

Selbstorganisierte Generierung und Adaption polygonaler Netze mit
variabler Topologie zur Beschreibung von Solids und 3D-Flächen

Diplomarbeit

zur Erlangung des akademischen Grades
Diplom-Informatiker / Diplom-Informatikerin

an der
Fachhochschule für Technik und Wirtschaft Berlin

Fachbereich Wirtschaftswissenschaften II
Studiengang Angewandte Informatik

1. Betreuer: Prof. Dr. Thomas Jung
2. Betreuer: Dipl.-Ing. Lothar Paul

Eingereicht von Jan Hambrecht

Berlin, den 4. November 2002

Inhaltsverzeichnis

1	Einführung	1
2	Verfahren zur Digitalisierung freigeformter räumlicher Objekte	3
2.1	Projektion einfacher geometrischer Muster	5
2.1.1	Lichtpunkttechnik	5
2.1.2	Lichtpunktstereoanalyse	6
2.1.3	Lichtschnittechnik	7
2.1.4	Statische Streifenprojektion	8
2.2	Projektion codierter Muster	8
2.2.1	Dynamische Streifenprojektion	9
2.2.2	Binärcodierte Lichtschnittechnik	10
2.2.3	Moiré-Technik	11
2.2.4	Farbcodierte Lichtschnittechnik	12
2.2.5	Aktive Stereoanalyse mit Farbe	13
2.3	Spezifik von 3D-Scandaten	13
3	Polygonale Netze	15
3.1	Einsatz und Bedeutung	15
3.2	Topologie und Geometrie	16
3.3	Triangulationsverfahren	17
3.3.1	Delaunay-Verfahren	17
3.3.2	Marching-Cubes-Verfahren	21
3.3.3	Competitive Mesh Adaption	23
3.3.4	Bewertung der Verfahren	25
4	Selbstorganisierte Systeme	29
4.1	Selbstorganisierte Systeme in der Natur	29
4.2	Biologische neuronale Netze	31
4.3	Künstliche neuronale Netze	32
4.3.1	Informationsverarbeitung eines Neurons	33
4.3.2	Informationsverarbeitung eines neuronalen Netzes	35
4.3.3	Wichtige Eigenschaften neuronaler Netze	36
4.3.4	Lernverfahren neuronaler Netze	38
4.4	Einige Modelle neuronaler Netze	41
4.4.1	Topologieerhaltende Abbildungen	41
4.4.2	Self-Organizing Feature Map	42

4.4.3	Growing Cell Structures	44
4.4.4	Neural Gas	48
4.4.5	Competitive Hebbian Learning	49
4.4.6	Growing Neural Gas	50
4.4.7	Bewertung der Modelle	53
5	Ein alternatives adaptives Triangulationsverfahren	57
5.1	Anforderungen	57
5.1.1	Erzeugung konsistenter Dreiecksnetze	57
5.1.2	Adaptivität	57
5.1.3	Flexible Startinitialisierung	58
5.1.4	Sinnvolle Abbruchkriterien	58
5.1.5	Integration in bestehende Softwarebibliothek	58
5.2	Konzeption und Entwurf	59
5.2.1	Grunddesign	59
5.2.2	Erweiterungen	60
5.2.3	Modularisierung	62
5.2.4	Klassenstruktur	63
5.3	Implementierung	67
5.3.1	Verfahrensbeschreibung	67
5.3.2	Probleme und Besonderheiten	74
5.4	Ergebnisse	82
5.4.1	Eine ebene Punktwolke	82
5.4.2	Ein gescannter Zahnstumpf	84
5.4.3	Adaption eines vorgefertigten Netzes	86
5.5	Fazit und Ausblick	89
	Literaturverzeichnis	90
	Index	92
	A Programmablaufplan	I
	B Eigenständigkeitserklärung	III

Abbildungsverzeichnis

2.1	Klassifizierung von 3D-Scanverfahren	4
2.2	Abschattung und Verdeckung bei nichtebenen Objekten	5
2.3	Vereinfachte Darstellung der Lichtpunkttechnik	6
2.4	Vereinfachte Darstellung der Lichtschnitttechnik	7
2.5	Dynamische Streifenprojektion bei telezentrischer Anordnung	9
2.6	Gray-Code für Anzahl der Muster $m = 3$	10
2.7	Vereinfachte Darstellung der Moiré-Technik	11
3.1	Voronoi-Diagramm und Delaunay-Triangulation	18
3.2	Schließen von Konkavitäten	19
3.3	Delaunay-Triangulation im dreidimensionalen Raum	20
3.4	Sieben Fälle zum Verbinden von Schnittpunkten	22
3.5	Dreiecksverfeinerung	25
4.1	Gesamtansicht und Nahaufnahme von Konvektionszellen	30
4.2	Genereller Aufbau eines künstlichen neuronalen Netzes	32
4.3	Aufbau eines künstlichen Neurons (Zelle)	33
4.4	Verschiedene Aktivierungsfunktionen	34
4.5	Mögliche Ausgabefunktionen für konkurrierendes Lernen	43
5.1	Verbinden zweier Teilnetze	61
5.2	Kollaborationsdiagramm	65
5.3	Schließen einer Konkavität	71
5.4	Abstand eines Punktes zu einem Simplex	73
5.5	Zellbewegung außerhalb der Nachbarzellen	76
5.6	Unvollständige Abbildung der Punktwolke	78
5.7	Geometrische Inkonsistenz beim Verbinden von Zellen	79
5.8	Raumteilung durch Slots	80
5.9	Simplex - Slot - Überschneidung	81
5.10	Triangulation einer ebenen Punktwolke mit separaten Teilgebieten	83
5.11	Triangulation eines gescannten Zahnstumpfes (Teilschritte)	85
5.12	Triangulation eines gescannten Zahnstumpfes (Endstadium)	86
5.13	Adaption eines vorgefertigten Dreiecksnetzes (Teilschritte)	87
5.14	Adaption eines vorgefertigten Dreiecksnetzes (Endstadium)	88
A.1	Programmablaufplan des adaptiven Triangulationverfahrens	II

Symbolverzeichnis

a_i	Aktivität einer Zelle c_i
$a_i(t)$	Aktivität einer Zelle c_i zum Zeitpunkt t
$C = \{c_1, \dots, c_n\}$	Menge der Zellen eines künstlichen neuronalen Netzes
c_i	Zelle i eines künstlichen neuronalen Netzes
$F(G)$	Aktivierungsfunktion
$G(x)$	Propagierungsfunktion
K	Menge der ungewichteten, symmetrischen Nachbarschaftsverbindungen zwischen den Zellen eines künstlichen neuronalen Netzes
$N_c = \{i \in C (c, i) \in K\}$	Menge der direkten topologischen Nachbarn der Zelle c
$ N_c $	Anzahl der direkten topologischen Nachbarn der Zelle c
net_i	gewichtete Eingabe zu einer Zelle c_i
$p(\xi)$	Wahrscheinlichkeitsverteilung der Eingabesignale im Eingaberaum
$w = (w_1, \dots, w_n)$	Gewichtsvektor
w_{ij}	Gewicht von c_j nach c_i
$x = (x_1, \dots, x_n)$	Eingabevektor
$y = (y_1, \dots, y_n)$	Ausgabevektor
ϵ	Lernrate für Zellen eines künstlichen neuronalen Netzes
ξ	Eingabesignal
$\ v\ $	euklidische Vektornorm von Vektor v

Kapitel 1

Einführung

Die Vermessung, Digitalisierung und Rekonstruktion freigeformter räumlicher Objekte ist ein Anwendungsfeld, welches seit wenigen Jahren aus Wirtschaft und Technik nicht mehr wegzudenken ist. Beispiele sind die computergestützte Konstruktion in der Automobilindustrie oder der Zahnmedizin, Reverse Engineering im Design-Bereich oder Anwendungen in der Bekleidungsindustrie.

Ausgangspunkt für die rechnerinterne Weiterverarbeitung sind reale freigeformte Vorlagen, welche mittels dreidimensionaler, zunehmend optischer Scanverfahren in Form dichter Punktwolken erfaßt werden. Diese sind dann in rechnerinterne Objektrepräsentationen wie Oberflächen- oder Volumenmodelle umzurechnen. Die dabei am meisten verbreitete Form sind Polygonnetze. Sie stellen eine Approximation der Objektoberfläche dar und sind durch geometrische und topologische Eigenschaften charakterisiert. Die Geometrie wird durch die korrespondierende Punktwolke bestimmt. Die Topologie ist bezogen auf die diskrete Punktwolke eine neue Qualität der Information.

Die automatische Erzeugung dieser Netze erfolgt durch verschiedene Triangulationsalgorithmen, wie dem Delauny- oder dem Marching-Cubes-Verfahren. Diese generieren Netze mit einerseits einer guten geometrischen Anpassung und andererseits einer verfahrensbedingten Topologie. Diese ist jedoch nicht immer optimal im Sinne der Nutzeranforderungen und nicht in jedem Fall konsistent (siehe spätere Definition).

Die Netze können aber auch durch die Anpassung schon vorgefertigter Modelle an die vermessene Punktwolke entstehen. Dies hat den Vorteil einer topologischen Konsistenz der erzeugten Oberflächenmodelle. Zusätzlich besteht die Möglichkeit der fließenden Anpassung dieser Modelle an sich verändernde Daten. Dabei ist zu unterscheiden, ob sich diese Anpassung nur auf die Geometrie (Form) des Netzes oder auch auf dessen Topologie auswirkt.

Ziel dieser Arbeit ist es, ein alternatives adaptives Triangulationsverfahren zu entwickeln.

Dieses soll die Fähigkeit haben, automatisiert Oberflächenmodelle aus beliebigen Punktwolken zu erzeugen, sowie bestehende Modelle an veränderte oder neue Punktwolken anzupassen.

Bestimmte Arten neuronaler Netze weisen einen hohen Grad an struktureller Ähnlichkeit zu Polygonnetzen auf. Dies legt den Gedanken nahe, spezifische adaptive Mechanismen aus der Neuroinformatik für die genannte Zielstellung zu untersuchen beziehungsweise zu implementieren.

Ausgangspunkt und Bestandteil der vorliegenden Arbeit ist die Analyse von Verfahren und Veröffentlichungen zu folgenden Themen:

- dreidimensionale Scanverfahren
- Generierung von Oberflächenmodellen aus Punktwolken
- Anpassung von Oberflächenmodellen an beliebige Punktwolken
- neuronale Netze

Diese Ausarbeitung ist in mehrere Kapitel unterteilt, welche sich mit den verschiedenen Teilgebieten dieses Themas auseinandersetzen:

- Im Kapitel 2 werden unterschiedliche dreidimensionale Scanverfahren zur Gewinnung von Punktwolken aus räumlichen Objekten vorgestellt.
- Das Kapitel 3 beschäftigt sich mit den Triangulationsalgorithmen, welche zur Erzeugung von Oberflächenmodellen aus Punktwolken eingesetzt werden. Dies schließt einen Vergleich anhand ihrer Eigenschaften und eine anschließende Bewertung mit ein.
- Die Grundlagen neuronaler Netze und Prinzipien der Selbstorganisation werden in Kapitel 4 vorgestellt. Hier werden einige Modelle neuronaler Netze beschrieben und miteinander verglichen.
- Das letzte Kapitel beinhaltet den Entwurf, die Implementierung sowie die erhaltenen Ergebnisse des entwickelten adaptiven Triangulationsverfahrens. Hier wird zusätzlich auf auftretende Probleme und deren Lösung eingegangen.

Kapitel 2

Verfahren zur Digitalisierung freigeformter räumlicher Objekte

Es gibt eine Vielzahl unterschiedlicher Ansätze, um freigeformte räumliche Objekte zu vermessen. Die verschiedenen Verfahren können wie in Abbildung 2.1 klassifiziert werden. Innerhalb dieses Abschnittes sollen einige 3D-Scanverfahren zur Oberflächenvermessung vorgestellt werden. Der Schwerpunkt dieses Kapitels liegt auf den optischen Scanverfahren, welche die Objektpunkte mittels einer Triangulierung berechnen.

Wie in der Abbildung zu sehen ist, wird bei diesen Verfahren zwischen passiven und aktiven unterschieden. Bei den passiven Verfahren werden Objekte mittels einer oder mehrerer Kameras beobachtet und aus den so gewonnenen Bildern die Tiefenwerte der Objektpunkte berechnet. Als Beispiel sei hier die statische Stereoanalyse (siehe dazu Klette et al., 1996, Kapitel 4: Statische Stereoanalyse) genannt.

Die aktiven Scanverfahren zeichnen sich durch die Projektion von Mustern auf die Objekte der zu beobachtenden Szene aus. Die Muster werden mittels einer oder mehrerer Kameras aufgenommen. Anschließend wird das aufgenommene Objektmuster analysiert und die Tiefenwerte der Objektpunkte berechnet. Die hier beschriebenen aktiven Verfahren arbeiten mit strukturierter Beleuchtung der zu vermessenen Objekte und Triangulierung zur Berechnung der Tiefenwerte.

Strukturierte Beleuchtung bedeutet, daß ein oder mehrere Lichtmuster auf die Objekte der Szene projiziert werden. Die von den Objekten reflektierten Lichtmuster werden von einer Kamera unter einem Winkel ϑ erfaßt und rechnergestützt ausgewertet. Durch Analyse der reflektierten Muster können die Entfernung zwischen Objekt und Kamera und die Lage des Objektes im Raum bestimmt werden. Bei der Verwendung einer strukturierten Beleuchtung sind meist mehrere Bildaufnahmen notwendig. Die zu vermessenden Objekte müssen deswegen fixiert werden, um eine korrekte Vermessung zu ermöglichen.

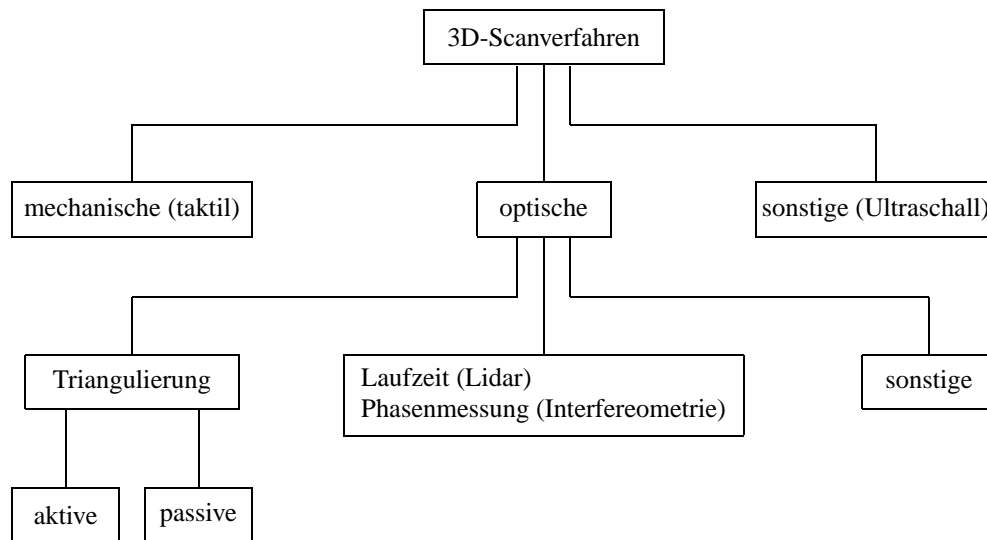


Abbildung 2.1: Klassifizierung von 3D-Scanverfahren

Die erreichbare Genauigkeit der optischen Scanverfahren hängt von mehreren Faktoren ab. Dazu gehören systematische Faktoren, welche alle Triangulierungsverfahren gleichermaßen beeinflussen:

- Sensorauflösung
- Größe des vermessenen Objektausschnitts
- Beobachtungs- beziehungsweise Triangulierungswinkel

Die Auflösung des Sensors (z.B. Kamera) spielt eine wichtige Rolle. Sie legt die Anzahl der Meßpunkte für den vermessenen Objektausschnitt fest. Dessen Größe bestimmt die Dichte der Meßpunkte. Die Meßgenauigkeit bei den Triangulierungsverfahren kann in Abhängigkeit von der Meßfeldgröße bis zu einem Mikrometer betragen. Der Beobachtungswinkel hat einen direkten Einfluß auf die Berechnung der Tiefenwerte. Bei einem großen Triangulierungswinkel und der Vermessung von nichtebenen Objekten kann es zu Abschattungen des projizierten Musters oder zu Verdeckungen von Objektteilen aus Sicht des Sensors kommen (siehe Abbildung 2.2). In diesen Bereichen können keine Objektpunkte vermessen werden. Es gibt aber auch Faktoren, welche einen unterschiedlich starken Einfluß auf die verschiedenen vorgestellten Verfahren haben:

- Oberflächeneigenschaften (Reflektion, Spiegelung, Stetigkeit)
- Umgebungshelligkeit oder Fremdlicht
- technische Parameter der verwendeten optischen und elektrischen Module

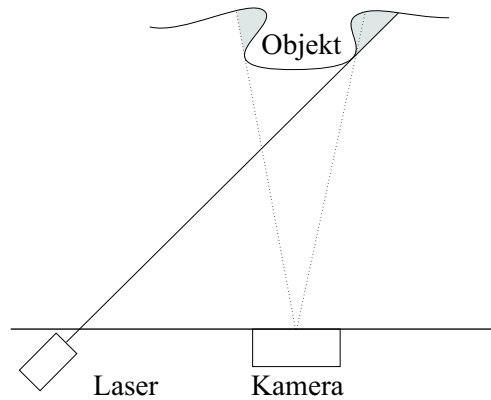


Abbildung 2.2: Abschattung und Verdeckung bei nichtebenen Objekten

Die optischen Verfahren sind auf homogene, diffus reflektierende Objektoberflächen angewiesen, um gute Ergebnisse zu erzielen. Aufgrund von Spiegelungen und Glanzlichtern kann es aber teilweise zu starken Verfälschungen der aufgenommenen Muster kommen. Die Umgebungshelligkeit kann zu einer Kontrastreduzierung in den aufgenommenen Mustern führen und so Meßungenauigkeiten hervorrufen.

Für detailliertere Informationen, die über diese Übersicht hinausgehen, sei auf die weiterführende Literatur wie Breuckmann (1993), Klette et al. (1996) und Luhmann (2000) verwiesen.

2.1 Projektion einfacher geometrischer Muster

Die projizierten Muster müssen im Kamerabild lokalisiert werden, um die Berechnung einer Tiefenkarte für das zu vermessende Objekt zu ermöglichen. Damit diese Lokalisierung einfach bleibt, werden bei den hier vorgestellten Verfahren nur einfache geometrische Muster verwendet. Diese Muster sind der Lichtpunkt und der Lichtstreifen.

2.1.1 Lichtpunkttechnik

Die Lichtpunkttechnik ist eine sehr einfache Methode zur optischen Oberflächenvermessung (siehe Breuckmann, 1993, S.126f). Dabei wird ein Lichtpunkt mittels eines Laserstrahls auf das Objekt projiziert. Das vom Objekt reflektierte Licht wird von einem Sensor (zum Beispiel einer Kamera) aufgefangen. Die Entfernung des beleuchteten Objektpunktes zur Kamera wird mittels einer Triangulierung berechnet.

In Abbildung 2.3 ist eine Meßanordnung mit senkrechter Beobachtung des Objektes darge-

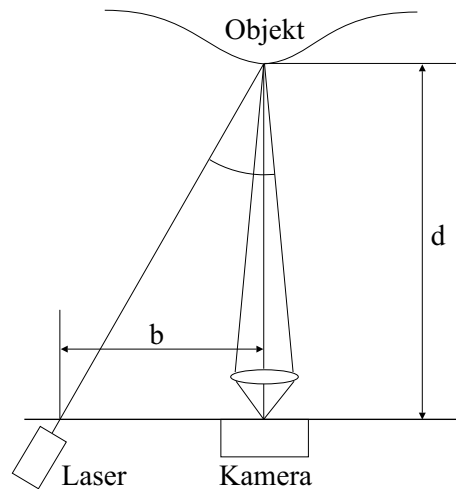


Abbildung 2.3: Vereinfachte Darstellung der Lichtpunkttechnik

stellt. Für diesen Fall gilt:

$$d = \frac{b}{\tan \vartheta} \quad (2.1)$$

Dabei ist b der Abstand zwischen Beleuchtungspunkt und Beobachtungspunkt (Basislänge) und d der zu ermittelnde Abstand zwischen Objektpunkt und Kamera. Wie aus der Gleichung 2.1 ersichtlich wird, steigt die Meßgenauigkeit bei zunehmenden Winkel ϑ und gleicher Basislänge b . Das heißt, daß sich bei gleicher Basislänge und abnehmenden Abstand vom Objekt der Winkel ϑ vergrößert und damit die Meßgenauigkeit steigt.

Dieses Verfahren kann auch zur flächenhaften Vermessung von Objekten eingesetzt werden - als sogenanntes Laser-Scan-Verfahren. Dafür ist jedoch ein sehr aufwendiges System für die zweidimensionale Positionierung des Lasers nötig.

Bei der praktischen Anwendung dieses Verfahrens sind einige zusätzliche Probleme zu beachten (siehe Klette et al., 1996, S.343f), auf die hier aber nicht näher eingegangen werden soll.

2.1.2 Lichtpunktstereoanalyse

Dieses Verfahren basiert auf der Lichtpunkttechnik, sowie der statischen Stereoanalyse (siehe Klette et al., 1996, S.344-346). Dabei wird der auf das zu vermessende Objekt projizierte Laserpunkt von zwei verschiedenen Kameras aufgenommen. Der Objektpunkt ist in beiden aufgenommenen Bildern sichtbar und kann als korrespondierendes Bildpunktpaar für die Stereoanalyse benutzt werden.

Der Vorteil gegenüber der Lichtpunkttechnik besteht darin, daß hier keine komplizierte und fehleranfällige Kalibrierung der Laserablenkeinheit zur Positionierung des Laserpunktes

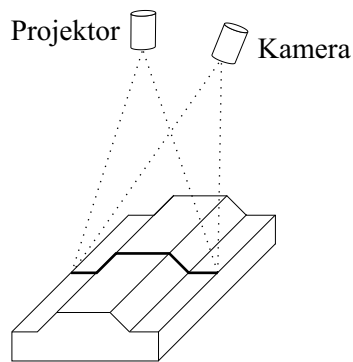


Abbildung 2.4: Vereinfachte Darstellung der Lichtschnitttechnik

vorgenommen werden muß. Dies ist möglich, da die Orientierung des Laserstrahls nicht in die Berechnung des Objektpunktes eingeht. Für einige Anwendungen kann es von Vorteil sein, daß mit diesem Verfahren die Vermessung von einzelnen Punkten oder Kurven möglich ist. Dadurch können die zu verarbeitenden Datenmengen eingeschränkt werden.

Von Nachteil sind jedoch die langen Rechenzeiten, da für jeden zu vermessenden Objektpunkt jeweils zwei Bilder verarbeitet werden müssen.

2.1.3 Lichtschnitttechnik

Die Lichtschnitttechnik ist als eine Erweiterung der Lichtpunkttechnik anzusehen (siehe Klette et al., 1996; Breuckmann, 1993). Hier wird nicht nur ein einzelner Laserpunkt, sondern ein Lichtband beziehungsweise eine Lichtebene auf die Objekte der Szene projiziert. Diese Lichtebene kann mittels Laserlicht und zylindrischen Linsen oder einem Diaprojektor mit Schlitzmaske erzeugt werden. In Abbildung 2.4 ist ein vereinfachter Meßaufbau dargestellt, bei dem die Ebene senkrecht auf das Objekt projiziert wird.

Die Schnittlinie der Lichtebene mit den Objekten der Szene ist im Kamerabild in Abhängigkeit vom Projektionswinkel und von der Objektgeometrie deformiert sichtbar und kann mittels Kantendetektion und Segmentierung des Bildes identifiziert werden. Die Auswertung der Schnittlinie erfolgt wieder über die Triangulierungsgesetze und liefert gleichzeitig die Tiefenwerte mehrerer Objektpunkte.

Die Tiefenauflösung des Verfahrens ist von der Breite der projizierten Lichtebene und der Auflösung des Bildsensors abhängig. Problematisch ist der Fall, bei dem die reflektierte Lichtebene im Nahbereich mehrere Pixel breit und im weiter entfernten Bereichen kleiner als ein Pixel breit ist. Dort kann sie mit der Kamera unter Umständen nicht mehr detektiert werden. Dadurch wird eine Berechnung des entsprechenden Tiefenwertes erschwert beziehungsweise unmöglich.

Ein Vorteil des Verfahrens ist die Möglichkeit der kompletten Messung der Objektoberfläche. Dabei wird das Objekt oder der Sensor gedreht oder verschoben und die reflektierte Lichtebene detektiert und ausgewertet.

