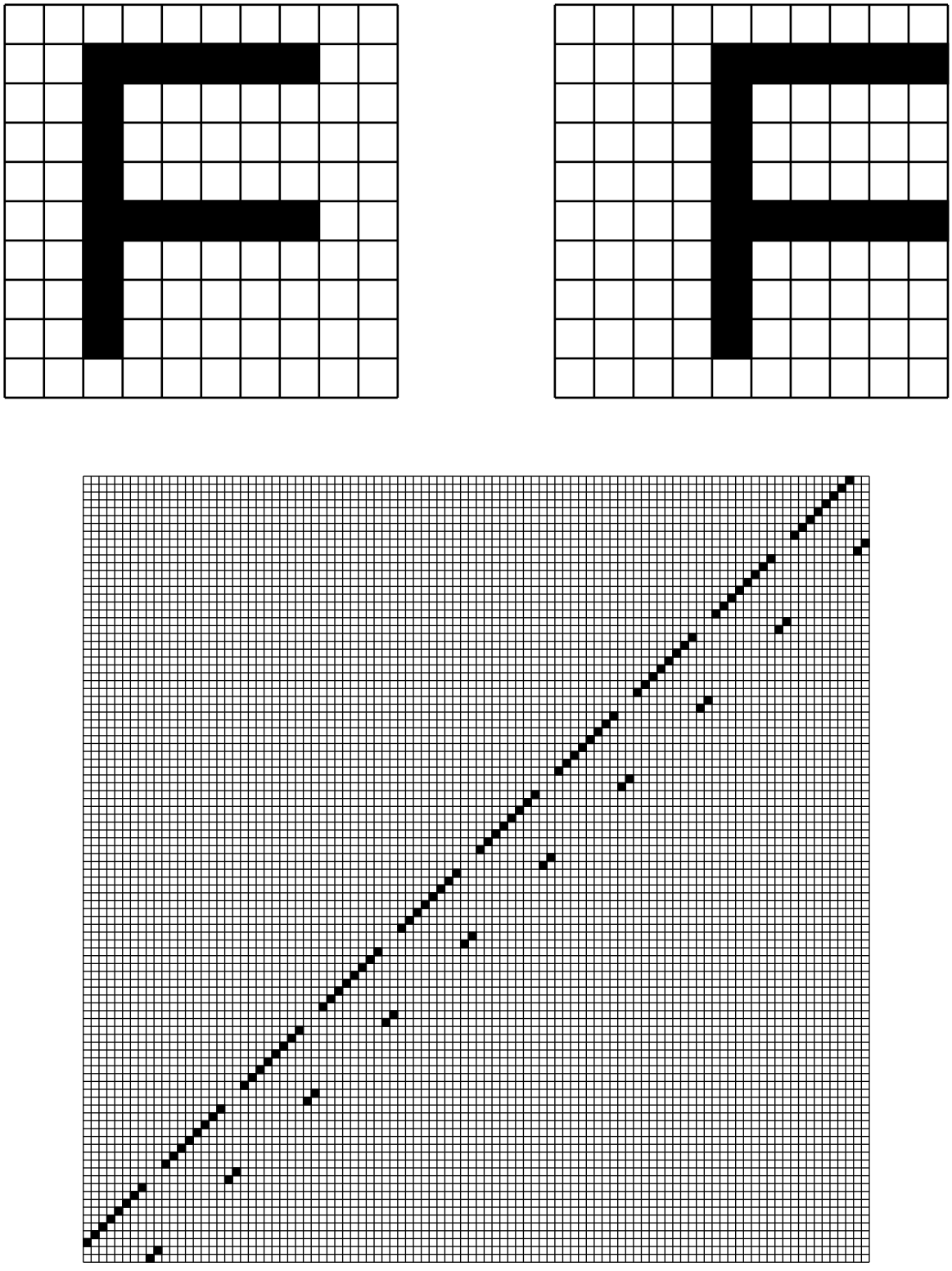

Abbildung A.1: Zweidimensionale Projektion einer Translation.