2.1.4 Statische Streifenprojektion

Diese Technik ist eine Erweiterung der schon erläuterten Lichtschnitttechnik (siehe Klette et al., 1996; Breuckmann, 1993). Hierbei werden gleichzeitig mehrere Lichtebenen, ein sogenanntes Streifengitter, auf das zu vermessende Objekt projiziert. Dadurch kann die Anzahl der insgesamt für die Berechnung der Tiefenkarte des Objektes benötigten Aufnahmen reduziert werden.

Da die Ebenen aber nicht mehr senkrecht auf das Objekt projiziert werden können, wird die Berechnung der Tiefenwerte für die Objektpunkte aufwendiger. Außerdem ist die eindeutige Zuordnung zwischen projizierten und aufgenommenen Lichtstreifen problematisch. Durch Verdeckungen und Unstetigkeiten der Objektoberfläche können einzelne Streifen im Kamerabild unterbrochen oder gänzlich unsichtbar bleiben. Dadurch kann nicht oder nur sehr schwer festgestellt werden, welche Streifen im Kamerabild erscheinen und welche nicht.

Aufgrund dieser Probleme wird diese Technik hauptsächlich für planare und stetige Objektoberflächen verwendet, bei denen die Identifikation der Streifennummer durch einfaches Abzählen möglich ist..

2.2 Projektion codierter Muster

Das Projizieren mehrerer oder komplexer Lichtmuster kann, wie oben beschrieben, wegen Verdeckungen und Abschattungen zu Problemen führen. So kann eine eindeutige Zuordnung zwischen projiziertem und aufgenommenem Muster nicht sichergestellt werden.

Durch räumliche oder zeitliche Codierung des projizierten Musters können diese Probleme umgangen werden. Dadurch ist es möglich, das aufgenommene Muster mittels dieser Codierung eindeutig zu identifizieren und zuzuordnen.

Folgende Ziele sollen durch Projektion codierter Muster erreicht werden:

- höhere Auflösung
- eindeutige Musteridentifikation

Für das Erreichen einer höheren Auflösung kann die dynamische Streifenprojektion oder die Moiré-Technik verwendet werden. Eine eindeutige Identifikation des reflektierten Musters kann über die binärcodierte und die farbcodierte Lichtschnitttechnik erzielt werden.

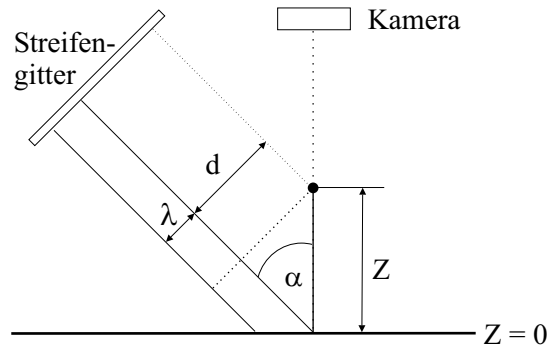


Abbildung 2.5: Dynamische Streifenprojektion bei telezentrischer Anordnung

2.2.1 Dynamische Streifenprojektion

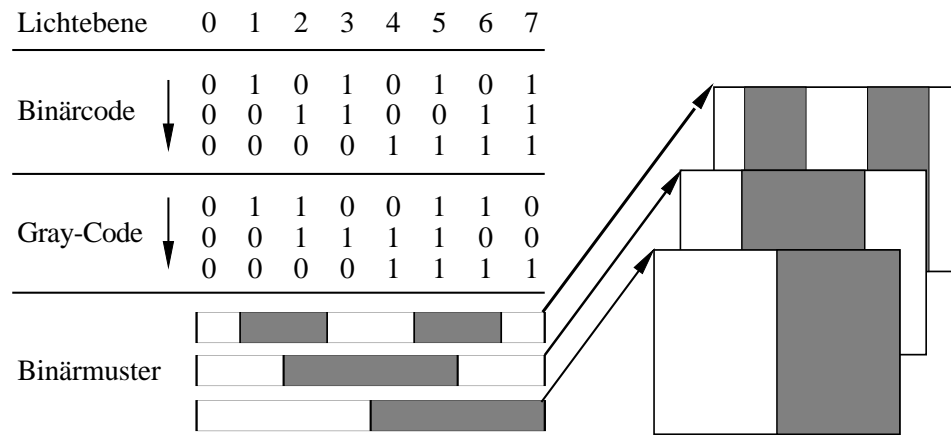
Bei der dynamischen Streifenprojektion, auch als *Phasen-Schiebe-Verfahren* bekannt, wird ein Streifengitter mit sinusförmiger Helligkeitsverteilung mit einer konstanten Wellenlänge λ auf das zu vermessende Objekt projiziert (Breuckmann, 1993; Luhmann, 2000). Die Streifen des Musters sind meist 5-10 mal breiter als die bei einer gewöhnlichen Streifenprojektion. Dies ist nötig, um angesichts der begrenzten Auflösung der Kamera den Helligkeitsverlauf innerhalb eines Streifens noch erkennen zu können. Das Streifenmuster wird als Interferogramm betrachtet. Für jedes Bildpixel kann jeweils ein Intensitätswert $I(x, y)$ berechnet werden. Dieser ist von der Hintergrundhelligkeit, der Streifenmodulation durch den Projektor und der Phase des Bildpunktes innerhalb der Helligkeitsverteilung abhängig.

Der Tiefenwert eines Bildpunktes wird über dessen genaue Phase ermittelt. Diese ist eindeutig bestimmt, wenn mindestens drei verschiedene Intensitätswerte für diesen Bildpunkt gemessen werden. Die Werte lassen sich durch m -maliges Verschieben des Streifengitters um jeweils $2\pi/m$ gewinnen. Für vier Verschiebungen um $\pi/2$ und den daraus bestimmten Intensitätswerten $I_1 \dots I_4$ lässt sich die Phase $\delta(x, y)$ eines Bildpunktes wie folgt berechnen:

$$\delta(x, y) = \arctan \left(\frac{I_2 - I_4}{I_3 - I_1} \right) \quad (2.2)$$

Die Phase eines Bildpunktes kann aufgrund der kontinuierlichen Streifenanordnung mit hoher Genauigkeit bestimmt werden. Dadurch ist es möglich, die erreichbare Tiefenauflösung um etwa eine Größenordnung gegenüber der herkömmlichen Streifenprojektion zu steigern. Bei einer telezentrischen Projektion und Beobachtung des Streifenmusters (siehe Abbildung 2.5), können die Tiefenwerte der einzelnen Pixel für die Tiefenkarte mittels der nachstehenden Gleichung ermittelt werden:

$$Z(x, y) = \frac{\lambda}{\sin \alpha} \cdot \delta(x, y) \quad (2.3)$$

Abbildung 2.6: Gray-Code für Anzahl der Muster $m = 3$

Die hohe Verarbeitungsgeschwindigkeit ist hierbei von Vorteil, so daß mehrere hunderttausend Objektpunkte in kürzester Zeit erfaßt werden können. Diese hochdichten Punktwolken müssen dann meist vor der Weiterverarbeitung reduziert und ausgedünnt werden. Ein Nachteil ist, daß sich die Phase nur innerhalb eines Streifens bestimmen läßt. Die Gesamtphase muß wieder durch Abzählen der einzelnen Streifen ermittelt werden. Dies ruft das Problem der Streifenidentifikation wie bei den anderen Streifenprojektionsverfahren hervor. Deshalb wird dieses Verfahren auch oft mit der binärcodierten Lichtschnitttechnik kombiniert.

2.2.2 Binärcodierte Lichtschnitttechnik

Bei dieser Technik werden mehrere Lichtebenen auf das zu vermessende Objekt projiziert (Luhmann, 2000; Klette et al., 1996). Es werden zeitlich versetzt m binär codierte Lichtmuster erzeugt, um eine eindeutige Identifikation der einzelnen Ebenen zu ermöglichen. Diese ergeben für jede der 2^m erlaubten Lichtebenen eine eindeutige Codierung durch einen m -stelligen Binär-code. Als Codierung wird häufig der sogenannte *Gray-Code* benutzt. Die Nummer n der einzelnen Lichtebenen kann mit folgender Funktion in ein Codewort g des *Gray-Code* überführt werden (siehe Press et al., 1992, S.896):

$$g(n) = n \text{ xor } (n \text{ div } 2) \quad (2.4)$$

In Abbildung 2.6 ist ein *Graycode* für eine Anzahl von $m = 3$ Lichtmustern veranschaulicht. Der *Gray-Code* hat die Eigenschaft, daß sich benachbarte Codeworte jeweils in einem Bit unterscheiden. Dadurch wird der Fehler minimiert, welcher bei falscher Zuordnung eines Punktes im Kamerabild zu einer bestimmten Lichtebene entsteht. Der Fehler ist also

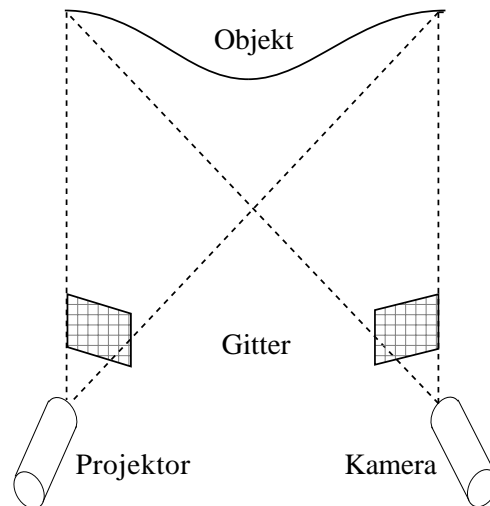


Abbildung 2.7: Vereinfachte Darstellung der Moiré-Technik

immer auf benachbarte Lichtebenen beschränkt.

Die vom Objekt reflektierten Muster werden von einer Kamera aufgenommen und anschließend binarisiert. Die dabei entstehenden Binärbilder werden in einem sogenannten Bitplanespeicher mit der Tiefe m abgelegt. Für jedes Pixel $p(x, y)$ im Bitplanespeicher kann nun eine m -stellige Bitfolge abgelesen werden, welche die Projektionsrichtung des Projektors zum korrespondierenden Objektpunkt eindeutig bestimmt. Sind die Position von Kamera und Projektor bekannt, lassen sich die Tiefenwerte der Objektpunkte mittels der Triangulierungsgesetze berechnen.

Die Oberfläche der gescannten Objekte muß nicht stetig sein, um korrekte Messungen zu erhalten. Wegen der begrenzten Auflösung wird diese Technik häufig als Grobmessung mit anschließender Feinmessung mittels des *Phasen-Schiebe-Verfahrens* eingesetzt.

2.2.3 Moiré-Technik

Diese Technik macht sich die aus der Wellenlehre bekannten Moiré-Muster zunutze, welche bei der Überlagerung regelmäßiger Strukturen entstehen. Diese Muster werden durch die Projektion von zwei periodischen Gittern auf die Oberfläche des zu vermessenden Objektes erzeugt. Es ist aber auch möglich, ein einzelnes Gitter zu projizieren und dieses durch ein zweites Gitter hindurch aufzunehmen. Dieses zweite Gitter ist bei Verwendung einer CCD-Kamera in Form ihres Pixel-Grids schon vorhanden. In beiden Fällen müssen die Gitter eine unterschiedliche Periodenlänge aufweisen (Klette et al., 1996). In Abbildung 2.7 ist der prinzipielle Meßaufbau für diese Technik veranschaulicht.

Ein Vorteil dieses Verfahrens ist die hohe erreichbare Meßgenauigkeit. Diese kann sich im Subpixel-Bereich bewegen, falls die projizierte Streifenbreite kleiner als die optische Streifenbreite ist (siehe Klette et al., 1996, S.359).

Es ist von Nachteil, daß mit diesem Verfahren nur relative Tiefenwerte für die Objektpunkte berechnet werden können. Diese Beschränkung kann durch Verwendung einer Referenzebene in der Szene oder durch Kombination mit der binärcodierten Lichtschnitttechnik umgangen werden (siehe Klette et al., 1996, S.358).

2.2.4 Farbcodierte Lichtschnitttechnik

Bei der farbcodierten Lichtschnitttechnik (siehe Klette et al., 1996, S.360-362) werden anders als bei der herkömmlichen Lichtschnitttechnik farbige Streifen verwendet. Das Farbmuster wird ausschließlich aus den Farben rot, grün, blau und weiß erzeugt, auf das Objekt projiziert und als Farbbild mittels einer Kamera aufgenommen. Die Farben können im Kamerabild leicht über ihre maximalen Intensitätswerte in den verschiedenen RGB-Farbkanälen identifiziert werden. Das setzt natürlich voraus, daß Objekte mit geringer Farbsättigung vermessen werden, da es sonst zu Farbverfälschungen kommen kann.

Wie in der statischen Streifenprojektion, besteht auch hier das Problem der Abschattung beziehungsweise Verdeckung einzelner Streifen des Musters. Es wird kein periodisches Muster sondern ein Muster aus einzelnen, kombinatorisch ermittelten Farbcodes verwendet, um die einzelnen Farbstreifen im Kamerabild eindeutig identifizieren zu können. Die einzelnen Codes haben die Beschränkung, daß benachbarte Streifen nicht die gleiche Farbe aufweisen dürfen. Die Anzahl M_0 der möglichen Codierungen kann durch nachfolgende Formel ermittelt werden:

$$M_0(K, L) = L \cdot (L - 1)^{K-1} \quad (2.5)$$

Dabei ist L die Anzahl der verwendeten Farben und K die Anzahl der Streifen je Farbcod. Das Farbmuster wird je nach Menge der zu projizierenden Streifen aus einer festen Anzahl von Farbcodes zusammengestellt.

Bei der Auswertung des Kamerabildes wird zeilenweise nach vollständigen Farbcodes gesucht. Unvollständige Farbcodes können durch die bekannte Position der erkannten Farbcodes im projizierten Muster richtig indiziert werden. Anschließend können mittels der Triangulierungsgesetze die Tiefenwerte der Objektpunkte berechnet werden. Ein Vorteil des Verfahrens ist, daß die Objektgeometrie schon mit einer einzigen Aufnahme bestimmt werden kann. Dies ist dadurch bedingt, daß sich durch die Verwendung von Farben mehr Zustände als bei der Binärcodierung darstellen lassen.

Nachteilig ist der relativ hohe Aufwand für die Identifikation der einzelnen Streifen des

Musters und die Abhängigkeit von den spektralen Reflektionseigenschaften der vermessenen Oberflächen.

2.2.5 Aktive Stereoanalyse mit Farbe

Genau wie bei der farbcodierte Lichtschnitttechnik wird bei diesem Verfahren ein Farbmuster auf das zu vermessende Objekt projiziert. Die Aufnahme des Objektmusters erfolgt jedoch unter Verwendung von zwei unterschiedlich platzierten Kameras als Stereobildaufnahme. Somit stellt dies eine Kombination aus Lichtpunktstereoanalyse und farbcodierter Lichtschnitttechnik dar.

Die Probleme mit Farbverfälschungen bei der Vermessung bunter Objekte, sind hier kaum von Bedeutung. Die verfälschten Farbstreifen erscheinen in beiden Kamerabildern und können einander zugeordnet werden. Eine Erweiterung des Farbcodes ist notwendig, um eine eindeutige Zuordnung der Bildpixel zu erreichen. Wie in Klette et al. (1996) beschrieben, existieren dafür zwei Ansätze. Dabei wird entweder ein Farbcode mit hohem Kontrast zwischen benachbarten Streifen (kontrastreiche Beleuchtung) oder ein Farbcode mit kontinuierlichen Übergängen zwischen benachbarten Streifen (kontinuierliche Beleuchtung) verwendet. Für die Zuordnung der Streifen der beiden Kamerabilder ist ein großer Kontrast zwischen den Streifen günstig. Es kann jedoch durch Tiefensprünge in der Szene, Abbildung von Streifen auf mehrere Pixel im Kamerabild und Fokussierungsprobleme bei dem Projektor zu Kontrastreduzierungen im aufgenommenen Streifenmuster kommen. Aufgrund dieser Probleme ist es ratsam, eine kontinuierliche Beleuchtung zu wählen.

Das zu erstellende Farbmuster kann, anders als bei der farbcodierten Lichtschnitttechnik, aus mehr als nur den drei Grundfarben und der Farbe weiß bestehen. Das Muster wird projiziert und mit den beiden Kameras aufgenommen. Anschließend werden die beiden Kamerabilder mittels einer Stereoanalyse untersucht und ausgewertet.

Die aktive Stereoanalyse hat verschiedene Vorteile. Sie ist unabhängig von den Objektfarben und setzt keine Kenntnis über den verwendeten Farbcode voraus. Da nur ein einziges Bildpaar aufgenommen werden muß, ist dieses Verfahren sehr schnell und auch bei dynamischen Szenen einsetzbar.

Ein Nachteil ist jedoch die im Vergleich zu Verfahren mit strukturierter Beleuchtung geringe erreichbare Tiefenauflösung.

2.3 Spezifik von 3D-Scandaten

Die Menge der gescannten Objektpunkte wird auch als Punktwolke bezeichnet. Diese ist die primäre und genaueste Repräsentation des gescannten Objektes. Trotzdem ist sie aufgrund

der teilweise riesigen Datenmengen und der hohen Redundanz wenig brauchbar. Da sie nur aus unzusammenhängenden Einzelpunkten besteht, ist sie auch schlecht visualisierbar. Die rechnerinterne Verwaltung solcher Punktwolken geschieht meist mit Hilfe von Punktlisten. Diese Listen enthalten die Tripel der Koordinaten der Punkte. Zusätzliche werden teilweise auch für jeden Punkt noch Farbwerte mitgeführt, welche aus den Kamerabildern extrahiert werden. Um die ungeordnete Punktwolke zu ordnen, wird meist ein Octtree verwendet.

Ein Octtree ist eine baumartige Struktur zur Indizierung eines dreidimensionalen Raumes. Jeder Knoten des Baumes besitzt genau eine einlaufende und acht auslaufende Kanten. Die Knoten und Blätter (Knoten ohne auslaufende Kanten) des Baumes repräsentieren einzelne Würfel, welche den dreidimensionalen Raum unterteilen. Diese Unterteilung ist hierarchisch und beginnt mit der Zerlegung des Raumes in acht einzelne Teilwürfel. Diese werden von den Knoten in der ersten Ebene des Baumes repräsentiert. Diese acht Octtree-Würfel werden wiederum in weitere acht Teilwürfel zerlegt, welche dann die zweite Ebene des Baumes bilden. Die Blätter des Baumes korrespondieren mit den kleinsten Würfeln des unterteilten Raumes. Die einzelnen Punkte der Punktwolke werden in die einzelnen Würfel einsortiert und damit in eine Ordnung gebracht.

Grundsätzlich kann bei der Digitalisierung von räumlichen Objekten davon ausgegangen werden, daß bei der Messung Fehler beziehungsweise Abweichungen entstehen. Diese Fehler können in zwei Kategorien eingeteilt werden. Zum einen sind das globale und zum anderen lokale Fehler.

Die globalen beziehungsweise systematischen Fehler betreffen das gesamte Meßsystem oder Meßverfahren. Diese führen zu systematischen Abweichungen während jeder ausgeführten Messung. Als Beispiel kann hier die Verzeichnung genannt werden, welche durch die Linse des Kamerasystems hervorgerufen wird. Dabei tritt ausgehend von einem Verzeichnungszentrum innerhalb des aufgenommenen Bildes eine radiale Verzerrung auf. Diese Verzerrung geht natürlich in die anschließende Berechnung der Tiefenkarte ein und führt so zu systematischen Abweichungen bei den berechneten Werten. Systematische Fehler können analysiert und mit Hilfe geeigneter Modelle und Ansätze minimiert werden.

Die lokal auftretenden hochfrequenten Fehler werden auch als Meßrauschen bezeichnet. Diese Fehler sind auf die Auflösungsbeschränkung des Meßsystems zurückzuführen. Man kann auch sagen, daß die Amplitude des Meßrauschens ein Maß für die Auflösung des Meßsystems darstellt. Das heißt, je kleiner die Amplitude des Rauschens, desto höher ist die Auflösung des Meßsystems. Die durch das Meßrauschen hervorgerufenen Fehler können bei der Messung durch Mittelung mehrerer Aufnahmen der Szene minimiert werden.

Kapitel 3

Polygonale Netze

In diesem Kapitel sollen die Geometrie und Topologie polygonaler Netze näher betrachtet werden. Dabei stehen besonders Dreiecksnetze und die zur Erzeugung dieser Dreiecksnetze verwendeten Triangulationsverfahren im Vordergrund.

3.1 Einsatz und Bedeutung

Flächige und räumliche Objekte können im Computer durch verschiedene geometrische Modelle repräsentiert werden. Dabei kann man zwischen Punktwolken, Flächenmodellen und Volumenmodellen unterscheiden.

Punktwolken bestehen lediglich aus einer Menge von Koordinaten, welche die Oberfläche des Objektes beschreibt, und gegebenenfalls Farbwerten, welche den einzelnen Punkten zugeordnet sind. Punktwolken werden meist durch Abtast- beziehungsweise Digitalisierungsverfahren, wie in Kapitel 2 beschrieben, gewonnen.

Volumenmodelle definieren den zu beschreibenden Körper über einfache, endliche Volumenelemente und können damit beliebig komplexe räumliche Objekte darstellen. Bekannte Vertreter der Volumenmodelle sind Sweep-Körper, CSG-Körper (Constructive Solid Geometry) sowie Voxel- beziehungsweise Octree-basierte Modelle.

Flächenmodelle approximieren die Oberfläche der darzustellenden Objekte durch beliebig viele Flächenelemente. Diese können eben, wie Dreieck, Viereck, Sechseck, Polyeder, oder gekrümmt, wie Splines, Bezier-Flächen oder NURBS (Nonuniform Rational B-Splines), sein. Die einzelnen Flächenelemente sind in sogenannte *polygonale Netze* eingebunden und bilden eine komplexe zusammengesetzte Oberfläche. Da Form und Anzahl der Elemente beliebig sind, ist theoretisch jede gewünschte Genauigkeit bei der Randrepräsentation eines Objektes denkbar. Flächenmodelle sind die häufigste Darstellungsform für die aus Abtastdaten beziehungsweise Punktwolken rekonstruierten Oberflächen von Objekten.

Dreiecksnetze, also polygonale Netze aus dreieckigen Flächenelementen, sind die am meisten verwendeten Flächenmodelle. Ihre Verarbeitung und Darstellung wird von einer Vielzahl von Grafikbibliotheken wie OpenGL und Direct3D sowie durch die Grafikprozessoren der meisten Grafikkarten direkt unterstützt. Dreiecksnetze werden rechnerintern meist durch Punkt- und Flächenlisten verwaltet. In der Punktliste werden alle Eckpunkte der Dreiecke mit ihren Koordinaten gespeichert. In der Flächenliste wird für jedes Dreieck des Dreiecksnetzes ein Tripel gespeichert, welches die Indizes der Eckpunkte dieses Dreiecks enthält. Durch ihre triviale Struktur sind polygonale Netze und insbesondere Dreiecksnetze sehr einfach manipulierbar. Ein Nachteil ist jedoch der Umfang an Daten, der sich bei großen und detaillierten Modellen sehr schnell negativ auf Verarbeitungs- und Darstellungsgeschwindigkeit auswirkt.

3.2 Topologie und Geometrie

Die Topologie eines Dreiecksnetzes beschreibt die Nachbarschaften der Eckpunkte der Dreiecke. Diese werden im Folgenden als *Vertex* beziehungsweise *Vertizes* bezeichnet. Die Nachbarschaften dieser Vertizes werden über die Dreieckskanten definiert. Die Verwendung und Manipulation von Dreiecksnetzen setzt voraus, daß zu jedem Zeitpunkt eine konsistente Netztopologie vorliegt. Eine Netz verfügt über eine konsistente Topologie, wenn folgende Bedingungen erfüllt sind:

- Jeder Punkt ist Eckpunkt von mindestens zwei Kanten.
- Jede Kante gehört zu mindestens einem und maximal zwei Dreiecken.
- Jede Fläche F_i teilt mindestens eine Kante mit einer anderen Fläche F_j .

Jede Operation, welche eine Veränderung der Topologie nach sich zieht, muß sicherstellen, daß die topologische Konsistenz erhalten bleibt. Ist dies nicht der Fall, so kann es bei der weiteren Verarbeitung zu Problemen kommen.

Betrachtet man die Topologie eines Dreiecksnetzes, so kann zwischen geschlossenen und nicht geschlossenen Netzen unterscheiden werden. Ein geschlossenes Dreiecksnetz zeichnet sich dadurch aus, daß es keinerlei Randdreiecke, Randkanten oder Randvertizes enthält. Diese sind jedoch bei nicht geschlossenen Netzen vorhanden. Ein Randdreieck besitzt mindestens eine Kante, welche nicht Bestandteil eines anderen Dreiecks ist. Diese wird als Randkante bezeichnet. Ein Randvertex ist Bestandteil einer Randkante.

Die Geometrie eines Dreiecksnetzes beschreibt die Position der Vertizes und somit die Lage der Dreiecke im Raum. Auch für die geometrischen Eigenschaften eines Netzes kann eine Konsistenz definiert werden. Ein Dreiecksnetz ist geometrisch konsistent, wenn es keinerlei Durchdringungen oder Überlappungen zwischen den Dreiecken des Netzes gibt. Die

Überprüfung eines gegebenen Dreiecksnetzes auf eine konsistente Geometrie ist sehr komplex. Diese kann nur global auf dem ganzen Netz erfolgen. Dazu muß jedes Dreieck auf Schnitte und Überlappungen mit allen anderen Dreiecken des Netzes getestet werden. Dies ist ein Problem mit einem quadratischen Laufzeitverhalten. Die Anzahl der Überprüfungen für einzelne Dreiecke steigt somit bei doppelter Dreieckszahl auf das vierfache.

Eine globale geometrische Eigenschaft eines polygonalen Netzes ist die Konvexität. Die folgende Definition kann verwendet werden, um eine Aussage über die Konvexität eines Polyeders, in diesem Fall ein einfach geschlossenes konsistentes Dreiecksnetz, treffen zu können.

Definition 3.1 (Konvexität)

*Ein Polyeder heißt **konvex**, wenn sämtliche Innenwinkel zwischen benachbarten Flächen (Flächenwinkel) kleiner als 180° sind.*

Die Winkel zwischen allen benachbarten Flächen müssen berechnet werden, um festzustellen, ob ein solches einfach geschlossenes Dreiecksnetz konvex ist. Gibt es mindestens einen Winkel zwischen zwei benachbarten Flächen, welcher größer als 180° ist, so ist das Dreiecksnetz nicht konvex sondern konkav.

Für Aussagen über die Qualität von Dreiecksnetzen werden abhängig von der Anwendung meist geometrische und topologische Eigenschaften als Kriterium herangezogen. Dabei sind unterschiedliche Qualitätsmerkmale, wie zum Beispiel die Größe der Innenwinkel der Dreiecke (siehe de Berg et al., 1997, S.183), die Anzahl der direkten Nachbarn der einzelnen Punkte oder die Länge der Dreieckskanten, gängig. Als optimal kann zum Beispiel ein Dreiecksnetz bezeichnet werden, welches nur aus Dreiecken mit möglichst großem minimalen Dreiecksinnenwinkel besteht. Ein optimales Dreieck wäre somit ein gleichseitiges Dreieck mit dem minimalen Dreiecksinnenwinkel von 60° .

3.3 Triangulationsverfahren

Für die Erzeugung eines Dreiecksnetzes aus einer beliebigen Menge von Punkten werden verschiedene Algorithmen verwendet. Diese werden auch als Triangulationsverfahren bezeichnet. In den folgenden Abschnitten sollen einige dieser Verfahren erläutert werden, welche auf unterschiedlichen Ansätzen und Herangehensweisen beruhen.

3.3.1 Delaunay-Verfahren

Das Delaunay-Triangulationsverfahren ist das bekannteste und weit verbreitetste Triangulationsverfahren. Es ist nach dem Mathematiker *Boris Nikolaevitch Delone* benannt (siehe de Berg et al., 1997, S.189).

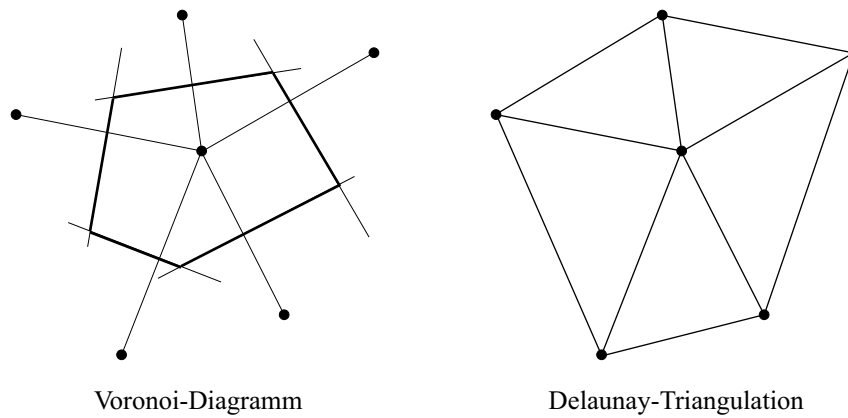


Abbildung 3.1: Voronoi-Diagramm und Delaunay-Triangulation

Das Delaunay-Verfahren wurde für die Triangulation von Punkten in der Ebene entwickelt. Es kann aber erweitert werden, so daß auch eine Anwendung für Punkte im dreidimensionalen Raum möglich ist. Das Verfahren soll hier für Punkte in der Ebene betrachtet werden. Eine wichtige Eigenschaft dieses Triangulationsverfahrens ist, daß es für eine beliebige Punktmenge in der Ebene eine optimale Triangulation erzeugt. Das heißt, die Delaunay-Triangulation erzeugt Dreiecke mit maximierten minimalen Innenwinkel (siehe de Berg et al., 1997, S.191).

Eine optimale Triangulation von Punkten in der Ebene hängt mit *Voronoi-Diagrammen* zusammen. Voronoi-Diagramme beschreiben die Aufteilung einer Ebene in einzelne Polygone. Folgende Definition beschreibt die genauen Bedingungen und Eigenschaften von Voronoi-Diagrammen.

Definition 3.2 (Voronoi-Diagramm)

Gegeben sei eine Menge $P = \{p_1 \dots p_n\}$ von n Punkten in der Ebene. Das Voronoi-Diagramm von P ist dann die Unterteilung der Ebene in n Polygone, eines für jeden Punkt in P , so daß innerhalb eines Polygons kein anderer Punkt der Menge P liegt.

Laut de Berg et al. (1997, S.149) folgt aus dieser Definition, daß ein beliebiger Punkt q der Ebene innerhalb eines bestimmten Voronoi-Polygons liegt, wenn der Abstand zwischen q und dem Zentrum p dieses Polygons kleiner als der Abstand zum Zentrum aller anderen Voronoi-Polygone ist.

Aus dem Voronoi-Diagramm kann eine Triangulation der Punkte der Ebene direkt erfolgen. Dazu werden die Zentren derjenigen Voronoi-Polygone verbunden, welche eine gemeinsame Grenze haben. Diese Verbindungen bilden dann die Kanten der Dreiecke, die erzeugt werden. Die dabei generierten Dreiecksnetze sind optimal im Sinne der Innenwinkel der Dreiecke und entsprechen dadurch einer Delaunay-Triangulation der Zentren der

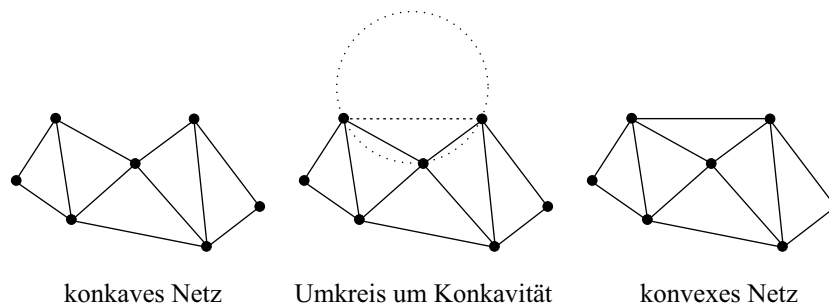


Abbildung 3.2: Schließen von Konkavitäten

Voronoi-Polygone. Aus dem Voronoi-Diagramm einer Punktmenge kann eine Delaunay-Triangulation derselben Punkte erzeugt werden. In Abbildung 3.1 ist das Voronoi-Diagramm sowie die daraus resultierende Delaunay-Triangulation einer Punktmenge, bestehend aus sechs Punkten, zu sehen.

Es muß jedoch nicht unbedingt das Voronoi-Diagramm der entsprechenden Punktwolke konstruiert werden, um eine Delaunay-Triangulation zu berechnen. Das Verfahren zur direkten Berechnung der Delaunay-Triangulation arbeitet mit Umkreisen um die zu triangulierenden Punkte. Im Folgenden soll nun beschrieben werden, wie die einzelnen Dreiecke berechnet werden.

Ein neues Dreieck wird jeweils ausgehend von einer Startkante aus 2 Punkten gebildet. Für das erste Dreieck muß ein geeignetes Punktpaar als Startkante ausgewählt werden. Das Paar besteht aus einem beliebigen Punkt und seinem geometrisch nächstgelegenen Nachbarn. Zu diesem wird nun der dritte Punkt mittels Umkreisen gesucht, um ein Dreieck zu bilden. Die Suche läuft über die gesamte Punktwolke, aus der jeweils ein Punkt ausgewählt wird. Um diesen und die beiden Endpunkte der Startkante wird ein Kreis derart konstruiert, daß alle drei Punkte auf dessen Rand liegen. Wenn innerhalb dieses Kreises kein anderer Punkt liegt, so kann ein neues Dreieck gebildet werden.

Die an das erste Dreieck anschließenden Dreiecke werden nun in gleicher Weise gebildet. Die Kanten des ersten Dreieckes dienen als Startkanten für die neuen Dreiecke. Wurde eine Kante zweimal als Startkante verwendet, wird sie als abgearbeitet markiert. Dies ist nötig, da jede Kante nur maximal von zwei Dreiecken referenziert werden darf, um topologische einwandfreie Netze zu erhalten (siehe Abschnitt 3.2). Dieser Schritt wird mit allen neu gebildeten Dreiecken wiederholt, bis die gesamte Punktwolke trianguliert ist.

Aus der Delaunay-Triangulation der Punktwolke kann auf sehr einfache Weise das Voronoi-Diagramm konstruiert werden. Dazu verbindet man die Mittelpunkte der einzelnen Umkreise der gebildeten Dreiecke. Diese Verbindungen bilden dann die Kanten der Voronoi-Polygone. Aus diesem Sachverhalt wird nochmals deutlich, daß es zwischen Voronoi-Dia-

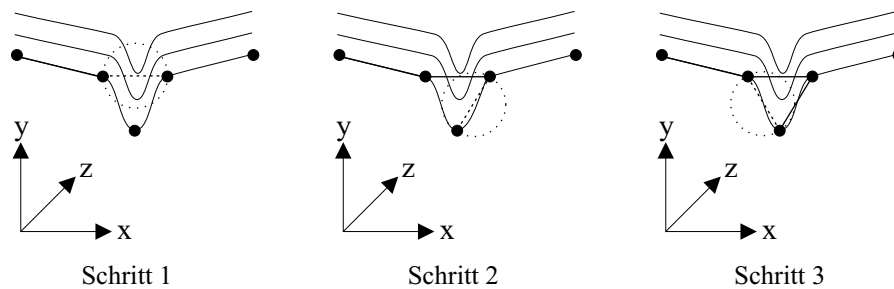


Abbildung 3.3: Delaunay-Triangulation im dreidimensionalen Raum

gramm und Delaunay-Triangulation einen sehr engen Zusammenhang gibt.

Eine weitere Eigenschaft der Delaunay-Triangulation ist die Entstehung von konvexen Netzen beziehungsweise Oberflächen. Da die Bildung neuer Dreiecke nur von dem Umkreis-kriterium abhängt, gibt es keine Möglichkeit Konkavitäten zu erhalten. Wie in Abbildung 3.2 zu sehen ist, werden diese immer durch neu gebildete Dreiecke geschlossen. Bei der Delaunay-Triangulation im zweidimensionalen Raum entstehen also immer geschlossene, konvexe und topologisch konsistente Dreiecksnetze.

An dem Algorithmus müssen einige Modifikationen vorgenommen werden, um die Delaunay-Triangulation auch für den dreidimensionalen Raum anwenden zu können. Als dreidimensionales Äquivalent für den Umkreis wird hier eine Kugel verwendet. Doch mit Hinzunahme der dritten Dimension entstehen Probleme, welche im zweidimensionalen Fall nicht vorhanden sind. So kann nicht mehr sichergestellt werden, daß bei der Triangulation der Punktwolke geschlossene, topologisch konsistente Netze erzeugt werden. Diese Problematik ist in Abbildung 3.3 dargestellt.

Die aus dem dreidimensionalen Raum stammenden Punkte und Dreiecke werden zur Anschaulichkeit in die x-y-Ebene projiziert. Die Oberfläche wird durch die drei im Bild sichtbaren Kurven angedeutet. Wie in der Abbildung zu sehen ist, weist die zu triangulierende Oberfläche einen Einschnitt auf. Die Methode zur Suche des dritten Punktes eines Dreiecks mit Hilfe einer Kugel führt hier zu Inkonsistenzen in der Topologie. Im ersten Schritt wird ein neues Dreieck gebildet, welches den Einschnitt in der Oberfläche überspannt. Im zweiten Schritt wird ein neues Dreieck erzeugt, welches auf der Innenseite des Einschnittes liegt. Im dritten Schritt kommt es dann zu Bildung eines Dreieckes, welches an eine Kante anschließt, die schon von zwei anderen Dreiecken referenziert wird. Damit ist die Topologie des Netzes nicht mehr konsistent. Der Algorithmus muß modifiziert werden, um solche Probleme befriedigend lösen zu können. Das bisherige rein lokale Kriterium für die Suche eines geeigneten dritten Punktes mittels einer Kugel ist für den Einsatz im dreidimensionalen Raum nicht ausreichend. Es muß also ein zusätzliches lokales oder globales Kriterium eingeführt werden. Dies könnte beispielsweise die Analyse der Oberflächenentwicklung

mittels des Fittens einer Ebene oder biquadratischen Fläche sein.

Die Suche nach einem geeigneten dritten Punkt zu einer Kante ist für große Punktwolken sehr rechenintensiv. Sie läuft immer über die gesamte Punktwolke, bis ein geeigneter Punkt gefunden wurde. Zur Beschleunigung dieser Suche muß ein Ansatz gefunden werden, um von einem globalen auf ein lokales Suchverfahren zu wechseln. Dies ist möglich, wenn der Raum der Punktwolke mit Hilfe eines Octtree unterteilt wird. Zu einem beliebigen Punkt der Punktwolke können dann die Punkte aus dem gleichen Würfel oder angrenzenden Würfeln im Octtree gefunden werden. Diese befinden sich auch in der Punktwolke in geometrischer Nachbarschaft zueinander. Damit kann also die Suche auf die unmittelbare Umgebung eines Punktes eingeschränkt und damit stark beschleunigt werden.

3.3.2 Marching-Cubes-Verfahren

Das Marching-Cubes-Verfahren ist ein Algorithmus zur Triangulation von sogenannten Isoflächen. Es ist also ursprünglich nicht zur Triangulation von Punktwolken entwickelt worden.

Die Voraussetzung zur Verwendung dieses Verfahrens ist eine Feldfunktion $F(x, y, z)$, welche im Raum definiert ist. Diese Funktion F muß für jeden Punkt im Raum einen Funktionswert besitzen. Die Gravitation eines Planeten kann als ein Beispiel für eine solche Feldfunktion dienen. An jedem Punkt im Raum kann der Wert der Gravitation ausgehend von diesem Planeten berechnet werden. Als Isofläche wird eine Fläche bezeichnet, auf deren Oberfläche die Funktionswerte der Funktion F konstant sind. Die Isofläche wird also durch alle jene Punkte im Raum definiert, für die die Feldfunktion F einen konstanten Wert C hat. Die Bedingungen für die Berechnung einer Isofläche sind demnach die Feldfunktion F und eine Schwelle oder Konstante C .

Beim Marching-Cubes-Verfahren wird der Raum in ein gleichmäßiges Raster aus einzelnen Würfeln unterteilt. Die Triangulation der Isofläche, welche durch eine Feldfunktion definiert ist, erfolgt durch Scannen des gesamten Raumes mit Hilfe von zwei übereinanderliegenden Schichten solcher Würfel. Während jeder Iteration wird für jeden dieser Würfel bestimmt, ob er Schnittpunkte mit der Isofläche besitzt. Dazu werden zuerst die Funktionswerte der Feldfunktion F für sämtliche Eckpunkte eines Würfels berechnet. Für jeden Eckpunkt ist nun ermittelt, auf welcher Seite von der Isofläche er sich befindet. Sind die beiden Eckpunkte einer Würfelkante auf verschiedenen Seiten der Isofläche, so steht fest, daß diese Kante von der Isofläche geschnitten wird. Nun kann der genaue Schnittpunkt der Isofläche mit dieser Kante berechnet werden. Anschließend müssen alle Schnittpunkte des Würfels mit der Isofläche verbunden werden, um Polygone beziehungsweise Dreiecke erzeugen zu können. Dabei können sieben verschiedene Fälle auftreten (siehe Abbildung 3.4 aus Wyvill et al., 1986, S.232). Ein Plus-Zeichen markiert alle Würfecken die sich

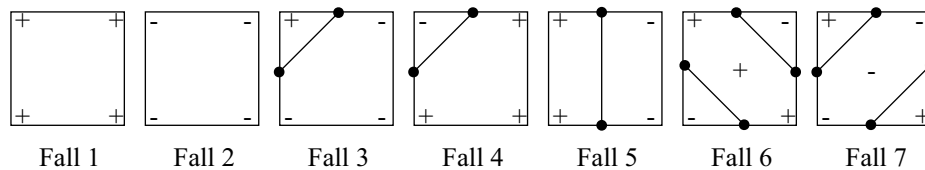


Abbildung 3.4: Sieben Fälle zum Verbinden von Schnittpunkten

innerhalb der Isofläche befinden, ein Minus-Zeichen alle Würfecken die sich außerhalb befinden. Werden die Schnittpunkte für alle Würfelseiten nach dem dargestellten Muster verknüpft, so entsteht ein Polygon. Dieses muß noch in geeigneter Weise in Dreiecke unterteilt werden.

Das Scannen des gesamten Raumes und das Testen aller Würfel des Rasters ist bei feiner Unterteilung sehr zeitintensiv, da die Feldfunktion jeweils für alle Eckpunkte eines Würfels berechnet werden muß. Die Anforderungen an die Rechenzeit steigen bei einer komplexen Feldfunktion weiter an. Die Anzahl der interessanten Würfel, welche von der Isofläche geschnitten werden, ist jedoch in Relation zur Gesamtzahl der Würfel klein. Ein Ansatz zur Beschleunigung des Verfahrens ist das Entlangwandern an der Isofläche. Zuerst muß ein Würfel gefunden werden, welcher von der Isofläche geschnitten wird. Von diesem Anfangswürfel ausgehend werden alle Nachbarn auf Schnitte mit der Isofläche geprüft. Wird einer der Nachbarn geschnitten, so werden wiederum dessen Nachbarn überprüft. Auf diese Weise wandert der Algorithmus auf der Isofläche entlang bis diese vollständig trianguliert ist.

Damit das Marching-Cubes-Verfahren zum Triangulieren einer Punktwolke verwendet werden kann, müssen einige Modifikationen vorgenommen werden. Das größte Problem dabei ist, aus der Punktwolke eine Feldfunktion abzuleiten, um die Isofläche berechnen zu können. Der Raum wird wieder in ein Raster aus Würfeln unterteilt. Dies kann durch die Verwendung eines Octtree erreicht werden. Die Punkte der Punktwolke werden in die jeweiligen Würfel des Octtree einsortiert. Die Isofläche, welche durch die Punktwolke definiert ist, wird lokal approximiert. Dazu wird an die Punkte innerhalb eines Würfels und gegebenenfalls innerhalb der Nachbarn dieses Würfels eine biquadratische Fläche gefittet. Diese Fläche stellt einen Ausschnitt aus der zu triangulierenden Isofläche dar. Für den gerade aktuellen Würfel wird der vorzeichenbehaftete Abstand zu dieser gefitteten Fläche berechnet. Das Vorzeichen zeigt an, auf welcher Seite der Isofläche sich der jeweilige Eckpunkt befindet. Nun werden nach dem oben beschriebenen Verfahren die Schnittpunkte der Würfelkanten mit der lokalen Isofläche berechnet. Anschließend werden die einzelnen Schnittpunkte verbunden und das so erhaltene Polygon in Dreiecke unterteilt.

Eine Voraussetzung für diese modifizierte Variante ist aber eine ausreichend dichte Punktwolke. Sonst kann es Bereiche geben, in denen nicht genügend Punkte zum Fitten einer

biquadratischen Fläche vorhanden sind. Solche Regionen finden sich häufig in den Randbereichen einer Punktwolke. Das hätte zu Folge, daß kein geschlossenes Dreiecksnetz erzeugt werden könnte.

3.3.3 Competitive Mesh Adaption

Competitive Mesh Adaption (CMA) ist ein in der Gesellschaft zu Förderung angewandter Informatik (GFaI) entwickeltes, adaptives Triangulationsverfahren (siehe Paul, 2000). Grundlage dieses Verfahrens ist die Adaption eines vorgefertigten Dreiecksnetzes mit konstanter Topologie an eine gegebene Punktwolke. Das bedeutet, daß sich nur die Geometrie des Dreiecksnetzes, also die Lage der Dreieckspunkte im Raum verändert. Die Topologie des Netzes, also die Nachbarschaftsbeziehungen zwischen den Dreieckspunkten, bleibt jedoch vollständig erhalten. Diese muß entsprechend der Form der Punktwolke gewählt werden, um sinnvolle Ergebnisse zu erzielen.

Die Adaption des Dreiecksnetzes läuft in einem oder mehreren Iterationsschritten ab. In jedem Durchgang wird für jeden Punkt des Dreiecksnetzes ein Verschiebungsvektor berechnet. Am Ende werden dann die Positionen aller Punkte mit Hilfe der berechneten Verschiebungsvektoren angepaßt. Das gesamte Dreiecksnetz wird also immer in einem Schritt verändert. Eine Iteration läuft wiederum in verschiedenen Teilschritten ab, welche im Folgenden erläutert werden.

Zuerst wird für jeden Vertex des Dreiecksnetzes ein korrespondierender Punkt in der Punktwolke gesucht. Dieser Korrespondenzpunkt ist der Punkt mit dem geringsten geometrischen Abstand zu diesem Vertex. Jeder Vertex und die Vertizes seiner vorher festgelegten topologischen Umgebung werden nun in Richtung des Korrespondenzpunktes verschoben. Dabei wird der jeweils aktuelle Vertex am stärksten und die Vertizes seiner topologischen Nachbarschaft in Abhängigkeit vom Grad ihrer Nachbarschaft verschoben. Der Grad der Nachbarschaft läßt sich erklären, wenn man das Dreiecksnetz mit seinen Vertizes und Kanten als Graph betrachtet. Der Abstand zweier Vertizes A und B wird als die Anzahl der Kanten des minimalen Weges zwischen A und B definiert. Der Nachbarschaftsgrad zwischen zwei Vertizes ist dann dem graphentheoretischen Abstand gleichzusetzen.

Nun wird für jeden Punkt der Punktwolke ein korrespondierender Vertex gesucht. Dieser ist der Vertex mit dem geringsten geometrischen Abstand zu diesem Punkt der Punktwolke. Der Vertex und die Vertizes seiner topologischen Umgebung werden nun in Richtung des Punktes der Punktwolke verschoben. Wie im vorherigen Schritt wird die Verschiebung der Vertizes in Abhängigkeit ihres Nachbarschaftsgrades berechnet.

Für die Suche der korrespondierenden Punkte der Punktwolke beziehungsweise Vertizes des Dreiecksnetzes werden zwei Octtree verwendet. Dazu müssen alle Punkte der Punktwolke sowie alle Vertizes des Dreiecksnetzes in jeweils einen Octtree einsortiert werden. Die Pa-

parameter der beiden Octtree müssen dabei identisch sein, damit die einzelnen Würfel eines Octtree die gleiche Raumposition wie die korrespondierenden Würfel des anderen Octtree einnehmen. Da sich die Positionen der Vertizes nach jeder Iteration ändert, muß der Octtree für die Vertizes jedes Mal neu aufgebaut werden.