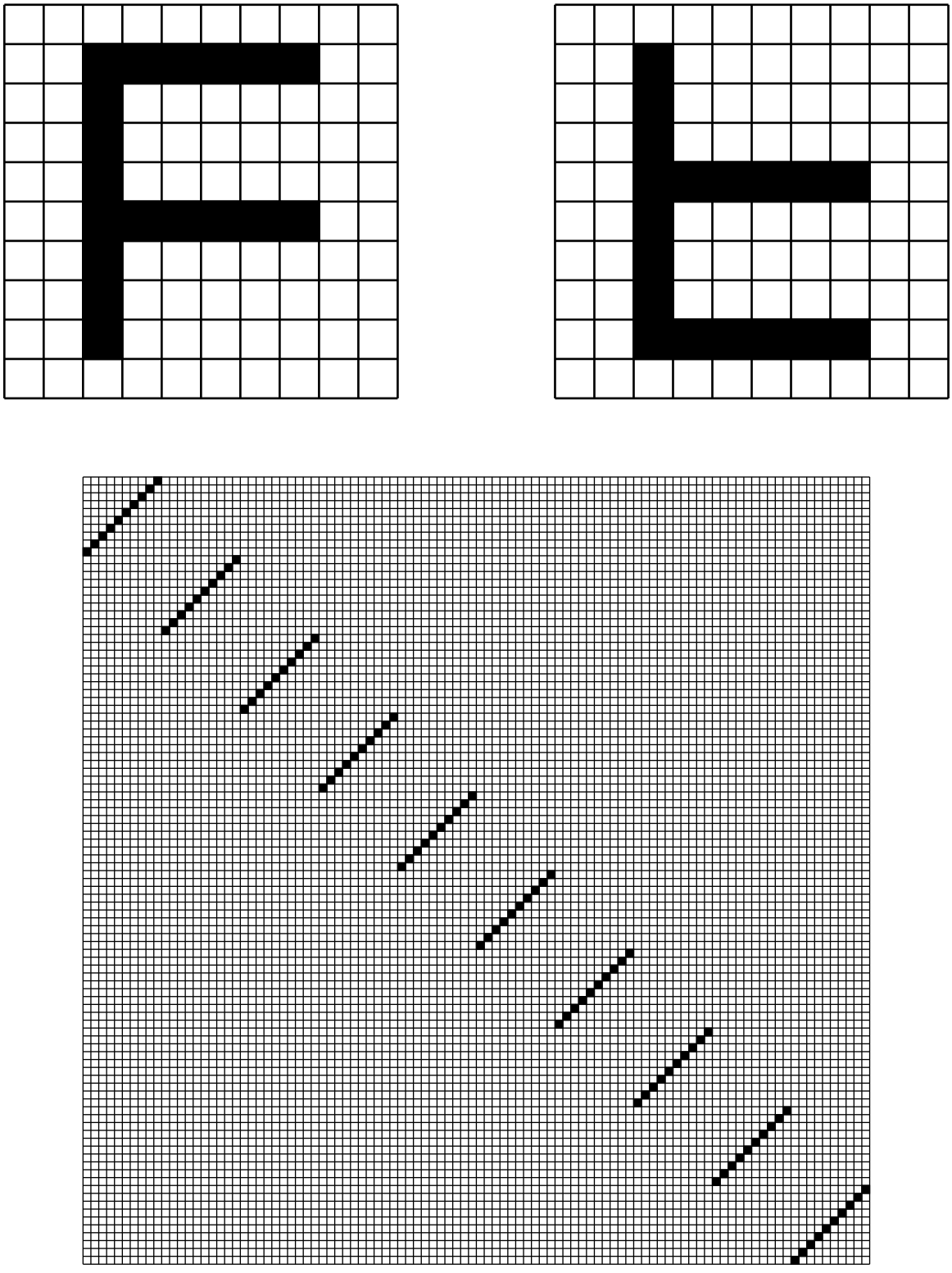

Abbildung A.2: Zweidimensionale Projektion einer Spiegelung.