Innerhalb jeder Iteration kann ein Ausgleich berechnet werden, um die Geometrie des Dreiecksnetzes zu verbessern. Dieser Geometrieausgleich erzeugt für jeden Vertex einen Korrekturvektor, welcher nach der Verschiebung des Vertex seine Position nochmals anpaßt. Für die Berechnung des Korrekturvektors stehen zwei Möglichkeiten zur Verfügung. Das erste der beiden Verfahren versucht, die von dem jeweiligen Vertex ausgehenden Kantenlängen auszugleichen. Letztendlich wird der Vertex in Richtung seines am weitesten entfernten direkten topologischen Nachbarn verschoben. Das geschieht unterschiedlich stark und ist abhängig von der Relation zwischen der mittleren und der maximalen Kantenlänge. Folgende Formel wird für die Verschiebungsstärke k verwendet:

$$k = 0.9 - \frac{\text{mittlere Kantenlänge}}{\text{maximale Kantenlänge}}$$

Auch das zweite Verfahren zur Berechnung des Korrekturvektors versucht, die Längen der vom jeweiligen Vertex ausgehenden Kanten auszugleichen. Hier wird der Vertex jedoch in Richtung des Zentrums seiner direkten topologischen Nachbarn gezogen.

Zusätzlich zu ausgeglichenen Dreiecken wird bei Dreiecksnetzen auch noch eine gleichmäßige Verteilung der Dreiecke innerhalb des Netzes angestrebt. Dieses Kriterium kann durch einen sogenannten *Gleitfilter* verbessert werden. Der Gleitfilter arbeitet global auf dem gesamten Dreiecksnetz. Durch seine Anwendung gleiten die Dreieckspunkte auf der Oberfläche entlang, welche somit größtenteils erhalten bleibt. Im ersten Schritt wird der jeweilige Vertex ins Zentrum seiner direkten topologischen Nachbarn verschoben. Dies kann jedoch die Oberfläche des Dreiecksnetzes beispielsweise an scharfen Ecken oder Kanten sehr stark verändern. Deshalb wird im zweiten Schritt der verschobene Vertex zurück auf die ursprüngliche Oberfläche projiziert. Dazu werden alle anliegenden Dreiecke überprüft, indem das Lot des Vertex auf die jeweilige Fläche berechnet wird. Der Vertex wird dann auf das Dreieck projiziert, zum dem das Lot beziehungsweise der Abstand am geringsten ist.

Wie oben erläutert, arbeitet das CMA-Verfahren mit konstanter Topologie. Das heißt, es werden keine neuen Punkte, Kanten oder Dreiecke erzeugt. Es soll jedoch eine beliebig genaue Adaption an die Punktwolke erreicht werden. Wenn die Anzahl der Dreiecke im vorgefertigten Dreiecksnetz zu gering ist, können teilweise die Dreiecke zu groß sein, um feine Details in der zu triangulierenden Oberfläche darzustellen. Für diesen Fall kann das Dreiecksnetz global verfeinert werden. Dies geschieht durch die Aufteilung jedes Dreieckes des Netzes in mehrere andere Dreiecke. In Abbildung 3.5 sind verschiedene Arten der Unterteilung dargestellt.

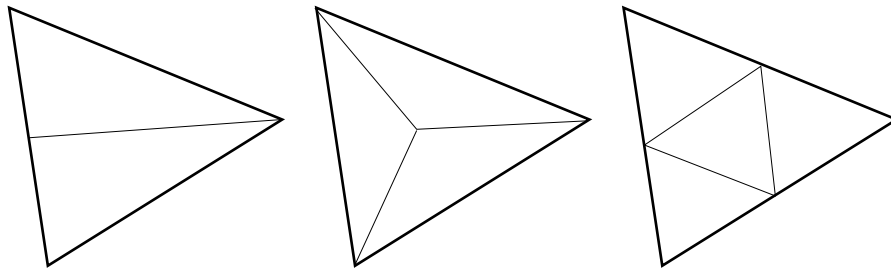


Abbildung 3.5: Dreiecksverfeinerung

3.3.4 Bewertung der Verfahren

In diesem Abschnitt sollen die drei vorgestellten Triangulationsverfahren anhand verschiedener Kriterien verglichen und bewertet werden. Die hier betrachteten Kriterien sind Topologie und Geometrie der generierten Dreiecksnetze, sowie der Rechenaufwand der Algorithmen.

3.3.4.1 Topologie

Die Topologie eines Dreiecksnetzes (siehe Abschnitt 3.2) ist das wichtigste Kriterium bei der Bewertung von Triangulationsverfahren. Das Ziel jedes Triangulationsverfahrens sollte die Erzeugung topologisch konsistenter Dreiecksnetze sein.

Das Delaunay-Verfahren liefert für den zweidimensionalen Raum topologisch einwandfreie und konsistente Netze. Für den Aufbau der Topologie werden ausschließlich lokale Information verwendet. Die Dreiecksnetze haben weiterhin die Eigenschaft, daß sie konvex und geschlossen sind. Die Erweiterung des Delaunay-Verfahrens für den dreidimensionalen Raum erzeugt jedoch nicht zwangsläufig topologisch konsistente Netze. Dieses Kriterium kann für dieses Verfahren nicht lokal erfüllt werden. Es sind also noch Erweiterungen notwendig, um dieses Kriterium über globale Informationen sicherstellen zu können. Diese könnten beispielsweise Informationen über den Krümmungsverlauf der zu triangulierenden Oberfläche sein.

Das Marching-Cubes-Verfahren generiert topologisch konsistente Netze, die aber nicht zwangsläufig geschlossen sind. Problematisch sind Bereiche in der Punktwolke, welche eine relativ geringe Punktdichte aufweisen. Die Anzahl der Punkte reicht dort nicht aus, um die Isofläche lokal zu approximieren. Dadurch werden in solchen Bereichen mitunter keine Dreiecke gebildet. Dies führt dazu, daß Löcher im erzeugten Dreiecksnetz entstehen.

Das CMA-Verfahren erzeugt in jedem Fall topologisch konsistente geschlossene Netze. Das Verfahren geht von einem konsistenten vorgefertigten Dreiecksnetz aus. Es verändert nur die Geometrie des Dreiecksnetzes und läßt die Topologie und Geschlossenheit des Netzes

unverändert. Die Verfeinerung des Dreiecksnetzes ist die einzige Möglichkeit die Topologie des Initialnetzes zu ändern. Diese wird aber ausschließlich über eine Unterteilung der schon vorhandenen Dreiecke umgesetzt. Dadurch kann sichergestellt werden, daß sich das Netz auch nach der Verfeinerung noch in einem konsistenten Zustand befindet. Eine objektangepaßte Modifikation der Topologie ist jedoch bei diesem Verfahren nicht möglich.

3.3.4.2 Geometrie

Die Geometrie eines Dreiecksnetzes ist ein wichtiges Kriterium, um etwas über dessen Genauigkeit und Qualität auszusagen. In erster Linie sollte das Netz die triangulierte Punktwolke möglichst genau abbilden. Zusätzlich spielt die Form der Dreiecke eine Rolle. Ein geometrisch optimales Dreiecksnetz enthält Dreiecke, bei denen der minimale Dreiecksinnenwinkel möglichst groß ist (siehe Abschnitt 3.2).

Das Delaunay-Verfahren erzeugt eine optimale geometrische Triangulation einer Punktwolke. Die generierten Dreiecke haben einen maximierten minimalen Dreiecksinnenwinkel. Bei diesem Verfahren wird zudem jeder Punkt der Punktwolke zu einem Vertex im Dreiecksnetz. Das bedeutet, daß je nach Dichte der Punktwolke die Größe und Anzahl der Dreiecke in Teilbereichen des Netzes steigt oder fällt. Dadurch wird die höchstmögliche Genauigkeit bei der Modellierung der Punktwolke erreicht. Soll das gebildete Dreiecksnetz aus Dreiecken gleicher Größe bestehen, so muß die Punktwolke eine gleichmäßige Dichte aufweisen. Die Dreiecksgröße ist somit nicht direkt steuerbar.

Das Marching-Cubes-Verfahren generiert geometrisch ausgewogene Dreiecksnetze. Das Verfahren verwendet eine gleichmäßige Aufteilung des Raumes in ein Raster aus Würfeln, um Dreiecke zu erzeugen. Dadurch ist die Dreiecksgröße der Netze sehr ausgeglichen. Die Größe der Dreiecke ist direkt über die gewählte Rastergröße steuerbar. Je größer das Raster, desto größer sind die gebildeten Dreiecke. Mit größerem Raster kann aber keine genaue Triangulierung der Punktwolke erreicht werden. Ist das Raster zu klein gewählt, so kann es dagegen zur verstärkten Bildung von Löchern im Dreiecksnetz kommen. Hier müssen geeignete Werte gefunden werden, um einerseits eine geschlossene Oberfläche und andererseits eine möglichst genaue Triangulierung der Punktwolke zu erreichen.

Das CMA-Verfahren kann ebenfalls geometrisch sehr ausgeglichene Netze erzeugen. Dies kann durch den Geometrieausgleich während der Adaption beziehungsweise den Gleitfilter sichergestellt werden. Dadurch wird eine gleichmäßige Verteilung der Dreiecke auf der Oberfläche sowie eine günstige Form der einzelnen Dreiecke erreicht. Ihre Größe ist durch eine mögliche globale Verfeinerung während der Adaption steuerbar. So kann schon während der Anpassung der Grad der Genauigkeit der Triangulation erhöht werden, allerdings auf Kosten der Rechenzeit.

3.3.4.3 Rechenaufwand

Mit der Verfügbarkeit von leistungsfähigen Computern tritt die Rechenzeit von Algorithmen immer mehr in den Hintergrund. Trotzdem soll hier kurz der Rechenaufwand der einzelnen vorgestellten Triangulationsverfahren betrachtet werden.

Das Delaunay-Verfahren ist sehr rechenzeitintensiv. Für eine Startkante eines neuen Dreiecks müssen immer alle Punkte der Punktwolke überprüft werden. Dies ist ein Problem mit einem quadratischen Laufzeitverhalten und führt bei großen Punktwolken zu langen Rechenzeiten. Diese sehr aufwendige Suche kann jedoch reduziert werden. Für die Delaunay-Triangulation im dreidimensionalen Raum wäre die Verwendung eines Octtree denkbar. Die Suche nach geeigneten Punkten zu einer Startkante könnte so auf die Nachbarschaft der Octtree-Würfel beschränkt und damit stark beschleunigt werden.

Das Marching-Cubes-Verfahren ist ein relativ schnelles Verfahren. Wie in der Beschreibung des Verfahrens in Abschnitt 3.3.2 schon erwähnt wurde, kann es durch Entlangwandern an der Isofläche beschleunigt werden. Damit werden nur die Würfel des Rasters untersucht, die auch wirklich im Schnittbereich der Isofläche liegen. Alle anderen, uninteressanten Würfel des Rasters können bei der Berechnung ausgelassen werden. Bei der Modifikation des Verfahrens für Punktwolken entsteht ein weiterer zeitintensiver Schritt: das lokale Fitten der parametrischen Fläche an die Punktwolke, um die Isofläche lokal zu approximieren. Da dieser Schritt für jeden Würfel an der Isofläche durchgeführt werden muß, ergeben sich wiederum längere Rechenzeiten. Für die Berechnung der parametrischen Fläche sollte die Anzahl der verwendeten Punkte klein gehalten werden, um den Rechenaufwand gering zu halten. Diese Begrenzung sollte jedoch mit Vorsicht benutzt werden. Die Anzahl der Punkte, welche in den Fit der parametrischen Fläche eingehen, darf nicht beliebig gesenkt werden, um verwertbare Ergebnisse zu erzielen.

Das CMA-Verfahren ist ein sehr rechenintensives Triangulationsverfahren. Die Suche nach dem nächsten Punkt zu einem Vertex und die Suche nach dem nächsten Vertex zu einem Punkt wird schon über die Verwendung zweier Octtree beschleunigt. Für die Punktwolke stellt dies einen großen Vorteil dar, da sie sich im Verlauf der Adaption nicht in ihrer Geometrie verändert. Das Einsortieren in den Octtree muß also nur einmal ausgeführt werden. Die Vertizes des Dreiecksnetzes sind jedoch variabel. Ihre Position wird nach jedem Adaptionsschritt verändert. Das hat zur Folge, daß sich auch ihre Position im Octtree verändert. Nach jedem Adaptionsschritt muß der Octtree für das Dreiecksnetz neu aufgebaut werden. Dieser Schritt beeinflußt die Laufzeit des Verfahrens stark negativ.

Kapitel 4

Selbstorganisierte Systeme

4.1 Selbstorganisierte Systeme in der Natur

Selbstorganisation ist ein wesentlicher Bestandteil in der Natur und für das Leben an sich. Es gibt viele verschiedene biologische, chemische und physikalische Systeme, welche den Prinzipien der Selbstorganisation unterliegen. Alle Systemen haben die Gemeinsamkeit, daß sie eine gewisse Ordnung beziehungsweise einen stabilen Zustand herausbilden. Die aus vielen autonomen Entitäten bestehenden Systeme können damit einen Zustand höherer Ordnung erreichen.

Die wahrscheinlich populärsten Beispiele für Selbstorganisation sind die Insektenstaaten, seien es Ameisen- oder Bienenvölker. Diese bestehen aus tausenden autonomen Insekten, die zum Funktionieren des Gesamtsystems beitragen. Jedes einzelne Insekt hat eine spezifische Aufgabe, die aber ausschließlich auf das Gesamtziel des Staates ausgerichtet ist. Bei diesen Insektenstaaten bildet sich eine natürliche Ordnung. Diese wird durch eine Hierarchie gestützt, die aus Königin, Soldaten und Arbeitern besteht. Jedes einzelne Insekt arbeitet dabei mit Hilfe seiner vorgeprägten Verhaltensmuster und kommuniziert mit einigen anderen Insekten des Staates. Mittels dieser zwischen den Insekten stattfindenden Kommunikation werden die komplexeren geordneten Verhaltensmuster des Insektenstaates erst ermöglicht.

Ein Beispiel für Selbstorganisation aus der Physik ist die *Rayleigh-Benárd-Instabilität* (siehe Coveney and Highfield, 1992, S.242-243). Bei diesem Experiment wird eine Flüssigkeit (zum Beispiel Silikonöl) auf eine Glasplatte gegeben und diese auf eine Wärmequelle gestellt. Wenn dem System Wärmeenergie von unten zugeführt wird, kommt es zu einer Temperaturdifferenz zwischen der unteren und der oberen Grenze der Flüssigkeit. Diese Differenz kann durch Wärmeleitung oder Konvektion ausgeglichen werden. Unterhalb einer bestimmten Schwelle verhält sich die Flüssigkeit makroskopisch gesehen ruhig und die Wärme wird ausschließlich durch Wärmeleitung transportiert. Wird diese jedoch über-

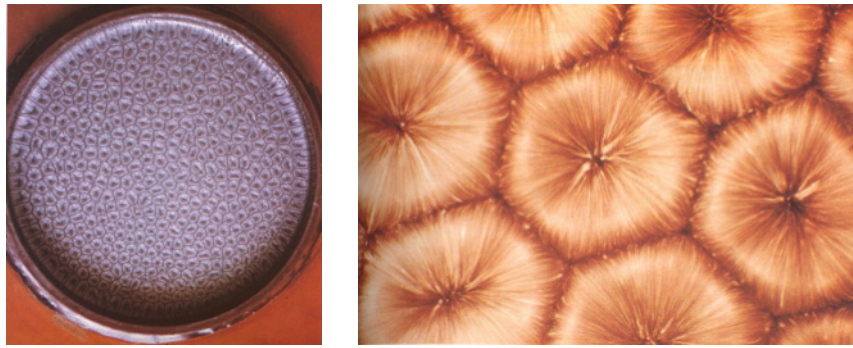


Abbildung 4.1: Gesamtansicht und Nahaufnahme von Konvektionszellen, entnommen aus Coveney and Highfield (1992, S.256ff)

schritten, dann setzt die Konvektion ein. Dabei steigen wärmere Moleküle am Grund der Flüssigkeit durch kühlere Flüssigkeitsschichten nach oben. Durch diese aufsteigenden Molekülströme entstehen bienenwabenförmige Muster innerhalb der Flüssigkeit (siehe Abbildung 4.1). Aufgrund der stärkeren Energiezufuhr wäre eigentlich zu erwarten, daß sich die ungeordnete Molekülbewegung verstärkt und einen immer chaotischeren Zustand erzeugt. Es bildet sich jedoch ein stabiler Zustand höherer Ordnung heraus, welcher solange bestehen bleibt, wie die Temperaturdifferenz erhalten wird. Es muß sich eine sehr große Anzahl von Molekülen in ihrer Bewegung angleichen, um die im Vergleich zu den einzelnen Molekülen riesigen Strukturen zu bilden. Die Moleküle dieses Systems sind nicht länger eine sich chaotisch bewegend Masse, sondern haben sich selbst organisiert.

Auch in der Chemie gibt es Beispiele für spontane Selbstorganisation. Solche sich selbst organisierenden chemischen Systeme werden auch *chemische Uhren* genannt. Die wohl bekannteste chemische Uhr ist die von *Boris Pawlowitsch Belusow* entdeckte und erforschte *Belusow-Shabotinski-Reaktion* (siehe Coveney and Highfield, 1992, S.257-262). Bei dieser Reaktion werden Zitronensäure, Kalziumbromat, schweflige Säure und Cer-Ionen miteinander vermischt. Diese Lösung kann in Korrespondenz zu den zwei verschiedenen Ladungszuständen der Cer-Ionen oszillieren. Dabei wechselt sie regelmäßig wie eine Uhr (daher auch der Name chemische Uhr) zwischen einem farblosen und einem kräftig gelben Zustand. Die Moleküle, die bei dieser Reaktion entstehen, bilden selbstständig geordnete Strukturen und Muster. Sie wechseln zwischen zwei stabilen Zuständen höherer Ordnung. Heute sind viele chemische Uhren bekannt, welche die gleichen Eigenschaften aufweisen. Es gibt mehrere Bedingungen, die erfüllt sein müssen, damit sich ein chemisches System in einen Zustand der Selbstorganisation begibt. Erstens sind diese Systeme fern von einem möglichen Gleichgewicht zu halten. Zweitens sollte ein Stoff, der an der Reaktion beteiligt ist, seine eigene Herstellungsrate beeinflussen können. Und drittens muß das System bistabil sein. Das heißt, es kann unter den gleichen Bedingungen in zwei verschiedenen stabilen

Zuständen vorliegen. Unter diesen Bedingungen können chemische Uhren in regelmäßigen Zeitabständen geordnete Muster und Strukturen durch Selbstorganisation ihrer Moleküle erzeugen.

Die biologischen neuronalen Netze sind ein weiteres Beispiel für biologische selbstorganisierte Systeme. Ihre nähere Beschreibung erfolgt im nächsten Abschnitt.

4.2 Biologische neuronale Netze

Die Erforschung und Entwicklung von künstlichen neuronalen Netzen wird von der Existenz und Funktionsweise der biologischen neuronalen Netze sehr stark beeinflusst. In diesem Abschnitt sollen die wichtigsten Eigenschaften und Merkmale der biologischen neuronalen Netze beschrieben werden. Im Folgenden wird für den Begriff neuronales Netz die Abkürzung NN verwendet.

Biologische NN bestehen aus Nervenzellen, auch Neuronen genannt, und Verbindungen zwischen den Nervenzellen. Das menschliche Gehirn besteht aus 10^{10} bis 10^{11} Nervenzellen. Dabei ist jede dieser Zellen durchschnittlich mit 10^3 bis 10^4 weiteren Nervenzellen verbunden (siehe Rigoll, 1994, S.31).

Ein Neuron besteht aus drei Hauptbestandteilen, dem Zellkörper mit Zellkern (Soma), dem Axon und den Dendriten. Das Axon stellt eine von der Nervenzelle wegführende Verbindung dar, wogegen die Dendriten zur Nervenzelle hinführende Verbindungen sind. Diesen Bestandteilen der Nervenzelle können verschiedene Aufgaben zugeordnet werden. Dabei dienen die Dendriten der Aufnahme, der Zellkern der Verarbeitung und das Axon der Weiterleitung der Informationen bei der Informationsverarbeitung. Ein weiterer Bestandteil sind die Synapsen, welche die Verbindungen zwischen den verschiedenen Neuronen ermöglichen. Die Synapsen können sich an den Dendriten oder direkt auf dem Zellkörper befinden. Es wird zwischen erregenden (exzitatorischen) und hemmenden (inhibitorischen) Synapsen unterschieden.

Die Kommunikation der Nervenzellen untereinander erfolgt über elektrische Potentiale. Die stark verästelten Dendriten sind über die Synapsen mit den Axonen anderer Neuronen verbunden. Von dort empfangen sie Ausgabesignale der verbundenen Nervenzellen in Form von elektrischen Potentialen, welche sie dem Zellkern zuleiten. Dort werden diese Potentiale aufsummiert. Wenn das Potential im Zellkern einen gewissen Schwellwert überschreitet, so erzeugt dieser einen kurzzeitigen elektrischen Potentialimpuls (Spike). Dieser Impuls wird über das Axon vom Zellkern weggeleitet. Das Axon kann Millimeterbruchteile bis einige Meter lang sein und ist genau wie die Dendriten sehr stark verästelt. Somit kann das vom Zellkern gebildete elektrische Potential an bis zu mehrere tausend verbundene Zielneuronen weitergeleitet werden (siehe Ritter et al., 1991, S.18-24).

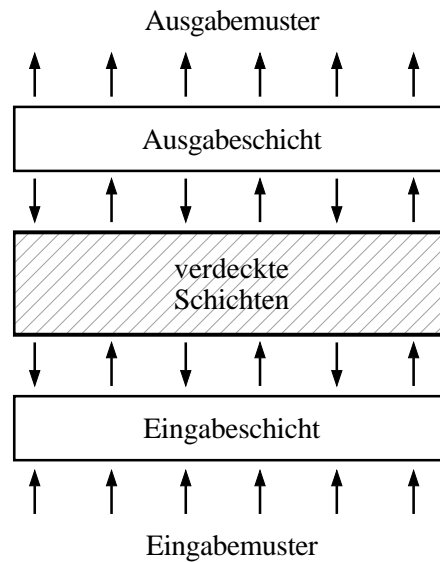


Abbildung 4.2: Genereller Aufbau eines künstlichen neuronalen Netzes

Für detailliertere Informationen über die Erzeugung und Weiterleitung der Impulse sei der Leser auf die weiterführende Literatur wie Rigoll (1994); Ritter et al. (1991); Schöneburg et al. (1990); Brause (1991) verwiesen.

4.3 Künstliche neuronale Netze

Wie bereits im Abschnitt 4.2 erwähnt, ist der Aufbau und die Funktionsweise künstlicher neuronaler Netze den biologischen neuronalen Netzen nachempfunden.

Ein künstliches NN besteht aus einer Menge $C = \{c_1, \dots, c_n\}$ von Verarbeitungseinheiten (Neuronen, Prozessorelemente, Einheiten, Zellen) sowie Verbindungen zwischen den Verarbeitungseinheiten. Bei diesen Verarbeitungseinheiten (im folgenden als Zellen bezeichnet) kann zwischen Eingabezellen, Ausgabezellen und internen oder verborgenen Zellen unterschieden werden. Die Eingabezellen sind mit einem externen Reiz oder Sensor verbunden, welcher Eingabesignale erzeugt. Die Eingabesignale werden durch einen n -dimensionalen Eingabevektor x definiert. Die Ausgabezellen erzeugen ein Ausgabesignal, welches durch einen m -dimensionalen Ausgabevektor y definiert ist.

Die Zellen des künstlichen NN sind in verschiedenen funktionalen Schichten angeordnet. Diese werden in die Eingabeschicht mit den Eingabezellen, die Ausgabeschicht mit den Ausgabezellen und die verdeckte Schicht mit den internen Zellen unterteilt. Dieser generelle Aufbau (siehe Abbildung 4.2) kann teilweise in abgewandelter Form auftreten, wobei

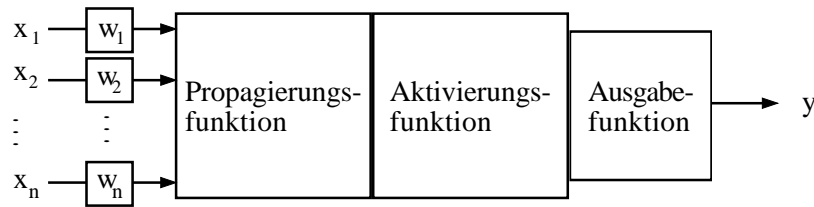


Abbildung 4.3: Aufbau eines künstlichen Neurons (Zelle)

zum Beispiel die verdeckte Schicht ganz entfallen kann, oder Eingabe- und Ausgabeschicht identisch sind. Es gibt auch NN, in denen die verdeckte Schicht mehrere Unterschichten enthält. Ein künstliches NN hat mindestens eine Schicht, welche wiederum mindestens eine Zelle umfassen muß. Die Zellen einer Schicht können mit beliebig vielen Zellen aus anderen Schichten verbunden sein.

4.3.1 Informationsverarbeitung eines Neurons

Genau wie sein biologisches Vorbild besitzt ein künstliches Neuron (Zelle) mindestens einen Eingang, der zur Informationsaufnahme dient. Die Zellen der Eingabeschicht sind direkt mit dem Eingabemuster verbunden, wogegen alle anderen Zellen ihre Informationen von den Zellen der darüberliegenden Schicht beziehen. Aus den Eingabesignalen erzeugt das einzelne Neuron nach einer mathematischen Vorschrift ein Ausgabesignal, welches über den Ausgang weitergeleitet wird. Diese mathematische Vorschrift wird durch einige Parameter beeinflusst, welche hauptsächlich durch reellwertige Gewichtungen w der Zelleingänge gegeben sind. Das Gewicht der Verbindung zwischen dem Ausgang der Zelle j und dem Eingang der Zelle i wird mit w_{ij} gekennzeichnet. Bei den Gewichten unterscheidet man zwischen festen und variablen Gewichten. Anders als feste Gewichtungen können die variablen Gewichtungen im Laufe des Lernprozesses verändert werden.

Die Verarbeitung der Informationen in einer einzelnen Zelle kann in drei Schritte unterteilt werden (siehe Abbildung 4.3).

Im ersten Schritt kommt die Propagierungsfunktion $G(x)$ zum Einsatz. Diese Funktion liefert meist die Summe net der gewichteten Eingabesignale zurück. Dabei werden die Eingabesignale x_j mit den zugehörigen Gewichten w_j multipliziert, was dem Skalarprodukt des Eingabevektors x und des Gewichtsvektors w entspricht.

$$G(x) = \sum_{j=0}^n x_j \cdot w_j = x \cdot w = net \quad (4.1)$$

Denkbar sind jedoch auch Multiplikationsfunktionen (Kreuzprodukt) oder Abstandsfunk-

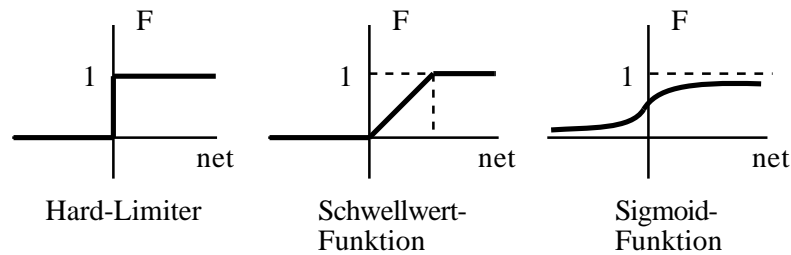


Abbildung 4.4: Verschiedene Aktivierungsfunktionen

tionen (euklidischer Abstand, Hamming-Abstand oder Vektornorm) zwischen Eingabevektor x und Gewichtsvektor w der Zellen.

Im zweiten Schritt wird das Ergebnis der Propagierungsfunktion an die sogenannte Aktivierungsfunktion $F(G)$ weitergeleitet. Die Aktivierungsfunktion berechnet die Aktivität einer Zelle in Abhängigkeit von deren aktuellen Eingabesignalen. Die Aktivierungsfunktion kann linear oder nichtlinear sein. Bei der linearen Variante entspricht die Aktivität a der Zelle dem Ergebnis net der Propagierungsfunktion.

$$F(G) = F(net) = a \quad (4.2)$$

Die am häufigsten verwendeten Aktivierungsfunktionen sind der sogenannte Hard-Limiter, die Schwellwertfunktion sowie die Sigmoid-Funktion (siehe Rigoll, 1994, S.44-45), welche in Abbildung 4.4 dargestellt sind. Da der Hard-Limiter nur die beiden Funktionswerte 0 und 1 zulässt, wird er fast ausschließlich bei binären NN verwendet. Diese Netze verarbeiten binäre Eingabesignale und erzeugen binäre Ausgabesignale. Die Sigmoid-Funktion liefert dagegen kontinuierliche Funktionswerte zwischen 0 und 1 und wird deswegen bei NN mit reellwertigen Ausgabewerten eingesetzt.

Im dritten Schritt, der Ausgabefunktion, wird der Wettbewerb zwischen den Zellen einer Netzschicht ermöglicht. Dabei wird durch die Ausgabefunktion geregelt, wie die Ausgabe zwischen den verschiedenen Zellen aufgeteilt wird. Die häufigsten Ausgabefunktionen sind die direkte und die *winner-takes-all*-Methode (siehe Schöneburg et al., 1990, S.51). Bei der direkten Methode ist die Ausgabe y der Zellen gleich ihrer Aktivierung a . Bei der *winner-takes-all*-Methode, einer Form des konkurrierenden Lernens, gewinnt das Neuron mit der höchsten Aktivität $\max(a_i)$ den Wettbewerb. Möglich ist auch, dass mehrere Zellen gewinnen und sich die Ausgabe nach einer definierten Regel untereinander aufteilen.

4.3.2 Informationsverarbeitung eines neuronalen Netzes

Die Informationen, die von künstlichen NN verarbeitet werden, können Signale, Bitmuster oder reelle Zahlenwerte sein.

Die Eingabesignale, auch als Eingabemuster bezeichnet, werden den Zellen in der Eingabeschicht präsentiert. Diese Zellen leisten eine Vorverarbeitung des Eingabemusters und leiten die transformierten Informationen an die Zellen in den verdeckten Schichten weiter. Hier erfolgt die hauptsächliche Verarbeitung der Informationen. Diese werden schließlich an die Zellen der Ausgabeschicht weitergeleitet, welche dann ein Ausgabemuster erzeugen.

Die Ausgänge der Zellen sind mit den Eingängen anderer Zellen verbunden, die wiederum eine Transformation des Musters durchführen können. Je mehr Zellschichten bei der Verarbeitung des Musters beteiligt sind, desto komplexere Transformationen können auf dem Eingangsmuster ausgeführt werden. Das resultierende Ausgangsmuster liegt nach Ende der Verarbeitung an den Ausgängen der Zellen der Ausgabeschicht an.

Die Verbindungen zwischen den einzelnen Zellen und die Richtung des Informationsflusses innerhalb des Netzes definieren die Architektur eines NN (siehe Patterson, 1996, S.35). Bei der Richtung des Informationsflusses innerhalb des Netzes wird zwischen zwei grundlegende Arten unterschieden.

Die erste und meist verwendete Art sind die *Feedforward-Netze*. In diesen Netzen existieren ausschließlich Verbindungen von einer Schicht zu einer jeweils höher gelegenen Schicht. Je höher eine Schicht gelegen ist, desto näher befindet sie sich an der Ausgabeschicht des Netzes. Es sind weder Verbindungen zwischen Zellen einer Schicht und Zellen einer darunterliegenden Schicht, noch zwischen Zellen der gleichen Schicht vorhanden. Die Berechnung des Ausgabemuster erfolgt bei Feedforward-Netzen als ein einziger Vorwärtsschritt (siehe Rigoll, 1994, S.48), weswegen man sie auch als statische Systeme bezeichnet.

Die zweite Art sind die *Feedback-Netze*. Die Besonderheit von Feedback-Netzen sind die Rückkopplungen zwischen den verschiedenen Netzschichten. Diese werden durch Verbindungen zwischen Zellen höherer Schichten und Zellen darunterliegender Schichten oder zwischen Zellen einer Schicht ermöglicht. Dabei dient die Ausgabe einer höheren Schicht als neue Eingabe für darunterliegende Schichten. Bei Feedback-Netzen wird das Ausgabemuster in mehreren rekursiven Teilschritten berechnet. Im Gegensatz zu den Feedforward-Netzen werden Feedback-Netze als dynamische Systeme bezeichnet. Durch geeignete Wahl der Gewichte kann ein Konvergenzverhalten des Systems erreicht und damit ein stabiles Ausgabemuster berechnet werden.

4.3.3 Wichtige Eigenschaften neuronaler Netze

Die NN verfügen gegenüber den klassischen Algorithmen über einige spezielle Eigenschaften. Diese sollen im folgenden Abschnitt beschrieben werden.

4.3.3.1 Lernfähigkeit

Wie in Abschnitt 4.3 bereits erläutert wurde, transformieren die Zellen das anliegende Eingangssignal mittels einer mathematischen Vorschrift in das Ausgangssignal. Die Parameter der mathematischen Vorschrift müssen angepaßt werden, um die gewünschte Transformation von Eingangssignal zu Ausgangssignal zu erhalten. Diese Parameter, welche hauptsächlich durch die Gewichte der Verbindungen zwischen den einzelnen Zellen gegeben sind, können vom NN erlernt werden. Zum Begriff des Lernens schreibt Schöneburg et al. (1990, S.45):

Als Lernen bezeichnet man die Modifikation von Verhalten aufgrund von Erfahrung. ... Gelernt wird, auf einen bestimmten Reiz mit einer bestimmten Reaktion zu antworten.

Ein NN kann von Beispielen lernen, welche dem Netz präsentiert werden. Es werden dabei Beispielmusterpaare an den Eingängen und Ausgängen des NN angelegt. Das Netz versucht nun mittels eines Optimierungsverfahrens die Netzparameter so anzupassen, daß das gewünschte Übertragungsverhalten zwischen Eingangssignal und Ausgangssignal erreicht wird. Diese Phase, in der das NN aus Beispielen lernt, nennt man *Lernphase* oder *Trainingsphase*. Das NN ist dabei in der Lage, aus den präsentierten Beispielen zu einer Verallgemeinerung (Generalisierung) der Beispieldaten zu kommen. Die Generalisierung ist für den Lernprozess sehr wichtig, da sie die Bildung von Musterklassen durch das NN ermöglicht. Diese Fähigkeit kann es bei der Verarbeitung von bis dahin noch unbekannten Musterpaaren anwenden. In dieser Phase, der *Anwendungsphase*, ist das Netz in der Lage, unbekannte Muster korrekt zu verarbeiten, wenn sie der allgemeinen Systematik der Beispieldaten entsprechen.

Einige künstliche NN sind in der Lage, auch in der Anwendungsphase weiterzulernen. Dadurch haben sie die Fähigkeit, sich ständig an die möglicherweise wechselnden Bedingungen anzupassen. Damit zeigen sie ein adaptives Verhalten, welches für bestimmte Anwendungen nützlich und notwendig sein kann.

Auf die verschiedenen Lernverfahren neuronaler Netze wird im Abschnitt 4.3.4 näher eingegangen.

4.3.3.2 Verarbeitung unscharfer Informationen

Wie ihre biologischen Vorbilder haben künstliche NN die Fähigkeit, fehlerhafte oder unvollständige Informationen korrekt zu verarbeiten. Diese werden auch als *unscharfe* Informationen bezeichnet. Viele Datenmengen wie Sprache oder Bilder bestehen aus unscharfen Informationen, die aber beispielsweise von den menschlichen Sinnesorganen sehr erfolgreich verarbeitet werden können. Ein an den Eingängen eines künstlichen NN anliegendes Signal kann verrauscht oder nur unvollständig sein. Trotzdem ist ein NN in der Lage, aus diesen unscharfen Informationen ein korrektes Ausgabemuster zu erzeugen.

Durch diese Eigenschaft sind künstliche NN besonders geeignet, um Sprach- und Bildsignale maschinell zu verarbeiten (siehe Rigoll, 1994, S.8).

4.3.3.3 Parallelverarbeitung

Wie im Abschnitt 4.3 beschrieben wurde, bestehen künstliche NN aus verschiedenen Schichten einzelner autonomer Zellen. Diese arbeiten voneinander unabhängig und kommunizieren über gewichtete Verbindungen miteinander. Ein NN kann also als ein System aus parallel arbeitenden Teilsystemen betrachtet werden.

Die Simulation eines künstlichen NN läuft meist auf einem konventionellen Rechner. Dabei werden die Operationen der einzelnen Zellen des Netzes sequentiell abgearbeitet. Vorstellbar wäre aber auch, die einzelnen Zellen mit ihren Operationen auf einem jeweils eigenen Prozessor zu realisieren. Diese Prozessoren könnten sehr einfach aufgebaut sein. Das gesamte Netz ließe sich dann durch die Verbindung der Einzelprozessoren realisieren. Dieser Ansatz würde einen riesigen Geschwindigkeitsvorteil bringen, da die einzelnen Zellen nicht mehr sequentiell sondern parallel arbeiten könnten. Durch Hinzufügen neuer Einzelprozessoren kann die Leistungsfähigkeit des Netzes erhöht werden.

Die Architektur der künstlichen NN kommt einer parallelen Realisierung sehr entgegen. Im Gegensatz zu herkömmlichen Algorithmen, die erst aufwendig parallelisiert werden müssen, weisen neuronale Netze eine natürliche Parallelität auf. Hinzu kommt, daß bei herkömmlichen Parallelrechnern der Kommunikationsaufwand bei Erhöhung der Prozessorzahl stark steigt. Das führt dazu, daß die Leistung von Parallelrechnern ab einer gewissen Prozessorzahl konvergiert. Da bei künstlichen NN die Verbindung zwischen den einzelnen Zellen sehr einfach ist, führt hier die Erhöhung der Anzahl der Prozessoren auch zu einer Erhöhung der Leistung.

Durch moderne Fertigungsverfahren in der Mikroelektronik lassen sich künstliche NN als Chip realisieren. Die einzelnen Zellen des Netzes sind dabei direkt auf dem Chip implementiert. Durch die erreichbaren kleinen Dimensionen und den Wegfall eines aufwendigen Steuerrechners sind verschiedenste Einsatzgebiete sowie eine kostengünstige Massenpro-

duktion möglich. Die resultierenden hohen Verarbeitungsgeschwindigkeiten erlauben es, Echtzeitanwendungen zu realisieren.

4.3.3.4 Fehlertoleranz

Künstliche NN weisen ein hohes Maß an Fehlertoleranz auf. Damit ist jedoch nicht die Toleranz gegenüber unvollständigen oder fehlerhaften Eingabemustern gemeint. Vielmehr bezieht sich der Begriff Fehlertoleranz auf Versagen beziehungsweise den Ausfall einzelner Zellen des Netzes.

Bei herkömmlichen sequentiellen Rechnern oder Parallelrechnern kann es passieren, daß bei Ausfall eines Transistors oder eines Prozessors eine massive Störung oder sogar ein Totalausfall des gesamten Rechnersystems auftritt. Bei einem künstlichen NN bewirkt jedoch ein Ausfall einzelner Zellen kaum eine Veränderung des Netzverhaltens. Je mehr Zellen ausfallen, desto schlechter wird zwar das Netzverhalten, es kommt dabei aber kaum zum Totalausfall (siehe Rigoll, 1994, S.12).

Dieses Verhalten kann mit der massiven Parallelität und hohen Konnektivität eines künstlichen NN erklärt werden. Die Funktionalität eines Netzes ist auf eine große Anzahl von Zellen und gewichteten Verbindungen zwischen den Zellen verteilt. Dadurch weisen die Netze eine gewisse Redundanz der im Netz gespeicherten Informationen auf. Somit kann der Ausfall einzelner Zellen das Gesamtsystem kaum verändern, da noch viele andere Zellen die Netzfunktionalität erhalten. Die Netze können bewußt überdimensioniert werden, um die Fehlertoleranz zu erhöhen und um eine noch höhere Verteilung der Information und Funktion innerhalb des Netzes zu ermöglichen.

Aufgrund dieser Eigenschaft sind künstliche NN für Anwendungen geeignet, in denen eine sehr hohe Verfügbarkeit gewährleistet werden muß.

4.3.4 Lernverfahren neuronaler Netze

Es gibt verschiedene Ansätze, um die optimalen Gewichtungen für das gewünschte Übertragungsverhalten zwischen Eingabe- und Ausgabemuster eines NN zu ermitteln. Diese können im Wesentlichen in das *überwachte Lernen* und das *unüberwachte Lernen* unterteilt werden.

4.3.4.1 Überwachtes Lernen

Dieses Lernverfahren wird auch als *Supervised Learning* bezeichnet. Wie der Name bereits erkennen läßt, erfolgt eine Überwachung des Lernfortschrittes des NN während der Trainingsphase. Das aus einem Beispieleingabemuster erzeugte Ausgabemuster y_{ist} wird

mit dem Sollausgabemuster y_{soll} verglichen. Mittels einer Fehlerfunktion kann die Abweichung zwischen den beiden Ausgabemustern beziehungsweise der Lernfortschritt berechnet werden. Eine oft verwendete Fehlerfunktion ist der quadratische Fehler.

$$\epsilon = (y_{soll} - y_{ist})^2 \quad (4.3)$$

Die berechnete Abweichung wird einer sogenannten Lernregel präsentiert, die daraus die Gewichtungen der Verbindungen zwischen den Zellen anpaßt. Häufig wird dafür die Gradientenregel verwendet:

$$\Delta w = -\beta \cdot \frac{\partial \epsilon}{\partial w} \quad (4.4)$$

Es ist zu erkennen, daß sich die Lernregel an den aktuellen Lernfortschritt ϵ anpaßt. Bei einer großer Abweichung zwischen den Ausgabemustern ergibt sich eine starke Veränderung der Gewichte, bei einer kleiner Abweichung werden die Gewichte kaum noch verändert. Dadurch wird im Verlauf der Trainingsphase der quadratische Fehler minimiert und die Gewichte werden optimal eingestellt.

Die Anpassung der Gewichte bei mehrschichtigen Netzen mit verdeckten Neuronenschichten ist sehr komplex. Die *Backpropagation*-Methode wurde entwickelt, um diesen Schritt zu erleichtern. Auf diese Methode soll hier nicht näher eingegangen werden. Weitere Details sind in der entsprechenden Literatur wie Patterson (1996) und Ritter et al. (1991) nachzulesen.

4.3.4.2 Unüberwachtes Lernen

Anders als beim überwachten Lernen wird bei diesem Lernverfahren der Lernfortschritt nicht überwacht. Dadurch ergeben sich vereinfachte Lernregeln, welche nicht dem aktuellen Lernfortschritt angepaßt werden müssen. Die Gewichte der Verbindungen zwischen den Zellen können so meist in einem einzigen Rechenschritt berechnet werden. Es sind aber auch kompliziertere Formen des unüberwachten Lernens möglich. Diese können beispielsweise in rekursiver Form implementiert werden.

Eine sehr einfache Lernregel für unüberwachtes Lernen ist die *Hebb'sche Lernregel*. Diese wurde den biologischen neuronalen Netzen entlehnt und ist eine Form des nicht-konkurrierenden Lernens. Eine vereinfachte Beschreibung lautet wie folgt:

Definition 4.1 (Hebb'sche Lernregel)

Werden zwei miteinander verbundene Neuronen gleichzeitig erregt, so verstärkt sich ihre synaptische Verbindung.

Diese Regel kann auf künstliche neuronale Netze angewendet werden. Sind zwei miteinander verbundene Zellen gleichzeitig aktiv, so wird das Gewicht ihrer Verbindung w_{ij} erhöht.

$$\Delta w_{ij} = \alpha \cdot x_i \cdot y_j \quad (4.5)$$

Die Variable x_i ist die i -te Komponente des Eingabevektors, y_j die j -te Komponente des Ausgabevektors und α eine Konstante für die Normalisierung.

Es gibt aber auch Formen des konkurrierenden Lernens. Ein Beispiel dafür ist die *winner-takes-all*-Methode. Hier lernt nur die Zelle, deren Gewichtsvektor der Eingabe am ähnlichsten ist, während unter den Verlierern kein Lernen stattfindet. Sie konkurriert dabei mit den anderen Zellen des neuronalen Netzes und hemmt diese Zellen in ihrer Ausgabe. Idealerweise wird beim Lernen des Gewinners, dessen Gewichtsvektor w in Richtung des Eingabevektors x verschoben. Dies geschieht laut Patterson (1996, S.403), indem ein Bruchteil des Differenzvektors $(x - w)$ zum Gewichtsvektor addiert wird.

Es ist typisch für das unüberwachte Lernen, daß das Erreichen der gewünschten Ausgabemuster nicht überwacht wird. Die Berechnung der dafür notwendigen Gewichtungen kann für einfache Netze in einem Rechenschritt, bei komplexeren Netzen iterativ mit Methoden der Selbstorganisation erfolgen.

4.3.4.3 Konstante und variable Lernrate

Für die Adaption der Gewinnerzelle eines Eingabesignals beim konkurrierenden Lernen wird vorwiegend eine sogenannte Lernrate ϵ verwendet. Diese beschreibt, wie stark der Gewichtsvektor einer Zelle in Richtung des zugehörigen Eingabesignals verschoben wird. Es gibt *konstante* und *variable* Lernraten. Die konstante Lernrate bleibt während der gesamten Lernphase des Netzes unverändert.

$$\epsilon = \epsilon_0 \quad \text{mit} \quad (0 < \epsilon_0 \leq 1) \quad (4.6)$$

Laut Fritzke (1997, S.13) repräsentiert der Gewichtsvektor w_c einer Zelle das Mittel aller Signale, für die diese Zelle der Gewinner war. Das hat zur Folge, daß das Netz immer lern- und anpassungsfähig bleibt. Andererseits kann es nicht konvergieren beziehungsweise in einen stationären Zustand übergehen.

Statt eine konstante Lernrate für alle Zellen des Netzes zu verwenden, kann für jede einzelne Zelle eine separate variable Lernrate eingeführt werden. Dieser als *k-means* bezeichnete Algorithmus verwendet folgende Formel für die Berechnung der aktuellen Lernrate einer Zelle (siehe Fritzke, 1997, S.14):

$$\epsilon(t) = \frac{1}{t} \quad (4.7)$$

Der Parameter t repräsentiert die Anzahl der Signale, für welche die jeweilige Zelle Gewinner war. Damit wird jeder Gewichtsvektor $w_c(t)$ zum arithmetischen Mittel der zu dieser Zelle c korrespondierenden Eingabesignale. Mit dieser Lernrate ist ein Konvergieren ebenso nicht möglich. Trotzdem bildet sich bei Verwendung einer festen Verteilung der Eingabesignale im Eingaberaum ein fast stationärer Netzzustand heraus.

Ein weiteres Beispiel für eine variable Lernrate ist die *exponentiell sinkende Lernrate*. Hier wird folgende Formel zur Berechnung der aktuellen Lernrate verwendet:

$$\epsilon(t) = e_i(e_f/e_i)^{t/t_{max}} \quad (4.8)$$

Die Variable e_i ist die initiale Lernrate, e_f die finale Lernrate, t die Anzahl der durchgeführten Adaptionsschritte und t_{max} die maximale Anzahl der durchzuführenden Adaptionsschritte. Für den Zeitpunkt $t = 0$ nimmt die Lernrate den Wert e_i an, da der Exponent t/t_{max} Null wird. Mit steigendem t wird der Exponent größer, bis er bei $t = t_{max}$ den Wert 1 annimmt. Die Lernrate hat zu diesem Zeitpunkt genau den Wert der finalen Lernrate e_f . Die gewählte initiale Lernrate sollte größer als die finale Lernrate sein. Dadurch kann in einem frühen Stadium der Lernphase eine hohe Adaptivität des Netzes erreicht werden. In einem späteren Stadium wird die Anpassungsfähigkeit des Netzes geringer. Dies führt zu einem stabileren Zustand des Netzes. Netze, welche diese Lernrate verwenden, haben keine feste Konvergenz. Sie bleiben daher auch im fortgeschrittenen Lernstadium noch lernfähig.

4.4 Einige Modelle neuronaler Netze

Im Folgenden sollen einige Modelle künstlicher NN vorgestellt werden, welche für die weiteren Kapitel dieser Arbeit von Bedeutung sind. Bei dieser Betrachtung liegt der Schwerpunkt auf den einschichtigen Netzwerken und dem unüberwachten Lernen (siehe dazu Abschnitt 4.3.4). Für Informationen über mehrschichtige Netzwerke oder überwachte Lernverfahren sei auf die Literatur wie Rigoll (1994), Schöneburg et al. (1990), Brause (1991) sowie Patterson (1996) verwiesen.

4.4.1 Topologieerhaltende Abbildungen

Wie im Abschnitt 4.3.4 beschrieben ist, stellt die gegenseitige Hemmung von Zellen einer Schicht einen wesentlichen Bestandteil für konkurrierende Lernverfahren dar. Diese Verfahrensweise des konkurrierenden Lernens kann noch verfeinert werden (siehe Brause, 1991, S.132). Dafür werden zwischen den einzelnen Zellen einer Netzwerkschicht Abstände beziehungsweise Nachbarschaften definiert. Denkbar wäre die regelmäßige Verteilung der einzelnen Zellen im zweidimensionalen oder dreidimensionalen Raum. Es werden also

jeder Zelle Raumkoordinaten zugeordnet. Dadurch können die globalen Wechselwirkungen zwischen den Zellen einer Netzwerkschicht in lokale Wechselwirkungen mit einem unterschiedlichen Einfluß auf die einzelnen Zellen gewandelt werden. Diese Anordnung bietet die Möglichkeit, eine Menge von hochdimensionalen Musterpaaren auf niederdimensionale Koordinaten abzubilden. Eine wichtige Eigenschaft dieser Abbildung ist, daß sie sich topologieerhaltend beziehungsweise nachbarschaftserhaltend verhält. Das bedeutet, daß benachbarte Muster im hochdimensionalen Eingaberaum auch im niederdimensionalen Ausgaberaum benachbart sind.