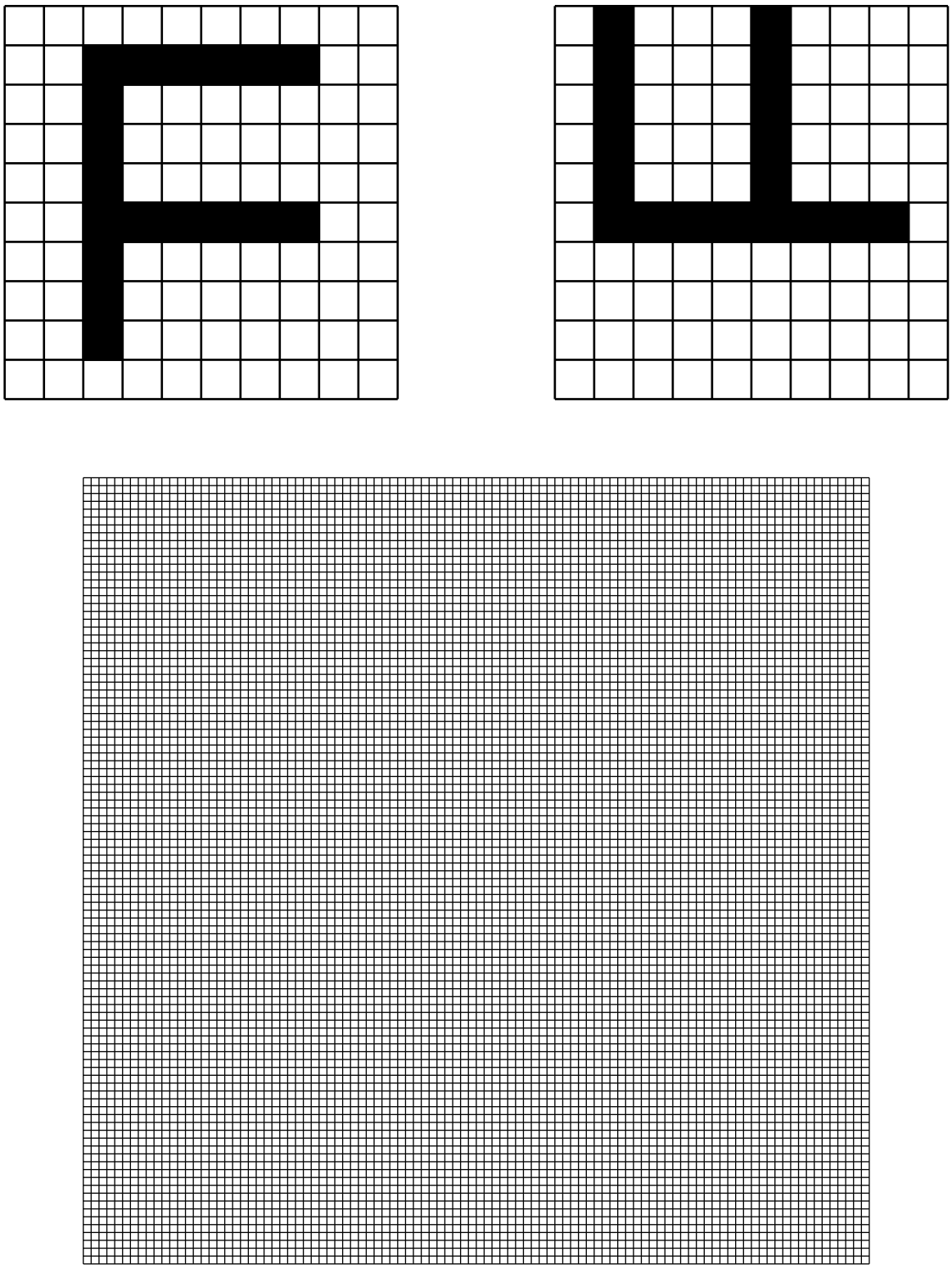

Abbildung A.3: Zweidimensionale Projektion der Überlagerung einer Rotation und Translation.

A.2 Darstellung von Abbildungen durch Karten

Eine Karte läßt sich gewinnen, indem für jedes Neuron einer Schicht der Schwerpunkt seiner Verbindungen zur anderen Schicht berechnet wird. Der Schwerpunkt läßt sich, analog zur Berechnung des physikalischen Schwerpunktes, nach folgender Formel berechnen:

$$g_a = \frac{\sum_b b \cdot J_{ab}}{\sum_b J_{ab}} \quad . \tag{A.1}$$

Dabei wird der Zielort jeder von dem Neuron ausgehenden Verbindung mit ihrer Stärke multipliziert, wir erhalten also einen gewichteten Zielvektor. Anschließend wird der Mittelwert dieser gewichteten Vektor für jedes Neuron berechnet.