4.4.2 Self-Organizing Feature Map

Das Modell der selbstorganisierenden Merkmalskarten (Self-Organizing Feature Map, kurz SOFM) wurde vom Finnen *Teuvo Kohonen* entwickelt (siehe Kohonen, 2001). Es geht von einer zweidimensionalen, nachbarschaftlichen Anordnung der Zellen in einem einschichtigen Netzwerk aus. Wird ein rechteckiges Netzmuster zugrunde gelegt, so besitzt jede Zelle vier direkte Nachbarn. Die Verbindung der Zellen untereinander führt zu einer Rückkopplung der Netzschicht auf sich selbst. Die Dimensionalität und Topologie des Netzes wird von vornherein festgelegt und kann nicht mehr verändert werden.

Allen Zellen der Schicht wird parallel die gesamte Eingabeinformation des Eingabevektors x präsentiert. Die einzelnen Zellen des Netzwerkes bilden über die in Abschnitt 4.3.1 beschriebenen Propagierungsfunktionen die Summe der gewichteten Eingaben net . Über die lineare Aktivierungsfunktion (siehe Formel 4.2) wird die Aktivität a der einzelnen Zellen berechnet. Sie bestimmt das sogenannte Erregungszentrum für das jeweilige Eingabesignal. Es wird diejenige Zelle s zum Erregungszentrum, welche die maximale Aktivität beziehungsweise höchste Korrelation für dieses Eingabesignal besitzt. Abweichend davon kann auch die Zelle mit dem kleinsten Abstand zwischen Eingabevektor und Gewichtsvektor $(x - w)$ als Erregungszentrum verwendet werden.

Mit Hilfe einer geeigneten Ausgabefunktion h_{rs} passen die Zelle s im Erregungszentrum und deren Nachbarzellen r ihre Gewichte an. Geeignete Ausgabefunktionen sind beispielsweise die *Gaußglocke* oder die *Mexikanerhut-Funktion*, welche in der Abbildung 4.5 dargestellt sind (siehe Brause, 1991; Ritter et al., 1991). Auf der x-Achse wird der Abstand einer Zelle zum Erregungszentrum abgetragen. Der Funktionswert an der Stelle x entspricht der Ausgabe einer Zelle mit dem Abstand x zum Erregungszentrum. Die Ausgabe einer Zelle ist also abhängig von ihrem Abstand zur Gewinnerzelle. Die Zellen in direkter Nachbarschaft des Erregungszentrums werden erregt, während alle weiter entfernten Zellen gehemmt werden. Es besteht somit ein Wettbewerb zwischen allen Zellen der Schicht, wobei nur die Gewinnerzelle und einige Zellen in der Umgebung des Gewinners lernen. Die Stärke der Adaption einer Zelle hängt von ihrer Ausgabe und der verwendeten exponentiell sinkenden

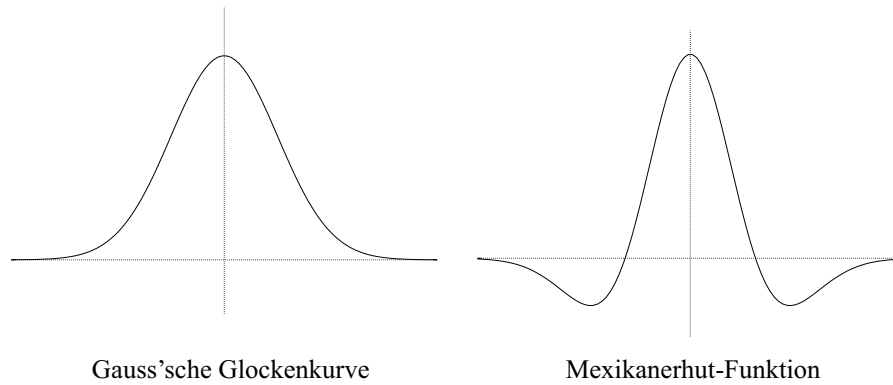


Abbildung 4.5: Mögliche Ausgabefunktionen für konkurrierendes Lernen

Lernrate ab. Der gesamte Algorithmus der selbstorganisierenden Merkmalskarten läßt sich laut Fritzke (1997, S.30f) in folgende Schritte einteilen:

1. Initialisieren der Menge C so daß sie $N = N_1 \cdot N_2$ Zellen c_i enthält

$$C = \{c_1 \dots c_N\}$$

Initialisieren der Gewichtsvektoren $w_{c_i} \in \mathcal{R}^n$ der Zellen mit Zufallswerten

Initialisieren der Menge der Verbindungen K derart, daß sie ein rechteckiges Netz-
muster $N_1 \times N_2$ bildet

Initialisieren des Parameters t mit $t = 0$

2. Generieren eines zufälligen Eingangssignals ξ aus dem Eingaberaum
3. Bestimmen der Gewinnerzelle s mit dem kleinstem Abstand zum Signal ξ :

$$\|\xi - w_s\| \leq \|\xi - w_c\| \quad (\forall c \in C)$$

4. Adaptieren aller Zellen r entsprechend der Ausgabefunktion h_{rs} und der Lernrate $\epsilon(t)$

$$\text{mit } \Delta w_r = \epsilon(t) \cdot h_{rs} \cdot (\xi - w_r)$$

$$\text{und } \epsilon(t) = \epsilon_i (\epsilon_f / \epsilon_i)^{t/t_{max}}$$

5. Erhöhen den Parameters t :

$$t = t + 1$$

6. Wiederholen ab Schritt 2, solange $t < t_{max}$ ist

4.4.3 Growing Cell Structures

Das Modell der *Wachsenden Zellstrukturen* wurde von *Bernd Fritzke* entworfen (Fritzke, 1993). Wie bei dem Modell der selbstorganisierenden Merkmalskarten, besitzt das NN eine feste Dimensionalität. Das Netzwerk ist darauf beschränkt, ausschließlich aus k -dimensionalen Simplexen zu bestehen. Die Variable k ist ein positiver ganzzahliger Wert, welcher im vornherein festgelegt werden muß.

Im Ausgangszustand besteht das Netz aus einigen Zellen und Verbindungen, welche einen k -dimensionalen Simplex formen. Für $k = 1$ ist dieser Simplex eine Linie, für $k = 2$ ein Dreieck, $k = 3$ ein Tetraeder und für $k > 3$ ein Hypertetraeder (siehe Fritzke, 1993, S.2). Jede Zelle hat einen n -dimensionalen Gewichtsvektor, der die Position der Zelle im n -dimensionalen Eingaberaum angibt. Die Abbildung des Eingaberaumes auf das Netzwerk erfolgt, indem jedes Eingangssignal auf die jeweils nächste Zelle abgebildet wird (siehe Abschnitt 4.4.2). Der Eingaberaum wird infolgedessen in eine Anzahl von Regionen unterteilt, welche durch die Gewichtsvektoren der einzelnen Zellen des Netzes repräsentiert werden. Dies entspricht der Unterteilung des Raumes in Voronoi-Regionen wie in Abschnitt 3.3.1 beschrieben ist. Die Voronoi-Region ist das dreidimensionale Äquivalent zum Voronoi-Polygon im zweidimensionalen Raum.

Während des selbstorganisierenden Adaptionsprozesses werden die Gewichtsvektoren der Zellen des Netzes verändert, sowie neue Zellen und Verbindungen erzeugt. Es werden jeweils nur die gewinnende Zelle und ihre direkten topologischen Nachbarn adaptiert. Die Adaption verläuft während des ganzen Prozesses mit konstanten Lernraten für den Gewinner und die Nachbarn des Gewinners. Gleichzeitig enthält jede Zelle einen lokalen Fehler, mit dessen Hilfe entschieden wird, an welcher Stelle im Netz neue Zellen einzufügen sind. Dieser enthält die Summe der quadratischen Abstände zu den Eingangssignalen, für welche diese Zelle Gewinner war. Je weiter die korrespondierenden Eingangssignale von der Zelle entfernt sind, desto größer ist dieser Fehler. Der größte Fehler markiert demnach die Stelle der schlechtesten Abbildung des Eingaberaumes. Eine neue Zelle wird immer in der Nachbarschaft der Zelle q mit dem größten lokalen Fehler eingefügt. Das Einfügen erfolgt durch Teilung der längsten von q ausgehenden Verbindung. Danach werden neue Verbindungen erzeugt, so daß das Netzwerk wieder ausschließlich aus k -dimensionalen Simplexen besteht.

Wenn ausschließlich Zellen in das Netz eingefügt werden, so sind alle Zellen direkt oder indirekt miteinander verbunden. Besteht jedoch der Eingaberaum aus mehreren separaten Gebieten, so treten teilweise sehr weitreichende Verbindungen auf. Es können auch Zellen existieren, welche in Gebieten mit einer sehr niedrigen oder nicht vorhandenen Eingabedatenverteilung $p(\xi)$ liegen (siehe Fritzke, 1993, S.9). Es ist sinnvoll, diese Zellen zu entfernen, um eine optimale Abbildung des Eingaberaumes auf das Netz zu erhalten. Es muß

natürlich sichergestellt werden, das die Netzkonsistenz (siehe Abschnitt 3.2) erhalten bleibt. Die folgenden Punkte beschreiben den Algorithmus der *Wachsenden Zellstrukturen*. Das Einfügen und Löschen von Zellen wird aus Gründen der Einfachheit und Übersichtlichkeit in separaten Abschnitten erläutert.

1. Wählen der Netzwerkdimension k

Initialisieren der Menge C mit $k + 1$ Zellen c_i :

$$C = \{c_1, \dots, c_{k+1}\}$$

Initialisieren der Gewichtsvektoren $w_{c_i} \in \mathcal{R}^n$ der Zellen mit Zufallswerten aus dem Eingaberaum

Initialisieren der Menge K der Verbindungen zwischen den Zellen, so daß alle Zellen miteinander verbunden sind und einen k -dimensionalen Simplex formen

2. Generieren eines zufälligen Eingabesignals ξ aus dem Eingaberaum

3. Bestimmen der Gewinnerzelle s mit dem kleinstem Abstand zum Signal ξ :

$$\|\xi - w_s\| \leq \|\xi - w_c\| \quad (\forall c \in C)$$

4. Aufsummieren des quadratischen Abstandes zwischen Eingabesignal und dem Gewichtsvektor des Gewinners s mittels einer lokalen Fehlervariable:

$$\Delta E_s = \|\xi - w_s\|^2$$

5. Adaptieren der Gewichtsvektoren des Gewinners und seiner direkten topologischen Nachbarn mittels der Lernraten ϵ_b und ϵ_n

$$\Delta w_s = \epsilon_b(\xi - w_s), \quad \Delta w_i = \epsilon_n(\xi - w_i) \quad (\forall i \in N_s)$$

6. Einfügen einer neuen Zelle, wenn die Anzahl der generierten Eingabesignale ein ganzzahliges Vielfaches eines Parameters λ_1 ist

7. Löschen von Zellen, wenn die Anzahl der generierten Eingabesignale ein ganzzahliges Vielfaches eines Parameters λ_2 ist

8. Verringern des lokalen Fehlers aller Zellen um einen Bruchteil β :

$$\Delta E_c = -\beta E_c \quad (\forall c \in C)$$

9. Wiederholen ab Schritt 2, bis ein Abbruchkriterium (zum Beispiel die Netzgröße) erreicht ist

Bei der Wahl der Parameter λ_1 und λ_2 ist zu beachten, daß das Einfügen von Zellen weitaus öfter als das Löschen von Zellen stattfinden sollte. Ansonsten kann das Wachsen des Netzes nicht mehr sichergestellt werden. Dadurch wäre es in seiner Funktionalität beschränkt.

4.4.3.1 Einfügen von Zellen

Neue Zellen werden an der Stelle im Netz mit der schlechtesten Abbildung des Eingaberaumes eingefügt. Diese Stelle wird durch die Zelle q mit dem größten lokalen Fehler markiert.

$$E_q \geq E_c \quad (\forall c \in C)$$

Das Einfügen der neuen Zelle erfolgt durch Splitten der von q ausgehenden längsten Verbindung. Dafür wird aus der Menge N der direkten topologischen Nachbarn von q die Zelle f mit dem größten Abstand zu q ermittelt.

$$\|w_q - w_f\| \geq \|w_q - w_c\| \quad (\forall c \in N_q)$$

Die neue Zelle r wird nun in die Menge der Zellen C des Netzes eingefügt. Der Gewichtsvektor von r wird aus den Gewichtsvektoren von q und f interpoliert.

$$C = C \cup \{r\}, \quad w_r = (w_q + w_f)/2$$

Das Einfügen neuer Zellen muß immer ein konsistentes Netz als Ergebnis haben. Es werden neue Verbindungen zwischen r und q , f sowie allen gemeinsamen Nachbarn von q und f zu der Menge K der Verbindungen des Netzes hinzugefügt, um eine konsistente Topologie zu erhalten.

$$K = K \cup \{(r, q), (r, f), (r, g_1) \dots (r, g_n)\} \quad \text{mit} \quad g \in N_q \cap N_f$$

Im Anschluß wird die originale Verbindung zwischen q und f aus K entfernt, da diese nicht mehr benötigt wird.

$$K = K \setminus \{(q, f)\}$$

Durch das Einfügen einer neuen Zelle, verkleinern sich die Voronoi-Regionen der Nachbarzellen. Der lokale Fehler aller Nachbarzellen von r wird abhängig von der Anzahl der Nachbarn und dem Parameter α verringert, um dies widerzuspiegeln.

$$\Delta E_i = -\frac{\alpha}{|N_r|} E_i \quad (\forall i \in N_r)$$

Der lokale Fehler der Zelle r wird schließlich auf den mittleren lokalen Fehler seiner Nachbarn gesetzt.

$$E_r = \frac{1}{|N_r|} \sum_{i \in N_r} E_i$$

4.4.3.2 Löschen von Zellen

Für den Löschvorgang muß zuerst entschieden werden, welche Zellen gelöscht werden können. Dafür muß die Verteilung der Eingabesignale ermittelt werden. Diese ist aber in der Regel unbekannt und kann daher nur approximiert werden. Für jede Zelle c wird zusätzlich eine reellwertige Zählvariable τ_c mitgeführt. Diese gibt im Grunde genommen an, für wieviele Eingabesignale eine Zelle der Gewinner war. Mittels dieser Variablen kann eine relative Signalverteilung h_c berechnet werden. Diese steht für die relative Frequenz, in der eine Zelle Gewinner für ein Eingabesignal wird.

$$h_c = \tau_c / \sum_{i \in C} \tau_i \quad (4.9)$$

Die relative Signalverteilung h_c kann mit dem Volumen der Voronoi-Region in Relation gesetzt werden, um einen lokalen Schätzwert \tilde{p}_c für die Signalverteilung innerhalb der zur Zelle c gehörenden Voronoi-Region zu erhalten.

$$\tilde{p}_c = h_c / |F_c| \quad (4.10)$$

Da die Berechnung der Volumina der einzelnen Voronoi-Regionen für höherdimensionale Räume sehr aufwendig ist, führt Fritzke einen weiteren Schätzwert \tilde{f}_c ein. Für diesen Wert wird das Volumen eines n -dimensionalen Hyperwürfels berechnet, wobei dessen Kantenlänge der mittleren Länge \bar{l}_c der von der Zelle c ausgehenden Verbindungen entspricht.

$$\tilde{f}_c = \bar{l}_c^n \quad \text{mit} \quad \bar{l}_c = \frac{1}{|N_c|} \sum_{i \in N_c} \|w_c - w_i\| \quad (4.11)$$

$|N_c|$ gibt die Anzahl der direkten topologischen Nachbarn der Zelle c an.

Es wäre vorteilhaft, all jene Zellen zu löschen, deren relative Signalverteilung \tilde{p} unter einen festgelegten Grenzwert fällt. Dieser ist jedoch stark von der Ausdehnung der Verteilung abhängig. Bei zwei unterschiedlich großen Gebieten mit gleicher Signalverteilung ergibt sich eine unterschiedliche Dichte der Eingabesignale. Die relative Signalverteilung muß noch normiert werden, um einen globalen Grenzwert benutzen zu können. Dies wird durch das Multiplizieren der relativen Signalverteilung \tilde{p} mit der Summe der Volumina der einzelnen

Voronoi-Regionen beziehungsweise deren Schätzwerten \tilde{f} erreicht.

$$\hat{p}_c = \tilde{p}_c \cdot \sum_{i \in C} \tilde{f}_i \quad (4.12)$$

Durch die Wahl eines geeigneten Grenzwertes für die normierte Signalverteilung \hat{p} der Umgebung der einzelnen Zellen des Netzes, ist es möglich, stark getrennte Signalverteilungen abzubilden.

Für die Löschoperation einer Zelle aus dem Netz gilt die Bedingung, daß die Konsistenz der Netzstruktur erhalten bleiben muß. Konsistent heißt in diesem Zusammenhang, daß das Netz nach dem Löschvorgang wieder ausschließlich aus Simplexen besteht. Mit folgender Vorgehensweise kann sichergestellt werden, das die Netzkonsistenz erhalten bleibt:

1. Entfernen aller Simplexe, an denen die zu löschende Zelle c beteiligt ist
2. Entfernen aller Nachbarn von c , wenn sie nicht mehr Bestandteil mindestens eines Simplex sind
3. Entfernen aller von den ehemaligen Nachbarn von c ausgehenden Kanten, welche nicht mehr Bestandteil eines Simplex sind

4.4.4 Neural Gas

Dieses neuronale Netz wurde von *Thomas Martinetz* und von *Klaus Schulten* entwickelt (siehe Martinetz and Schulten, 1991). Es besitzt keinerlei Topologie, wie sie etwa bei den selbstorganisierenden Merkmalskarten vorhanden ist. Infolge dessen werden die Gewichtsvektoren der einzelnen Zellen unabhängig von ihrer topologischen Anordnung adaptiert. Die Zellen lernen abhängig von der Signalverteilung im Eingaberaum, welche Region des Raumes sie repräsentieren. Dies sind die Voronoi-Regionen, welche durch die Gewichtsvektoren der Zellen definiert werden. Der Grad der Adaption wird dabei vom relativen Abstand der Zellen zum Eingabesignal bestimmt. Für jedes Eingabesignal ξ wird eine Rangordnung aller Zellen gebildet. Diese werden nach ihrem Abstand zum Eingabesignal $(\xi - w)$ geordnet. Die Gewichtsvektoren werden anschließend abhängig von ihrer Position innerhalb dieser Rangordnung stärker oder schwächer adaptiert. Eine mögliche Ausgabeformel für die Wichtung der Adaption lautet:

$$h_\lambda(k_i) = \exp(-k_i/\lambda) \quad (4.13)$$

Die Variable k_i enthält die Anzahl der Zellen, welche vor der Zelle c_i innerhalb der Rangordnung liegen. Die Anzahl der Zellen, welche adaptiert werden sollen, ist durch die Va-

riable λ festgelegt. Diese Form des konkurrierenden Lernens wird als *winner-takes-most*-Methode bezeichnet.

Der *Neural Gas*-Algorithmus kann in folgende Schritte eingeteilt werden (siehe Fritzke, 1997, S.20):

1. Initialisieren der Menge C mit N Zellen c_i :

$$C = \{c_1, \dots, c_n\}$$

Initialisieren der Gewichtsvektoren $w_{c_i} \in \mathcal{R}^n$ der Zellen mit Zufallswerten

Initialisieren des Parameters t mit $t = 0$

2. Generieren eines zufälligen Eingabesignals ξ aus dem Eingaberaum
3. Ordnen aller Zellen aus C nach ihrem Abstand zum Eingabesignal ξ
4. Adaptieren aller Zellen c_i entsprechend der Ausgabefunktion $h_\lambda(k_i)$ aus der Formel 4.13:

$$\Delta w_i = \epsilon(t) \cdot h_\lambda(k_i) \cdot (\xi - w_i)$$

und mit folgenden Zeitabhängigkeiten:

$$\lambda(t) = \lambda_i (\lambda_f / \lambda_i)^{t/t_{max}}$$

$$\epsilon(t) = \epsilon_i (\epsilon_f / \epsilon_i)^{t/t_{max}}$$

5. Erhöhen den Parameters t :

$$t = t + 1$$

6. Wiederholen ab Schritt 2, solange $t < t_{max}$ ist

Für die zeitabhängigen Parameter λ und ϵ müssen geeignete initiale Werte (λ_i, ϵ_i) und finale Werte (λ_f, ϵ_f) festgesetzt werden.

4.4.5 Competitive Hebbian Learning

Dieses Modell wurde ebenfalls von *Thomas Martinetz* und *Klaus Schulten* entwickelt. Das Besondere bei diesem Ansatz ist, daß hier die Gewichtsvektoren in keiner Weise verändert werden. Die einzigen Anpassungen erfolgen auf der Ebene der Verbindungen zwischen den einzelnen Zellen. Für jedes Eingabesignal werden diejenigen zwei Zellen bestimmt, bei welchen der Abstand zwischen Eingabesignal und Gewichtsvektor $(\xi - w)$ minimal ist. Zwischen diesen beiden gewinnenden Zellen wird eine neue Verbindung erzeugt, falls diese noch nicht vorhanden ist. Dadurch entsteht ein aus den Zellen und ihren Verbindungen

bestehendes Netz. Dieses hat die Eigenschaft, daß es die Topologie der Eingabesignale im Eingaberaum optimal abbildet (siehe Fritzke, 1997, S.21). Im Verlauf der Lernphase werden so zwischen den festen Zellen topologische Nachbarschaften herausgebildet.

Der Algorithmus des *Competitive Hebbian Learning* (kurz CHL) kann durch folgende Schritte beschrieben werden (siehe Fritzke, 1997, S.21):

1. Initialisieren der Menge C mit N Zellen c_i :

$$C = \{c_1, \dots, c_n\}$$

Initialisieren der Gewichtsvektoren $w_{c_i} \in \mathcal{R}^n$ der Zellen mit Zufallswerten

Initialisieren der Menge K der Verbindungen zwischen den Zellen:

$$K = \emptyset$$

2. Generieren eines zufälligen Eingabesignals ξ aus dem Eingaberaum
3. Bestimmen der Zellen s_1 und s_2 aus der Menge der Zellen C derart, daß der Gewichtsvektor von s_1 den kleinsten und der Gewichtsvektor von s_2 den zweitkleinsten Abstand zum Eingabesignal ξ hat:

$$\begin{aligned} \|\xi - w_{s_1}\| &\leq \|\xi - w_c\| \quad (\forall c \in C) \\ \|\xi - w_{s_2}\| &\leq \|\xi - w_c\| \quad (\forall c \in C \setminus \{s_1\}) \end{aligned}$$

4. Wenn zwischen den Zellen s_1 und s_2 keine Verbindung existiert, dann wird sie erzeugt:

$$K = K \cup \{(s_1, s_2)\}$$

5. Wiederholen ab Schritt 2, bis die maximale Anzahl von Signalen erreicht wird

Da dieser Ansatz ausschließlich Verbindungen zwischen den Zellen generiert, wird er kaum separat genutzt. Seine Anwendung erfolgt meist als Ergänzung zu anderen Modellen neuronaler Netze.

4.4.6 Growing Neural Gas

Die Entwicklung dieses NN erfolgte ebenfalls durch *Bernd Fritzke*. Während des Adaptionsprozesses werden in das Netzwerk neue Zellen und Verbindungen eingefügt und teilweise aus ihm entfernt. Da jedoch öfter Zellen und Verbindungen neu erzeugt werden,

wächst das Netz, indem es sich immer weiter ausbreitet. Dieser Effekt wird durch die Verbindung des Wachstumsmechanismus der *Wachsenden Zellstrukturen* und dem topologieabbildenden Mechanismus des *Competitive Hebbian Learning* erzielt.

Beginnend mit einigen wenigen Zellen, wird das Netz fortwährend erweitert. Das Einfügen neuer Zellen geschieht dort, wo die größten Abweichungen zum Eingaberaum bestehen. Dazu führt jede Zelle einen lokalen Fehler mit, der die akkumulierten Abstände zu den korrespondierenden Eingabesignalen enthält. Die Zelle mit dem größten Fehler markiert die Stelle, an der die neue Zelle einzufügen ist.