Diese Berechnung ist zwar etwas aufwendiger und vielleicht zunächst nicht ganz so intuitiv, wie die obige Projektion, dafür lassen sich die Ergebnisse leicht verstehen und besser beurteilen. Die identische Abbildung ergibt hier ein Gitter mit einheitlichen Abständen.

Anhang B

Wavelets

In diesem Kapitel soll eine kurze Einleitung in die Theorie der Wavelets und insbesondere in die dyadische Wavelettransformation nach Mallat gegeben werden, für eine umfassendere Darstellung sei der Leser auf [Chu92a] und [Chu92b] verwiesen.

B.1 Bezug von Fourier- zur Wavelettransformation

Betrachten wir zunächst den Hilbertraum $L^2(0,2\pi)$ aller quadratintergrablen Funktionen f, die auf dem Interval $[0,2\pi)$ definiert sind und die Bedingung

$$\int_0^{2\pi} |f(x)|^2 \, dx < \infty \tag{B.1}$$

erfüllen. Der Vektorraum $L^2(0,2\pi)$ wird oft auch als der Raum der 2π-periodischen quadratintergrablen Funktionen bezeichnet, da die Funktionen in $L^2(0,2\pi)$ periodisch auf $\mathbf{R}$ fortgesetzt werden können. Jede Funktion f in $L^2[0,2\pi)$ besitzt eine Fourierreihe

$$f(x) = \sum_{n=-\infty}^{\infty} c_\omega e^{i\omega x} \ , \tag{B.2}$$

mit

$$c_\omega = \frac{1}{2\pi} \int_0^{2\pi} f(x) e^{-i\omega x} \, dx \ . \tag{B.3}$$

Dies bedeutet, f wird zerlegt in unendlich viele, paarweise orthogonale Komponenten $g_\omega(x) := c_\omega e^{i\omega x}$. Dabei bezieht sich orthogonal auf das Skalarprodukt

$$\langle g_{\omega_1}, g_{\omega_2} \rangle^* := \frac{1}{2\pi} \int_0^{2\pi} g_{\omega_1}(x) \overline{g_{\omega_2}(x)} \, dx \ , \tag{B.4}$$

wobei $\overline{x}$ für die zu x konjugiert komplexe Zahl steht. Die Funktionen $w_\omega(x) := e^{i\omega x}$ bilden eine orthonormale Basis von $L^2(0,2\pi)$. Hierbei sollte bemerkt werden, daß alle w_ω mittels einer Skalierung aus

$$w(x) := e^{ix} \tag{B.5}$$

gewonnen werden. Skalierung bedeutet hier $w_\omega(x) = w(\omega x)$ für alle $\omega \in \mathbf{Z}$.

Betrachten wir nun den Hilbertraum $L^2(\mathbf{R})$, so muß wegen

$$\int_{-\infty}^{\infty} |f(x)|^2 \, dx < \infty \tag{B.6}$$

für jede Funktion f in $L^2(\mathbf{R})$ gelten

$$\lim_{x \to \pm\infty} f(x) = 0 \ . \tag{B.7}$$

Dies hat zur Folge, daß die Funktion

$$w(x) = e^{ix} = \cos x + i \sin x \tag{B.8}$$

nicht in $L^2(\mathbf{R})$ liegt, da diese sinusförmige Welle nicht im Unendlichen verschwindet. Wir suchen also nach *Wellen*, die den Raum $L^2(\mathbf{R})$ erzeugen und (aus praktischen Gründen) sehr schnell gegen Null gehen, wenn x gegen $\pm\infty$ geht.

Das Skalarprodukt und die Norm für den Hilbertraum $L^2(\mathbf{R})$ lauten

$$\langle f, g \rangle := \int_{-\infty}^{\infty} f(x)\overline{g(x)} \, dx \tag{B.9}$$

$$\|f\| := \sqrt{\langle f, f \rangle} \ . \tag{B.10}$$

Wegen

$$\begin{aligned} \|f(\frac{x}{s})\| &= \sqrt{\int_{-\infty}^{\infty} |f(\frac{x}{s})|^2 \, dx} \\ &= \sqrt{s}\|f(x)\| \ , \end{aligned} \tag{B.11}$$

haben alle Funktionen von der Form

$$\psi_s(x) = \frac{1}{\sqrt{s}} \psi(\frac{x}{s}) \ , \qquad s \in \mathbf{R} \tag{B.12}$$

die Norm eins, wenn die Funktion ψ die Norm eins hat.

Um mit der begrenzten Funktion ψ ganz $\mathbf{R}$ abdecken zu können muß sie offensichtlich verschoben werden. Berücksichtigen wir weiterhin, daß wir eine Basis von $L^2(\mathbf{R})$ suchen, so benötigen wir Wellen verschiedener Frequenzen. Wir setzen

$$\psi_{j,k} := \frac{1}{\sqrt{2^j}} \psi(\frac{x-k}{2^j}) \ , \qquad j, k \in \mathbf{Z}^* \ . \tag{B.13}$$

Die $\psi_{j,k}$ bilden eine Menge kleiner Wellen, sogenannte *Wavelets*, die aus dem *Mutter-* oder *Basiswavelet* ψ, mittels einer dyadischen Skalierung um den Faktor $\frac{1}{2^j}$ und einer Verschiebung um k gewonnen werden.

In $L^2(\mathbf{R})$ können die Koeffizienten der Fourierreihe B.2 mittels

$$c_\omega = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} \, dx \ , \tag{B.14}$$

*Aus Gründen der leichteren Berechenbarkeit benutzen wir Zweierpotenzen für die Skalierung.

berechnet werden, dies kann auch durch das Skalarprodukt

$$c_\omega = \langle f, w_\omega \rangle \tag{B.15}$$

ausgedrückt werden. Analog läßt sich eine *Waveletreihe*

$$f(x) = \sum_{j=-\infty}^{\infty} \sum_{k=-\infty}^{\infty} c_{j,k} \psi_{j,k}(x) \ , \tag{B.16}$$

mit den Koeffizienten

$$c_{j,k} = \langle f, \psi_{j,k} \rangle \tag{B.17}$$

definieren. Die allgemeine *Wavelettransformation* auf $L^2(\mathbf{R})$ bezüglich dem Basiswavelet ψ lautet somit

$$(W_s^\psi f)(k) := \frac{1}{\sqrt{|s|}} \int_{-\infty}^{\infty} f(x) \overline{\psi\left(\frac{x-k}{s}\right)}\, dx \ , \tag{B.18}$$

dabei bezeichnet $s \in \mathbf{R}$ die Dilatation und $k \in \mathbf{R}$ die Translation des Mutterwavelets ψ. Die Wavelettransformation beinhaltet mit $\psi = e^{ix}$ die Fouriertransformation

$$(\mathcal{F}f)(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x}\, dx \ . \tag{B.19}$$

Für die Fouriertransformierte von $(W_{2^j}^\psi f)(k)$ gilt die Gleichung

$$\widehat{(W_{2^j}^\psi f)}(k) = \hat{f}(k) \hat{\psi}(2^j k) \ . \tag{B.20}$$

Wird ψ so gewählt, daß es die folgende *Stabilitätsbedingung*

$$A \leq \sum_{j=-\infty}^{\infty} |\hat{\psi}(2^j k)|^2 \leq B \ , \qquad 0 < A \leq B < \infty \tag{B.21}$$

erfüllt, so ist gewährleistet, daß die Wavelets den gesamten Frequenzraum abdecken und zur Rekonstruktion geeignet sind. Dabei ist das *duale* Wavelet ψ^* gegeben durch die Beziehung

$$\hat{\psi}^*(k) := \frac{\overline{\hat{\psi}(k)}}{\sum_{j=-\infty}^{\infty} |\hat{\psi}(2^j k)|^2} \ . \tag{B.22}$$

Die Rückgewinnung der Funktion $f \in L^2(\mathbf{R})$ ist mittels der folgenden Rekonstruktionsformel möglich

$$f(x) = \sum_{j=-\infty}^{\infty} \int_{-\infty}^{\infty} \left(\frac{1}{\sqrt{2^j}} (W_{2^j}^\psi f)(k)\right)\left(\frac{1}{2^j} \psi^*\left(\frac{x-k}{2^j}\right)\right) dk \tag{B.23}$$

B.2 Orthogonale Wavelets

Das Mutterwavelet ψ kann so gewählt werden, daß die Menge $\{\psi_{j,k}(x)\}$ eine Orthonormalbasis von $L^2(\mathbf{R})$ bildet und somit der Bedingung

$$\langle \psi_{j,k} \psi_{l,m} \rangle = \delta_{j,l} \cdot \delta_{k,m}, \qquad j,k,l,m \in \mathbf{Z}^\dagger \tag{B.24}$$