Zwischen zwei Zellen c_i und c_j wird eine Verbindung erzeugt, wenn die Voronoi-Regionen dieser Zellen im Raum nebeneinander liegen. Diese Verbindung repräsentiert dann die Ähnlichkeit beziehungsweise die Nachbarschaftsbeziehung der Zellen im Eingaberaum. Für jede dieser Verbindungen wird ein Alter mitgeführt, welches bei jedem Adaptionsschritt für die jeweils gewinnende Zelle erhöht wird. Überschreitet eine Verbindung das festgelegte maximale Alter T , so wird sie aus der Menge K der Verbindungen entfernt.

Der komplette Algorithmus für den Ansatz des *Growing Neural Gas* lautet wie folgt (siehe Fritzke, 1997, S.25):

1. Initialisieren der Menge C derart, daß sie zwei Zellen c_1 und c_2 enthält:

$$C = \{c_1, c_2\}$$

Initialisieren der Gewichtsvektoren $w_{c_i} \in \mathcal{R}^n$ der Zellen mit Zufallswerten

Initialisieren der Menge K der Verbindungen zwischen den Zellen:

$$K = \emptyset$$

2. Generieren eines zufälligen Eingabesignals ξ aus dem Eingaberaum
3. Bestimmen der Zellen s_1 und s_2 aus der Menge der Zellen C derart, daß der Gewichtsvektor von s_1 den kleinsten und der Gewichtsvektor von s_2 den zweitkleinsten Abstand zum Eingabesignal ξ hat:

$$\|\xi - w_{s_1}\| \leq \|\xi - w_c\| \quad (\forall c \in C)$$

$$\|\xi - w_{s_2}\| \leq \|\xi - w_c\| \quad (\forall c \in C \setminus \{s_1\})$$

4. Wenn zwischen den Zellen s_1 und s_2 keine Verbindung existiert, dann wird diese erzeugt und ihr Alter gesetzt:

$$K = K \cup \{(s_1, s_2)\}, \quad age_{(s_1, s_2)} = 0$$

5. Aufsummieren des quadratischen Abstandes zwischen dem Eingabesignal und dem Gewichtsvektor des Gewinners s_1 mittels einer lokalen Fehlervariable:

$$\Delta E_{s_1} = \|\xi - w_{s_1}\|^2$$

6. Adaptieren der Gewichtsvektoren des Gewinners und seiner topologischen Nachbarn mittels der Lernraten ϵ_b und ϵ_n :

$$\Delta w_{s_1} = \epsilon_b(\xi - w_{s_1}), \quad \Delta w_i = \epsilon_n(\xi - w_i) \quad (\forall i \in N_{s_1})$$

7. Erhöhen des Alters aller von der Zelle s_1 ausgehenden Verbindungen:

$$age_{(s_1,i)} = age_{(s_1,i)} + 1 \quad (\forall i \in N_{s_1})$$

8. Entfernen von Verbindungen welche das maximale Alter T erreicht haben.
Entfernen aller Zellen welche keine Verbindungen zu anderen Zellen haben.
9. Einfügen einer neuen Zelle, wenn die Anzahl der generierten Eingabesignale ein ganzzahliges Vielfaches eines Parameters λ ist:

- Bestimmen der Zelle q mit dem maximalen lokalen Fehler:

$$E_q \geq E_c \quad (\forall c \in C)$$

- Bestimmen der Zelle f mit dem maximalen lokalen Fehler aus der Menge der direkten topologischen Nachbarn von q :

$$E_f \geq E_c \quad (\forall c \in N_q)$$

- Einfügen einer neuen Zelle r mit einem aus den Zellen q und f interpolierten Gewichtsvektor:

$$C = C \cup \{r\}, \quad w_r = (w_q + w_f)/2$$

- Einfügen von Verbindungen zwischen der Zelle r und den Zellen q und f
Löschen der Verbindung zwischen den Zellen q und f

$$K = K \cup \{(r, q), (r, f)\}, \quad K = K \setminus \{(q, f)\}$$

- Verringern der lokalen Fehler der Zellen q und f um eine Bruchteil α :

$$\Delta E_q = -\alpha E_q, \quad \Delta E_f = -\alpha E_f$$

- Interpolieren des lokalen Fehlers der Zelle r aus den Fehlern der Zellen q und f :

$$E_r = (E_q + E_f)/2$$

10. Verringern des lokalen Fehlers aller Zellen um einen Bruchteil β :

$$\Delta E_c = -\beta E_c \quad (\forall c \in C)$$

11. Wiederholen ab Schritt 2, bis ein Abbruchkriterium (zum Beispiel die Netzgröße) erreicht ist

4.4.7 Bewertung der Modelle

In diesem Abschnitt erfolgen ein Vergleich und eine Bewertung der verschiedenen oben vorgestellten NN. Um eine bessere Vergleichbarkeit zu ermöglichen, wurden nur einige Eigenschaften dieser Modelle als Kriterien herangezogen. Dazu gehören die Lernrate, die Anzahl der Zellen, die Topologie sowie die Dimensionalität der Netze.

4.4.7.1 Lernrate und Konvergenz

Wie in Abschnitt 4.3.4 beschrieben wurde, unterscheiden sich künstliche NN in der verwendeten Lernrate. Die hier beschriebenen Modelle weichen unter anderem durch die Konstanz beziehungsweise Variabilität der Lernrate voneinander ab.

Die SOFM sowie das *Neural Gas*-Modell nutzen eine exponentiell sinkende Lernrate. Durch geeignete initiale und finale Werte kann ein sehr gutes Adaptionverhalten des NN erreicht werden. Im fortgeschrittenen Lernstadium sind diese Netze noch ebenso flexibel und anpassungsfähig wie am Anfang. Der Einfluß von spät auftretenden Eingabewerten wird zwar immer schwächer, bleibt aber erhalten. Sie konvergieren aufgrund dessen nicht gegen einen festen Endzustand. Daher müssen geeignete Abbruchkriterien gefunden werden, um die Lernphase zu beenden. Bei der SOFM wird die Stärke der Adaption der Zellen während eines Adaptionsschrittes über die gewählte Ausgabefunktion gesteuert. Durch eine geeignete Parametrisierung der Ausgabefunktion kann die Adaptionstärke sehr präzise beeinflusst werden.

Beim *Growing Neural Gas* und bei den *Wachsenden Zellstrukturen* wird hingegen eine feste Lernrate verwendet. Beide Modelle besitzen jeweils zwei verschiedene Lernraten. Die erste wird für die jeweilige Gewinnerzelle eines Adaptionsschrittes und die zweite für die direkten topologischen Nachbarn des Gewinners verwendet. Damit kann die Stärke der Adaption in zwei Stufen gesteuert werden. Aufgrund der festen Lernrate bleibt das Netz immer

im gleichen Maße anpassungsfähig. Dies ist besonders bei einer unbekannten Anzahl von Signalen von Vorteil. Dadurch haben auch später auftretende Eingabesignale noch einen hohen Einfluß. Dies hat aber auch zur Folge, daß diese Netze nicht konvergieren. Es müssen also auch hier geeignete Abbruchbedingungen gefunden werden, um die Lernphase abzuschließen.

Das Modelle des *Competitive Hebbian Learning* verändert die Gewichtsvektoren der einzelnen Zellen des Netzes nicht. Deswegen wird hier keine Lernrate eingesetzt. Der Lernprozess beschränkt sich hier ausschließlich auf die Verbindungen zwischen den Zellen.

4.4.7.2 Anzahl der Zellen

Bei den Modellen der SOFM, des *Neural Gas* und des *Competitive Hebbian Learning* bleibt die Anzahl der Zellen des Netzes unverändert. Die Netze werden am Anfang mit einer festen Anzahl von Zellen initialisiert. Es muß also vorher kalkuliert werden, wieviele Zellen für den Adaptionsprozess benötigt werden. Weiterhin ist zu bedenken, wie viele Musterklassen im Eingaberaum erkannt werden, beziehungsweise wie genau der Eingaberaum abgebildet werden soll. Diese Faktoren sind natürlich immer von dem zu lösenden Problem abhängig und können nicht pauschal bestimmt werden.

Beim *Growing Neural Gas* und bei den *Wachsenden Zellstrukturen* ist die Anzahl der Zellen variabel. Wie aus dem Namen dieser Modelle schon hervorgeht, sind sie wachsende NN. Beide werden mit einer sehr kleinen Anzahl von Zellen initialisiert. Während der Lernphase werden ständig neue Zellen eingefügt, wodurch sich die Netze langsam über den Eingaberaum ausbreiten. Neue Zellen sind dort einzufügen, wo der akkumulierte Fehler zwischen dem Eingabesignal und der Ausgabe der Zellen am größten ist. Bei den *Wachsenden Zellstrukturen* können außerdem Zellen gelöscht werden, welche sich in Bereichen des Eingaberaumes mit einer geringen Signalverteilung befinden. Dadurch kann sich das Netz an den Eingaberaum optimal anpassen und diesen sehr gut abbilden.

4.4.7.3 Topologie und Netzdimensionalität

Die Topologie eines Netzes beschreibt die Verbindungen der Zellen untereinander. Die verschiedenen Netzmodelle werden in Hinblick auf ihre Topologie in zwei unterschiedliche Kategorien eingeteilt. Zur ersten Kategorie gehören die Modelle mit einer festen und zur zweiten Kategorie die Modelle mit einer variablen Topologie. Weiterhin ist zwischen einer Topologie mit fester Netzdimensionalität und solcher mit einer variablen Netzdimensionalität zu unterscheiden. Diese bezeichnet die Dimension des Ausgaberaumes, auf den der höher- oder gleichdimensionale Eingaberaum abgebildet wird.

Als einziges der oben beschriebenen Netzmodelle besitzt das *Neural Gas*-Modell überhaupt

keine Topologie. Das heißt, daß zwischen den einzelnen Zellen keinerlei Verbindungen bestehen. Die einzelnen Zellen werden lediglich anhand ihrer Ähnlichkeit zum Eingabesignal unterschiedlich stark adaptiert.

Die SOFM besitzt als einziges der vorgestellten Modelle feste Nachbarschaften. Die Verbindungen zwischen den Zellen werden am Anfang des Adaptionprozesses initialisiert. Zusammen mit der vorher festgelegten Anzahl von Zellen ergibt sich daraus ein Netzwerk, welches sich in Größe und Topologie nicht mehr verändert. Die einzigen Anpassungen, die während des Adaptionprozesses vorgenommen werden, betreffen die Gewichtsvektoren der Zellen des Netzwerkes. Die SOFM hat eine feste Dimensionalität. Die Zellen sind meist in einem zweidimensionalen Gitter angeordnet. Der Eingaberaum wird in diesem Fall auf einen zweidimensionalen Ausgaberaum abgebildet.

Alle anderen Netzmodelle besitzen hingegen eine variable Topologie. Beim *Competitive Hebbian Learning* erfolgt eine Veränderung ausschließlich über die Verbindungen zwischen den Zellen. Dabei werden jeweils zwei Zellen verbunden, welche einem gegebenen Eingabesignal am ähnlichsten sind. Diese spiegeln also geometrische Nachbarschaften der Zellen innerhalb des Netzwerkes wider. Die Nachbarschaften der Zellen werden dabei ständig an neu auftretende Eingabesignale angepaßt. Die Netzdimensionalität ist bei diesem Modell variabel und von der Dimension des Eingaberaumes abhängig.

Das Modell der *Wachsenden Zellstrukturen* verfügt ebenfalls über eine variable Topologie. Diese ist auch aufgrund der dynamischen Größe des Netzwerkes, in Hinblick auf die Anzahl der Zellen, notwendig. Die Netzdimensionalität dieses Modells wird vor dem Adaptionprozess festgelegt. Ein solches Netz besteht immer aus sogenannten Simplexen mit einer Dimension k . Beim Einfügen und Löschen von Zellen muß auf eine topologische Konsistenz geachtet werden. Das Verändern von Nachbarschaften erfolgt immer derart, daß die Netzdimensionalität erhalten bleibt. Wie auch die SOFM ist dieses Modell dazu geeignet, hochdimensionale Eingabedaten auf niedrigdimensionale Ausgangsdaten abzubilden.

Beim *Growing Neural Gas* ist eine variable Netztopologie mit variabler Dimensionalität vorhanden. Dies ist auf die Verwendung des Wachstumsmechanismus der *Wachsenden Zellstrukturen* sowie die Erzeugung von Zellverbindungen des *Competitive Hebbian Learning* zurückzuführen. Die Anzahl der Zellen innerhalb des Netzes verändert sich während der Adaption ständig. Die Zellnachbarschaften werden im Laufe der Lernphase dauernd den auftretenden Eingabesignalen angepaßt. Damit spiegelt das Netz immer die Dimension des Eingaberaumes wider.

Kapitel 5

Ein alternatives adaptives Triangulationsverfahren

In diesem Kapitel soll das im Rahmen dieser Arbeit entwickelte adaptive Triangulationsverfahren vorgestellt werden. Ausgangspunkt sind die in Kapitel 3 und Kapitel 4 erläuterten Grundlagen. Es wird auf die gestellten Anforderungen, die darauf basierende Konzeption sowie die resultierende Implementierung eingegangen. Am Ende erfolgt eine Bewertung des entwickelten Verfahrens und ein Ausblick auf mögliche Erweiterungen und Verbesserungen.

5.1 Anforderungen

5.1.1 Erzeugung konsistenter Dreiecksnetze

Die erste und wichtigste Anforderung an das Verfahren ist die Erzeugung konsistenter Dreiecksnetze. Diese Anforderung wird an alle Triangulationsverfahren gestellt. Die Bedingungen, welche ein konsistentes Dreiecksnetz erfüllen muß, sind in Abschnitt 3.2 beschrieben. Ausgangspunkt für die Triangulation ist eine Menge von Punkten im zweidimensionalen oder dreidimensionalen Raum.

5.1.2 Adaptivität

Das Verfahren soll adaptiv sein. Die Triangulation einer Punktwolke ist somit als Adaptionsprozess zu verstehen. Ein Dreiecksnetz wird dabei an eine vorgegebene Punktwolke angepaßt.

5.1.2.1 Topologische Adaption

Das generierte Dreiecksnetz soll eine variable Topologie besitzen. Diese muß während des Adaptionprozesses veränderbar sein. Das heißt, daß das Einfügen und Löschen von Vertices des Dreiecksnetzes während der Adaption möglich ist. Die Abbildung der Punktwolke auf das Netz soll topologieerhaltend sein.

5.1.2.2 Geometrische Adaption

Das Netz soll auch geometrisch an die Punktwolke angepaßt werden. Dafür müssen die Positionen der Vertices des Netzes in Abhängigkeit von der geometrischen Form und Lage der Punktwolke verändert werden. Das Ziel ist eine möglichst genaue Triangulation der Punktwolke.

5.1.3 Flexible Startinitialisierung

Es ist die Aufgabe des Algorithmus, über eine gegebene Punktwolke ein Dreiecksnetz zu erzeugen. Dies kann durch eine Triangulation oder durch die Anpassung eines vorgefertigten Dreiecksnetzes erfolgen. Das setzt eine flexible Initialisierung des Verfahrens voraus.

5.1.4 Sinnvolle Abbruchkriterien

Der Adaptionprozess muß über geeignete Abbruchkriterien verfügen. Diese sind beispielsweise der maximale Abstand zwischen gegebener Punktwolke und generiertem Dreiecksnetz, die maximale Anzahl der Knoten im erzeugtem Dreiecksnetz oder die maximale Kantenlänge der gebildeten Dreiecke.

5.1.5 Integration in bestehende Softwarebibliothek

Bei der Implementierung ist eine Integration in die bestehende Softwarebibliothek zu gewährleisten, um die Verwendbarkeit und Wartbarkeit zu erleichtern. Dies beinhaltet die Nutzung bestehender Softwaremodule sowie die bestmögliche Kompatibilität der neu implementierten Module. Die bestehende Klassenbibliothek ist in der Programmiersprache C++ geschrieben. Sie enthält unter anderem verschiedene Module zur Verwaltung dreidimensionaler Objekte wie Punktwolken, Dreiecksnetze oder analytische Objekte. Außerdem stellt sie verschiedene Werkzeuge zur Manipulation und Erzeugung dieser Objekte zur Verfügung. Die zu implementierenden Klassen sind ebenso in dieser Sprache zu entwickeln, um die volle Kompatibilität und Verwendung von objektorientierten Programmier Techniken zu ermöglichen.

5.2 Konzeption und Entwurf

Die Entwicklung über eine Konzeption und einen Entwurf ist nötig, um aus den gestellten Anforderungen sowie den erläuterten Grundlagen ein Gesamtkonzept zu entwickeln. Dies ermöglicht es den Überblick über die steigende Komplexität zu behalten. Das erarbeitete Konzept soll die grundlegenden Designentscheidungen widerspiegeln und so als Wegweiser für die spätere Implementierung dienen.

5.2.1 Grunddesign

Aus den gestellten Anforderungen geht hervor, daß das zu entwickelnde Triangulationsverfahren adaptiv arbeiten soll. Die Triangulation der Punktwolke erfolgt deshalb in Form eines Adaptionsprozesses. Das Ergebnis eines solchen Prozesses ist ein konsistentes Dreiecksnetz. Es geht also darum, eine netzartige Struktur an eine Menge von Eingabedaten anzupassen. Dies legt die Verwendung eines neuronalen Netzes als Grundstruktur nahe. Ein NN ist eine solche netzartige Struktur aus Zellen (Vertizes) und Verbindungen zwischen den Zellen (Kanten). Es wird verwendet um die Daten eines Eingaberaumes auf einen Ausgaberaum abzubilden. Die Punkte der Punktwolke definieren in diesem Fall den Eingaberaum, wogegen das generierte Dreiecksnetz den Ausgaberaum definiert.

Wie in Kapitel 4 beschrieben ist, gibt es eine Vielzahl unterschiedlicher Architekturen neuronaler Netze. Die verschiedenen Anforderungen müssen genau betrachtet und analysiert werden, um eine für diese Anwendung geeignete Netzarchitektur zu finden.

Das Triangulationsverfahren hat die Aufgabe, Dreiecksnetze zu erzeugen. Konsistente Dreiecksnetze bestehen ausschließlich aus Dreiecken, welche wiederum aus Eckpunkten und Kanten zusammengesetzt sind. Wie in Abschnitt 3.2 erläutert wurde, sind einzelne Kanten oder Eckpunkte nicht erlaubt. Ein Dreiecksnetz ist also ein Netz, welches ausschließlich aus Bestandteilen mit gleicher Dimensionalität (Dreiecken) bestehen. Weiterhin wird gefordert, daß die Netze eine variable Topologie aufweisen. Das heißt, die Nachbarschaften zwischen Vertizes müssen veränderbar sein. Diese werden durch die Verbindungen zwischen den Vertizes, den einzelnen Dreieckskanten, definiert. Dadurch können topologieverändernde Operationen, wie das Einfügen und Löschen von Knoten, durchgeführt werden. Diese Operationen beeinflussen jedoch nicht nur die Topologie, sondern auch die Anzahl der Vertizes des Dreiecksnetzes. Wird dieser Sachverhalt auf ein NN übertragen, so muß dieses eine variable Anzahl von Zellen ermöglichen.

Zusammengefaßt lauten die Eigenschaften des gesuchten NN wie folgt:

- feste Polygondimensionalität
- variable Topologie

- variable Zellenanzahl

Diese Eigenschaften wurden in der Bewertung der in Kapitel 4 vorgestellten neuronalen Netze verglichen. Für die erwünschten Eigenschaften, kommen die Modelle des *Growing Neural Gas* sowie der *Wachsenden Zellstrukturen* in die engere Wahl. Da jedoch bei dem ersteren keine feste Dimensionalität der polygonalen Bestandteile vorhanden ist, bleibt das Modell der *Wachsenden Zellstrukturen* übrig.

Dieses NN ist wie die anderen vorgestellten Modelle einschichtig. Dies hat den Vorteil, daß das zu generierende Dreiecksnetz direkt aus der Struktur des neuronalen Netzes am Ende der Lernphase abgelesen werden kann. Somit gibt es eine direkte Analogie zwischen den Zellen des neuronalen Netzes und den Vertices des Dreiecksnetzes, sowie den Verbindungen zwischen den Zellen und den Dreieckskanten.

5.2.2 Erweiterungen

Das Modell der *Wachsenden Zellstrukturen* wurde im vorigen Abschnitt als Basis für die Entwicklung des adaptiven Triangulationsverfahrens ausgewählt. Für die Erfüllung aller gestellten Anforderungen ist es jedoch nötig, einige Erweiterungen vorzunehmen.

5.2.2.1 Verbinden von Zellen

Wie bei der Beschreibung der *Wachsenden Zellstrukturen* dargestellt wurde, besitzt das Verfahren eine variable Topologie. Es verfügt über Teilschritte, in denen neue Zellen in das bestehenden Netz eingefügt oder schon vorhandene Zellen aus ihm entfernt werden können. Das Einfügen neuer Zellen ermöglicht es dem Netz, zu wachsen und sich auszubreiten. Durch das Entfernen von Zellen erhält es die Fähigkeit, sich in Teilnetze aufzuspalten. Dies ist vor allem von Vorteil, wenn die Daten im Eingaberaum stark separiert sind.

Es sollte aber auch die Möglichkeit bestehen, daß sich separate Netze oder zwei angrenzende Teile desselben Netzes verbinden können. Das Verfahren ist dahingehend zu erweitern, daß neue Verbindungen zwischen den Zellen des Netzes erzeugt werden können. Dieser Schritt muß wieder ein konsistentes Netz zum Ergebnis haben, welches ausschließlich Dreieckssimplexe enthält. Es reicht also nicht, jeweils eine einzelne Verbindung zwischen zwei Zellen der verschiedenen Teilnetze zu bilden. Vielmehr muß aus den neu erzeugten Verbindungen mindestens ein neuer Simplex entstehen.

Das Verbinden von Zellen soll durch Kombination des Modells der *Wachsenden Zellstrukturen* mit dem *Competitive Hebbian Learning* ermöglicht werden. Bei diesem Verfahren wird zwischen den zwei gewinnenden Zellen für ein gegebenes Eingabesignal eine Verbindung erzeugt, falls diese noch nicht vorhanden ist (siehe Abschnitt 4.4.5). Es muß aber noch

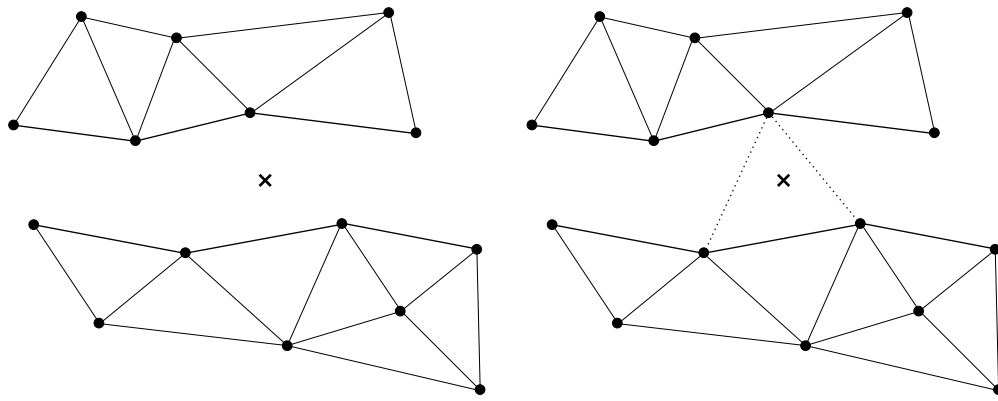


Abbildung 5.1: Verbinden zweier Teilnetze

erweitert und angepaßt werden, um den Bedingungen für die Erzeugung eines konsistenten Netzes zu genügen.

Für die Bildung eines Simplex durch die neu erzeugten Verbindungen sind mehr als zwei Zellen notwendig. Deswegen sollten die Verbindungen immer zwischen den drei für ein gegebenes Eingangssignal gewinnenden Zellen eingefügt werden. Damit die beiden Teilnetze sich nur in den Randbereichen verbinden können, müssen darüber hinaus alle drei Gewinner Randzellen des jeweiligen Netzes sein.

In Abbildung 5.1 ist ein idealisierter Fall für das Verbinden zweier separater Teilnetze dargestellt. Auf der linken Seite sind diese zusammen mit einem Eingangssignal (in Form eines Kreuzes) abgebildet. Auf der rechten Seite sind die zu erzeugenden Verbindungen zwischen den drei Gewinnerzellen der beiden Netze eingezeichnet.

Das zu entwickelnde adaptive Triangulationsverfahren ist demnach eine Verbindung zweier vorgestellter Netzmodelle. Es soll die Vorteile beider Verfahren in sich vereinigen und dadurch die gestellten Anforderungen erfüllen. Durch die Verwendung der *Wachsenden Zellstrukturen* und des *Competitive Hebbian Learning* wird ein optimales Wachsen und Ausbreiten des Netzes ermöglicht. Es ähnelt deswegen dem Modell des *Growing Neural Gas*. Es sind jedoch Unterschiede zwischen beiden vorhanden. Diese zeigen sich bei der Polygondimensionalität sowie der Möglichkeit, Zellen aus dem Netz zu löschen.