$^\dagger \delta_{j,k}$ ist das Kroneckersymbol mit $\delta_{j,k} := \begin{cases} 1 & : \ j = k \\ 0 & : \ j \neq k \end{cases}$

genügt. Jede Funktion $f \in L^2(\mathbf{R})$ kann in der Form von Gleichung B.16 geschrieben werden, mit Koeffizienten

$$c_{j,k} = (W_{2^j}^{\psi} f)(k) \ . \tag{B.25}$$

Zwei Bilder, die sich nur in einer Translation unterscheiden, können, bei Verwendung orthogonaler Wavelets, sehr unterschiedliche Wavelettransformationen besitzen. Diese Eigenschaft orthogonaler Wavelets stellt einen großen Nachteil bei der Bildverarbeitung dar.

B.3 Dyadische Wavelettransformation nach Mallat

Hier sollen die wichtigsten Eigenschaften der Mallatschen Wavelettransformation zusammengefaßt werden. Umfassendere Informationen können in [MZ91] und [Lyr93] gefunden werden. Bei dieser Transformation handelt es sich um nicht orthogonale Wavelets.

B.3.1 Eindimensionale Transformation

Sei $\theta(x) \in L^2(\mathbf{R})$ eine Glättungsfunktion mit der Norm eins, beispielsweise die Gaußsche Glockenkurve. Aus ihr wird mittels Ableitung das Mutterwavelet für die Mallatsche Wavelettransformation gewonnen

$$\psi(x) = \frac{d}{dx}\theta(x) \ . \tag{B.26}$$

Die Wavelets ergeben sich nach der Formel

$$\psi_{2^j}(x) = \frac{1}{2^j}\psi(\frac{x}{2^j}) \ . \tag{B.27}$$

Die Wavelettransformation wird bei Mallat als die Faltung

$$(W_{2^j}^{\psi} f)(y) := (f * \psi_{2^j})(y) = \int_{-\infty}^{\infty} f(x)\psi_{2^j}(y-x)\,dx \tag{B.28}$$

notiert, was für reelle ψ, f äquivalent zu der Gleichung B.18 ist. Durch Umformung erhält man

$$\begin{aligned}(W_{2^j}^{\psi} f)(y) &= (f * \psi_{2^j})(y) \\ &= (f * 2^j \frac{d}{dy}\theta_{2^j})(y) \\ &= 2^j \frac{d}{dy}(f * \theta_{2^j})(y) \ .\end{aligned} \tag{B.29}$$

Die lokalen Extrema der Transformierten $(W_{2^j}^{\psi} f)(y)$ entsprechen den Wendepunkten des mit der Glättungsfunktion θ_{2^j} gefalteten Signals f. Ist θ_{2^j} eine Gaußfunktion, so ist eine Bestimmung dieser Extrema äquivalent zu einer Kantenextraktion nach Canny [Can86]. Für kleine j ist die Glättung der Funktion f durch θ_{2^j} vernachlässigbar, und die Extrema entsprechen den Stellen der schärfsten Veränderungen von $f(x)$. Bei großen Werten für j wird das Signal stark geglättet, und wir erhalten nur die scharfen Änderungen großer Strukturen. Dabei entsprechen die Maxima den scharfen Änderungen und die Minima den langsamen Verändungen.

Die Rekonstruktion des ursprünglichen Signals ist durch

$$f(x) = \sum_{j=-\infty}^{\infty} (W_{2^j}^{\psi} f) * \psi_{2^j}^{*}(x) \tag{B.30}$$

möglich.

B.3.2 Zweidimensionale Transformation

Analog zur eindimensionalen Transformation erhalten wir die Wavelets in der Form

$$\psi_{2^j}^1(x,y) = \frac{\partial}{\partial x}\theta_{2^j}(x,y) \ , \tag{B.31}$$

$$\psi_{2^j}^2(x,y) = \frac{\partial}{\partial y}\theta_{2^j}(x,y) \ . \tag{B.32}$$

Dabei ist die Skalierung mit dem Faktor 2^j gegeben durch

$$\psi_{2^j}^1(x,y) = \frac{1}{2^{2j}}\psi^1(\frac{x}{2^j},\frac{y}{2^j}) \ , \tag{B.33}$$

$$\psi_{2^j}^2(x,y) = \frac{1}{2^{2j}}\psi^2(\frac{x}{2^j},\frac{y}{2^j}) \ . \tag{B.34}$$

Die Wavelettransformation einer Funktion $f(x,y) \in L^2(\mathbf{R}^2)$ besteht aus den beiden Komponenten

$$(W_{2^j}^{\psi^1} f)(x,y) = f * \psi_{2^j}^1(x,y) \ , \tag{B.35}$$

$$(W_{2^j}^{\psi^2} f)(x,y) = f * \psi_{2^j}^2(x,y) \ . \tag{B.36}$$

Wie in Gleichung B.29 kann gezeigt werden, daß

$$\begin{pmatrix} (W_{2^j}^{\psi^1} f)(x,y) \\ (W_{2^j}^{\psi^2} f)(x,y) \end{pmatrix} = 2^j \vec{\nabla}(f * \theta_{2^j})(x,y) \tag{B.37}$$

gilt, das heißt die Positionen der Kanten auf der jeweiligen Auflösungsskala können aus $(W_{2^j}^{\psi^1} f)(x,y)$ und $(W_{2^j}^{\psi^2} f)(x,y)$ gewonnen werden. Die zweidimensionale dyadische Wavelettransformation ist gegeben durch das Zweitupel:

$$\mathbf{W}f = ((W_{2^j}^{\psi^1} f)(x,y), (W_{2^j}^{\psi^2} f)(x,y)) \ . \tag{B.38}$$

Für die Fouriertransformation von $(W_{2^j}^{\psi^1} f)(x,y)$ und $(W_{2^j}^{\psi^2} f)(x,y)$ gilt

$$\widehat{(W_{2^j}^{\psi^1} f)}(k_x,k_y) = \hat{f}(k_x,k_y)\hat{\psi}^1(2^j k_x, 2^j k_y) \ , \tag{B.39}$$

$$\widehat{(W_{2^j}^{\psi^2} f)}(k_x,k_y) = \hat{f}(k_x,k_y)\hat{\psi}^2(2^j k_x, 2^j k_y) \ . \tag{B.40}$$

Damit lautet die Stabilitätsbedingung im Zweidimensionalen

$$A \leq \sum_{j=-\infty}^{\infty} (|\hat{\psi}^1(2^j k_x, 2^j k_y)|^2 + |\hat{\psi}^2(2^j k_x, 2^j k_y)|^2) \leq B \ . \tag{B.41}$$

Die Formel für die Rückgewinnung der Funktion $f(x,y) \in L^2(\mathbf{R}^2)$ lautet

$$f(x,y) = \sum_{j=-\infty}^{\infty} ((W_{2^j}^{\psi^1} f) * {\psi_{2^j}^1}^*(x,y) + (W_{2^j}^{\psi^2} f) * {\psi_{2^j}^2}^*(x,y)) \ , \tag{B.42}$$

mit

$$\psi_{2^j}^{1\,*}(x,y) = \frac{\overline{\hat{\psi}^1(k_x,k_y)}}{2\sum\limits_{j=-\infty}^{\infty}|\hat{\psi}^1(2^j k_x, 2^j k_y)|^2} \ , \tag{B.43}$$

$$\psi_{2^j}^{2\,*}(x,y) = \frac{\overline{\hat{\psi}^2(k_x,k_y)}}{2\sum\limits_{j=-\infty}^{\infty}|\hat{\psi}^2(2^j k_x, 2^j k_y)|^2} \ . \tag{B.44}$$

Literaturverzeichnis

[Bra91] R. Brause. *Neuronale Netze.* B. G. Teubner Stuttgart, 1991.

[Beh93] K. Behrmann. Leistungsuntersuchung des „Dynamischen Link-Matchings“ und Vergleich mit dem Kohonen-Algorithmus. Diplomarbeit, Universität Karlsruhe, Institut für Logik, Komplexität und Deduktionssysteme, durchgeführt am Institut für Neuroinformatik, Ruhr-Universität Bochum, 1993.

[Can86] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[Chu92a] C. K. Chui, editor. *An Introduction to Wavelets.* Academic Press, 1992.

[Chu92b] C. K. Chui, editor. *Wavelets: A Tutorial in Theory and Applications.* Academic Press, 1992.

[CK89] A. C. C. Coolen and F. W. Kujik. A learning mechanism for invariant pattern recognition in neural networks. *Neural Networks*, 2:495–506, 1989.

[FH88] A. Fuchs and H. Haken. Pattern recognition ans associative memory as dynamical processes in a synergetic system. *Biological Cybernetics*, (60):107–109, 1988.

[Hay82] M. H. Hayes. The reconstruction of a multidimensional sequence from the phase or magnitude of its fourier transform. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 30(2):140–154, 1982.

[HKP91] J. Hertz, A. Krogh, and R. G. Palmer. *Introduction to the Theory of Neural Computation.* Addison-Wesley Publishing Company, 1991.

[HN90] R. Hecht-Nielsen. *Neurocomputing.* Addison-Wesley Publishing Company, 1990.

[Hop82] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. In *Proceedings of the National Academy of Sciences*, volume 79, pages 2554–2558, 1982.

[KMvdM94] W. Konen, T. Maurer, and C. v. d. Malsburg. A fast dynamic link matching algorithm for invariant pattern recognition. *Neural Networks*, 1994. In press.

[Knu73] D. E. Knuth. *The Art of Computer Programming*, volume 3 Sorting and Searching. Addison-Wesley Publishing Company, 1973.

[Koh82] T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, (43):59–69, 1982.

[Koh84] T. Kohonen. *Self-Organization and Associative Memory.* Spinger-Verlag, third edition, 1984.

[KV93] W. Konen and J. C. Vorbrüggen. Applying dynamic link matching to object recognition in real world images. In S. Gielen, editor, *Proceedings of the International Conference on Artificial Neural Networks.* North-Holland, Amsterdam, 1993.

[Lyr93] H. E. Lyre. Bildverarbeitung mit Wavelets zur Extraktion von Mehrskalen-Kanten. Diplomarbeit, Universität Dortmund, Fachbereich Physik, durchgeführt am Institut für Neuroinformatik, Ruhr-Universität Bochum, 1993.

[MP88] M. L. Minsky and S. A. Papert, editors. *Perceptrons: An Introduction to Computational Geometry.* MIT Press, 1988.

[MZ91] S. Mallat and S. Zhong. Characterization of signals from multiscale edges. Technical report, Courant Institute of Mathematical Sciences, New York University, 1991.

[RMS91] H. Ritter, T. Martinetz, and K. Schulten. *Neuronale Netze.* Addison-Wesley Publishing Company, zweite edition, 1991.

[Ros58] F. Rosenblatt. The perception: a probabilistic model for information storage and organization in the brain. *Psychological Review*, 65:386–408, 1958.

[Sed88] R. Sedgewick. *Algorithms.* Addison-Wesley Publishing Company, second edition, 1988.

[Sha92] C. J. Shatz. Das sich entwickelnde Gehirn. *Spektrum der Wissenschaft*, pages 44–52, November 1992.

[She88] G. M. Shepherd. *Neurobiology.* Oxford University Press, second edition, 1988.

[ST80] R. F. Schmidt and G. Thews. *Physiologie des Menschen.* Springer-Verlag, zwanzigste edition, 1980.

[vdM73] C. v. d. Malsburg. Self-organization of orientation sensitive cells in the striate cortex. *Kybernetik*, 14:85–100, 1973.

[WvdM96] L. Wiskott and C. v. d. Malsburg. Face Recognition by Dynamic Link Matching. Internal report, IRINI 96-05, Institut für Neuroinformatik, Ruhr-Universität Bochum, 1996.
ebenfalls im elektronischen Buch Sirosh, J., Miikkulainen, R., and Choe, Y., editors, *Lateral Interactions in the Cortex: Structure and Function.* The UTCS Neural Networks Research Group, Austin, TX. Electronic book, ISBN 0-9647060-0-8,
http://www.cs.utexas.edu/users/nn/web-pubs/htmlbook96 Kap. 11.

[Zad91] S. Zadel. Einführung in Neuronale Netzwerke und deren Verwirklichung durch elektrische und optische Systeme. Diplomarbeit, Universität Stuttgart, Institut für Netzwerk- und Systemtheorie, 1991.