5.2.2.2 Adaption von Teilnetzen

Das Verfahren hat nicht nur die Aufgabe neue Dreiecksnetze zu erzeugen, sondern auch vorgefertigte Dreiecksnetze anzupassen. Zusätzlich soll die Möglichkeit geschaffen werden, auch lokale Teilnetze zu adaptieren, um die Anwendung möglichst flexibel und universell zu gestalten. In diesem Fall sind aus dem Gesamtnetz Bereiche auszuwählen, welche während der Lernphase nicht verändert werden.

Während der Adaption ist zu entscheiden, welche Zellen des Netzes Bestandteil des zu adaptierenden Teilnetzes sind. Die einzelnen Zellen müssen jeweils eine Kennzeichnung tragen, welche besagt, ob diese Zelle adaptiert werden darf. Diese ist bei allen Operationen, die auf dem Netz ausgeführt werden, zu berücksichtigen.

5.2.3 Modularisierung

Beim Entwurf des Algorithmus wurde auf eine gute Modularisierung in einzelne Komponenten Wert gelegt. Dies ermöglicht eine verbesserte Wartbarkeit und Erweiterbarkeit der einzelnen Programmteile. Damit wird eine Art Rahmenwerk bereitgestellt, um ähnlich arbeitende Adaptionenverfahren schnell entwickeln zu können. Die einzelnen Module müssen identifiziert und analysiert werden, um die Implementierung modular zu gestalten.

Die verschiedenen Algorithmen der in Kapitel 4 vorgestellten Netzmodelle können in verschiedene Phasen beziehungsweise Aufgabengruppen unterteilt werden. Diese sind wiederum in verfahrensabhängige und verfahrensunabhängige Aufgabengruppen zu unterscheiden. Die Unterteilung ist notwendig, um zu einer Abstraktion und Gliederung der verwendeten Algorithmen zu kommen. Es ist sinnvoll, diejenigen Teile zu finden, welche unabhängig von den verschiedenen Netzmodellen und Algorithmen sind. Durch geeignete Mittel der Objektorientierung sollen dann verschiedene Softwaremodule entstehen, welche die Grundlage für verschiedene Adaptionenverfahren bilden.

5.2.3.1 Verfahrensabhängige Aufgaben

Verfahrensabhängige Aufgaben sind entweder auf den jeweiligen Algorithmus speziell abgestimmt oder werden nur bei einem bestimmten Algorithmus ausgeführt. Für diese Teilaufgaben macht es keinen Sinn, eigene abstrakte Module zu entwickeln. Diese müssen deshalb immer für den jeweiligen Algorithmus neu implementiert werden.

Vom Verfahren abhängige Teile sind beispielsweise die Initialisierung, die Adaption der einzelnen Zellen des neuronalen Netz oder geeignete Abbruchkriterien.

Die Initialisierung haben alle neuronalen Netze als Teil des Adaptionenprozesses gemeinsam. Welche Schritte dabei jedoch ausgeführt werden, ist jeweils von den Eigenschaften des Netzes abhängig. Unterschiede gibt es zum Beispiel in der Menge der Knoten und Kanten des jeweiligen Netzes.

Die eigentliche Anpassung der einzelnen Zellen an ein beliebiges Eingangssignal ist der spezifische Teil eines Adaptionenprozesses. Hier spielen Kriterien wie Lernrate, Topologie und Netzdimensionalität des jeweiligen Netzes eine große Rolle. Es ist kaum möglich diesen Teil zu verallgemeinern, da er zu stark auf die Eigenschaften des jeweiligen Netzes abgestimmt ist.

Die für ein NN geeigneten Abbruchkriterien sind für verschiedene Netze ebenfalls sehr unterschiedlich. Diese sind meist sehr genau auf das jeweilige Ziel des Adaptionsprozesses abgestimmt. Das Lernziel kann bei jedem Netz ein anderes sein und ist deshalb sehr netzspezifisch.

5.2.3.2 Verfahrensunabhängige Aufgaben

Verfahrensunabhängige Aufgaben und Teilalgorithmen sind bei allen Netzmodellen in gleicher oder sehr ähnlicher Weise vorhanden. Die Abstraktion und Generalisierung solcher Teile ist leicht zu verwirklichen und erhöht die Wiederverwendbarkeit für verschiedene Algorithmen.

Beispiele für verfahrensunabhängige Teilaufgaben sind die Verwaltung der Daten des neuronalen Netzes, die Verwaltung der Punktwolke als Menge der Eingabedaten, sowie die Propagierungs- beziehungsweise Abstandsfunktion der Zellen des neuronalen Netzes.

Bei allen betrachteten Modellen von NN ist immer eine Menge von Zellen als Teil des Netzes vorhanden. Bei einigen kommt noch die Menge der Verbindungen zwischen den Zellen hinzu. Es ist also sinnvoll ein Modul zu entwickeln, welches die Daten und Operationen eines NN beinhaltet. Dieses kann dann als Grundlage für die verschiedenen Netzmodelle dienen.

Bei dem hier angestrebten Triangulationsverfahren soll ein Netz an eine Punktwolke adaptiert werden. Die Punktwolke stellt hierbei die Menge der Eingabedaten dar. Die verschiedenen Netzmodelle haben die Generierung eines geeigneten Eingabesignals aus dieser Menge der Eingabedaten gemeinsam. Die Aufgabe der Verwaltung der Punktwolke und das Generieren eines Eingabesignals sind also relativ unabhängig von dem jeweiligen Netzmodell.

Bei den in Kapitel 4 vorgestellten Netzmodellen wird als Propagierungsfunktion der Zellen eine Abstandsfunktion verwendet. Diese berechnet den Abstand zwischen dem generierten Eingabesignal und dem Gewichtsvektor der Zelle. Mit Hilfe dieses Ergebnisses wird die Aktivität einer Zelle berechnet. Die Ermittlung des Abstandes läuft meist darauf hinaus, zu einem bestimmten Eingabesignal die am besten korrespondierende Zelle oder Zellen zu suchen.

5.2.4 Klassenstruktur

Aus den im vorigen Abschnitt umrissenen Teilaufgaben kann eine geeignete Klassenstruktur entwickelt werden. Diese sollte die Gliederung der verschiedenen Algorithmen der Netzmodelle widerspiegeln. Die verfahrensunabhängigen Teile werden in eigene abstrakte Module beziehungsweise Klassen ausgelagert. Die verfahrensabhängigen Teile sind jeweils in einer eigenen Klasse für jeden zu implementierenden Algorithmus zusammenzufassen.

Ein wichtiger Aspekt ist die Verwendung des Model-View-Control-Modells (MVC) bei der Programmierung. Dieses Modell schreibt die Trennung von Algorithmus beziehungsweise Programmlogik (Model), graphischer Benutzerschnittstelle (View) und Programmsteuerung (Control) vor. Die hier zu entwickelnden Klassen gehören ausschließlich dem ersten Teil des MVC-Modells an. Deswegen ist zu beachten, daß diese Klassen keinerlei Programmcode für die Ansteuerung der Benutzerschnittstelle oder die Programmsteuerung enthalten.

Weiterhin ist beim Entwurf der Klassenstruktur die Integration in die bestehende Klassenbibliothek wichtig. Die in der Klassenbibliothek enthaltenen Klassen, Strukturen und Funktionen sind soweit wie möglich zu verwenden. Dadurch kann die Wiederverwendbarkeit erhöht und eine verbesserte Wartbarkeit der Programmteile sichergestellt werden. Folgende Module dienen als Verbindung zur bestehenden Softwarebibliothek:

<i>C3DUserObj</i>	Diese Klasse ist die Grundlage für die Entwicklung neuer Datenklassen. Sie stellt unter anderem Funktionen bereit, um die enthaltenen Daten in einem OpenGL-Kontext zu visualisieren. Abgeleitete Klassen können damit ohne Änderung des Programmcodes der darstellenden Programme verwendet werden.
<i>C3DCloudObj</i>	Sie dient als Datenklasse zur Verwaltung der Punktwolke.
<i>CG3DOctTree</i>	Dies ist die Basisklasse für einen Octtree. Sie stellt Funktionen zum Einsortieren von dreidimensionalen Vektoren und zur Verwaltung der enthaltenen Octtree-Würfel bereit.
<i>CG3DOctCell</i>	Die Daten und Funktionen eines einzelnen Würfels im Octtree werden durch diese Klasse zur Verfügung gestellt. Dieser verwaltet die Indizes der Vektoren, deren Position in seinen Bereich fällt. Jeder Würfel umfaßt eine beliebige Anzahl von Vektorindizes. Die zugehörige Octtree-Klasse erzeugt und verwendet die Objekte dieser Klasse.

Die entwickelte Klassenstruktur ist in Abbildung 5.2 dargestellt. Die Beziehungen und Funktionen der einzelnen Module sollen im Folgenden erklärt werden.

Die Klasse *CAdaption* ist ein zentraler Bestandteil dieser Struktur. Sie dient als Basis für die Implementierung eines beliebigen Adaptionsverfahrens. Die Klasse wird abstrakt definiert und soll ein generisches Interface für alle davon abgeleiteten Klassen bereitstellen. Sie stellt den kleinsten gemeinsamen Nenner für verschiedene Adaptionsverfahren dar und soll deren unspezifische Funktionalitäten implementieren. Wie aus dem Diagramm ersichtlich ist, hat die *CAdaption* Beziehungen zu weiteren Klassen.

Die wichtigste dieser assoziierten Module ist die Klasse *CNetwork*. Diese beinhaltet die Daten eines NN in Form von Zellen und Verbindungen. Außerdem stellt sie einige Operationen

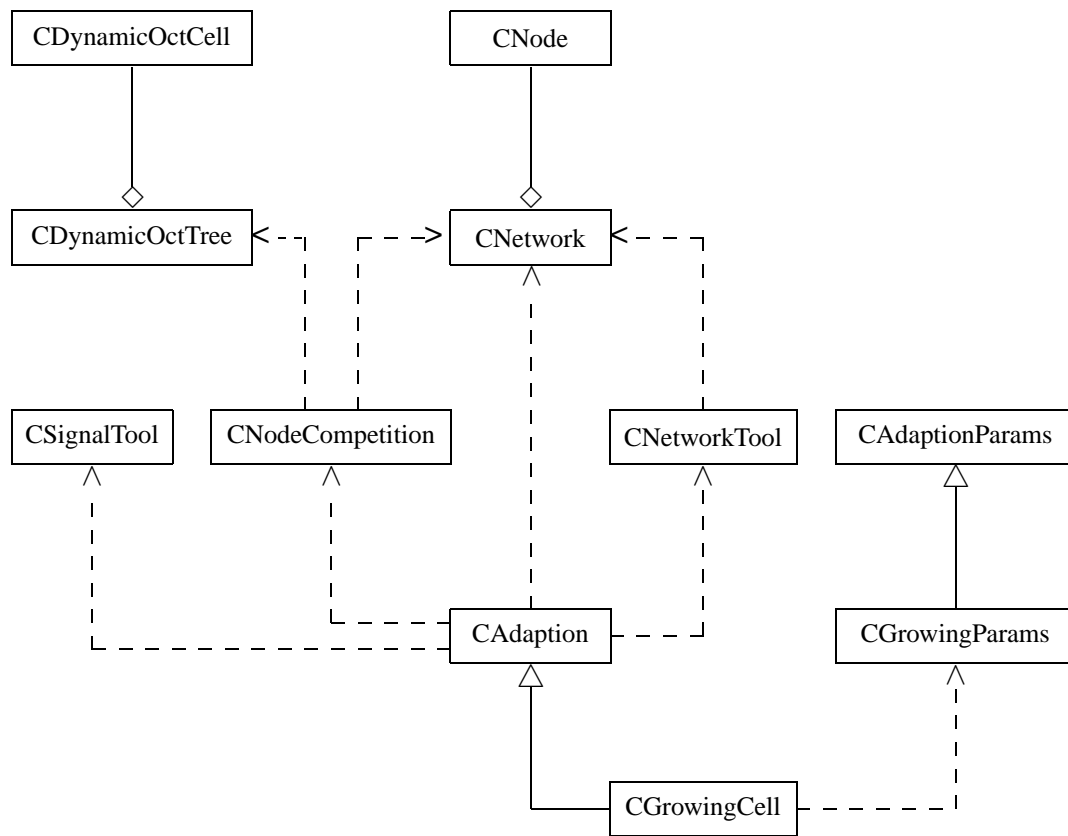


Abbildung 5.2: Kollaborationsdiagramm

für die Manipulation des Netzes zur Verfügung. Die Daten sollen in bestehenden Programmen dargestellt und teilweise interaktiv verändert werden können. Es war notwendig die Klasse von *C3DUserObj* abzuleiten, um die Visualisierung dieser Daten ohne Änderung des Programmcodes der darstellenden Programme zu erreichen. Dadurch ist es möglich, *CNetwork* in die Menge der vorhandenen Datenklassen für dreidimensionale Objekte wie beispielsweise Punktwolken oder Dreiecksobjekte einzufügen und gleichermaßen zu verwenden.

Die Daten eines einzelnen Netzknotens beziehungsweise einer Zelle sind in der Klasse *CNode* zusammengefaßt. Diese bestehen beispielsweise aus dem Gewichtsvektor und den Verbindungen zu anderen Knoten. *CNode* stellt verschiedene Funktionen zur Manipulation dieser Daten zu Verfügung.

Die Klasse *CSignalTool* wird ebenfalls von *CAdaption* verwendet. Sie soll die Verwaltung der Punktwolke als Menge der Eingabedaten übernehmen. Die Daten der Punktwolke werden über eine eigene Klasse *C3DCloudObj* aus der Klassenbibliothek verwaltet und be-

reitgestellt. Dadurch wird wiederum die Integration der vorhandenen Module aus der Softwarebibliothek ermöglicht. *CSignalTool* stellt darüber hinaus verschiedene Funktionen zur Auswahl von Eingabesignalen aus der Punktwolke zu Verfügung.

Es wird eine neue Klasse benötigt, um die Anzahl der in *CNetwork* enthaltenen Basisoperationen zur Manipulation des Netzes möglichst gering zu halten. Diese soll verschiedene allgemeine Operationen zur Analyse und Veränderung der Daten eines Netzes beinhalten. Diese Funktionalitäten sind in der Klasse *CNetworkTool* zusammengefaßt. Diese enthält eine Referenz auf ein Netz, auf welchem die definierten Operationen ausgeführt werden.

Der Wettbewerb zwischen den Zellen wird über die Klasse *CNodeCompetition* ermöglicht. Dieser beschränkt sich bei den verschiedenen Adaptionprozessen hauptsächlich auf die Suche der nächsten Zellen zu einem Eingabesignal. *CNodeCompetition* nutzt die Funktionalität eines Octtree für diese Suche. Außerdem enthält sie jeweils Referenzen auf eine Punktwolke sowie auf ein Netz.

Der Octtree wird von der Klasse *CDynamicOctTree* bereitgestellt. Sie ist von der Basis-Klasse *CG3DOctTree* aus der Klassenbibliothek abgeleitet, welche Funktionen zur Verwaltung von dreidimensionalen Vektoren zur Verfügung stellt. *CDynamicOctTree* unterstützt die Verwendung einer variablen Anzahl von Vektoren. Hinzu kommt, daß die einsortierten Vektoren nicht statisch sein müssen. Dadurch ist ein Verschieben von Vektoren innerhalb des Octtree möglich.

Die Klasse *CDynamicOctCell* enthält die Daten eines einzelnen Octtree-Würfels. Diese Klasse ist von *CG3DOctCell* abgeleitet, welche einen Würfel im *CG3DOctTree* darstellt. Eine Octtree-Implementation besteht also immer aus der eigentlichen Octtree-Klasse sowie der zugehörigen Octtree-Würfel-Klasse.

Die für die verschiedenen Adaptionsverfahren spezifischen Funktionen und Daten werden jeweils in einem spezialisierten Modul implementiert. Die für das zu entwickelnde adaptive Triangulationsverfahren spezifische Klasse ist *CGrowingCell*. Sie ist von *CAdaption* abgeleitet.

Die Parameter eines Adaptionsverfahrens werden in einer eigenen Klasse zusammengefaßt. Für die Basisklasse *CAdaption* gibt es die zugehörigen Klasse *CAdaptionParams*. Diese beinhaltet einige generelle Parameter, welche für verschiedene Verfahren sinnvoll sind. *CGrowingParams* erbt diese und enthält zusätzlich die für das zu entwickelnde Adaptionsverfahren spezifischen Parameter.

Wie in der kurzen Beschreibung der einzelnen Module bereits erwähnt wurde, gibt es mehrere Verbindungspunkte zu der bestehenden Softwarebibliothek. Dies zeigt sich beim Modul der Punktwolke ebenso wie bei der Grundlage für die Octtree-Funktionalitäten.

5.3 Implementierung

In diesem Abschnitt wird auf die Umsetzung des adaptiven Triangulationsverfahrens näher eingegangen. Dazu gehört eine detaillierte Beschreibung sowie der komplette Algorithmus des Verfahrens. Weiterhin werden die bei der Umsetzung des Verfahrens aufgetretenen Probleme geschildert und die dafür gefundenen Lösungen vorgestellt.

5.3.1 Verfahrensbeschreibung

In Abschnitt 5.2.1 wurde bereits analysiert, welches der vorgestellten Modelle neuronaler Netze als Grundlage für dieses Verfahren am geeignetsten ist. Dabei sind das Modell der *Wachsenden Zellstrukturen* sowie das Modell des *Competitive Hebbian Learning* ausgewählt worden.

5.3.1.1 Algorithmus

Der Algorithmus der adaptiven Triangulation kann in folgende Schritte unterteilt werden. Der Einfachheit wegen sind die einzelnen Teilschritte bewußt recht allgemein und kurz gehalten. Sie werden in den nächsten Abschnitten detailliert beschrieben. Der komplette Algorithmus ist in Anhang A als Programmablaufplan dargestellt.

1. Initialisieren der Menge C und der Menge K der Verbindungen zwischen den Zellen
2. Generieren eines zufälligen Eingabesignals ξ aus dem Eingaberaum
3. Bestimmen der Gewinnerzellen für das Eingabesignal ξ
4. Adaptieren der Gewichtsvektoren des 1.Gewinners und seiner direkten topologischen Nachbarn
5. Erzeugen von Verbindungen zwischen den drei Gewinnern, falls diese noch nicht existieren
6. Einfügen einer neuen Zelle, wenn die Anzahl der generierten Eingabesignale ein ganzzahliges Vielfaches eines Parameters λ_1 ist
7. Löschen von Zellen, wenn die Anzahl der generierten Eingabesignale ein ganzzahliges Vielfaches eines Parameters λ_2 ist
8. Verringern des lokalen Fehlers aller Zellen um einen Bruchteil β :

$$\Delta E_c = -\beta E_c \quad (\forall c \in C)$$

9. Wiederholen ab Schritt 2, bis das gewählte Abbruchkriterium erreicht ist

Schritt 1 - Initialisierung

Beim Modell der *Wachsenden Zellstrukturen* erfolgt die Initialisierung der Menge der Zellen und der Menge der Verbindungen derart, daß das Netz aus einem einzelnen k -dimensionalen Simplex besteht. Die Netzdimensionalität für das adaptive Triangulationsverfahren ist auf $k = 2$ festgelegt. Deswegen müssen drei ($k + 1$) Zellen mit ihren Verbindungen initialisiert werden.

$$C = \{c_1, c_2, c_3\}$$

Die Gewichtsvektoren sind auf zufällig gewählte Werte aus dem Eingaberaum zu setzen. Da die zu triangulierende Punktwolke den Eingaberaum definiert, werden drei zufällig gewählte Punkte als Gewichtsvektoren verwendet. Nun werden die einzelnen Zellen miteinander verbunden, so daß sie ein Dreieck bilden.

$$K = \{(c_1, c_2), (c_1, c_3), (c_2, c_3)\}$$

Das Verfahren soll auch die Adaption vorgefertigter Dreiecksnetze ermöglichen. Es ist also notwendig, das NN mit den Punkt- und Kanteninformationen eines Dreiecksnetzes zu initialisieren. Die Anzahl der Zellen entspricht der Anzahl von Punkten des Dreiecksnetzes. Die Menge der Verbindungen wird aus der Menge der Dreieckskanten initialisiert. Jede Zelle enthält den Ortsvektor des korrespondierenden Punktes als Gewichtsvektor.

Schritt 2 - Auswahl des Eingabesignals

Die Auswahl der Eingabesignale aus der Punktwolke erfolgt zufällig. Die Punkte der Punktwolke werden indiziert, um nicht jedes zufällig ausgewählte Eingabesignal neu zu berechnen. Dieser Index wird einmal vor dem Start des Adaptionsprozesses berechnet und enthält die Indizes der Punkte in einer zufälligen Reihenfolge. Zusätzlich kann damit sichergestellt werden, daß jeder Punkt der Punktwolke in den Adaptionsprozess mit eingeht.

Schritt 3 - Wettbewerb zwischen Zellen

Der Wettbewerb der Zellen untereinander wird über ihren euklidischen Abstand zum Eingabesignal entschieden. Wie in der Konzeption des Verfahrens erläutert ist, werden für jedes generierte Eingabesignal ξ die drei Gewinner gesucht. Dies sind die Zellen s_1, s_2, s_3 deren Gewichtsvektoren den kleinsten Abstand zum Eingabesignal ξ haben.

$$\begin{aligned} \|\xi - w_{s_1}\| &\leq \|\xi - w_c\| \quad (\forall c \in C) \\ \|\xi - w_{s_2}\| &\leq \|\xi - w_c\| \quad (\forall c \in C \setminus \{s_1\}) \\ \|\xi - w_{s_3}\| &\leq \|\xi - w_c\| \quad (\forall c \in C \setminus \{s_1, s_2\}) \end{aligned}$$

Für jede Zelle des Netzes müßte die Abstandsfunktion berechnet werden, um die Gewinner zu bestimmen. Bei einigen wenigen Zellen ist dies ohne großen Rechenaufwand möglich. Dieser wächst jedoch mit steigender Anzahl der Zellen. Hier bietet sich die Verwendung eines Octtree an, um die Suche nach den Zellen mit dem geringsten Abstand zum Eingabesignal zu beschleunigen. Dabei sind die Punkte der Punktwolke und die Zellen des NN in den gleichen Octtree einsortiert. Ausgangspunkt für die Suche ist der Octtree-Würfel, welcher das Eingabesignal enthält. Dieser und die direkten Nachbarwürfel werden nun untersucht. Für alle Zellen, die in diesen Würfeln enthalten sind, wird der Abstand zum Eingabesignal berechnet. Nach der Ermittlung der drei Zellen mit dem geringsten Abstand kann die eigentliche Adaption ausgeführt werden. Sind in der direkten Nachbarschaft des Startwürfels keine Zellen vorhanden, dann wird die Suchumgebung vergrößert. Dies geschieht solange, bis die drei gewinnenden Zellen bestimmt sind.

Schritt 4 - Adaption von Zellen

Die Anpassung einzelner Zellen lehnt sich eng an die Adaption der *Wachsenden Zellstrukturen* an. Die Gewinnerzelle s wird in die Richtung des Eingabesignals ξ gezogen. Die Stärke der Bewegung ist abhängig vom Abstand zwischen der Zelle und dem Eingabesignal und von der vorher festgelegten Lernrate ϵ_b . Anschließend werden auch die direkten topologischen Nachbarn des Gewinners in Richtung des Eingabesignals bewegt. Ihre Adaptionsstärke wird von der Lernrate ϵ_n beeinflusst.

$$\begin{aligned}\Delta w_s &= \epsilon_b(\xi - w_s) \\ \Delta w_i &= \epsilon_n(\xi - w_i) \quad (\forall i \in N_s)\end{aligned}$$

Abschließend wird der lokale Fehler der Gewinnerzelle s um das Quadrat des Abstandes zum Eingabesignal erhöht.

$$\Delta E_s = \|\xi - w_s\|^2$$

Abweichend vom Vorgehen der *Wachsenden Zellstrukturen*, wird die Bewegung des Gewinners eingeschränkt. Die obere Schranke ist dessen halbe mittlere Kantenlänge. Dies bedeutet, daß er sich nur innerhalb seiner approximierten Voronoi-Region bewegen kann.

Schritt 5 - Verbinden von Zellen

Das Verbinden einzelner Zellen während des Adaptionsprozesses wird über die Erweiterung der *Wachsenden Zellstrukturen* durch das Modell des *Competitive Hebbian Learning* ermöglicht. Dadurch sollen sich separate Teilnetze wieder zu einem Gesamtnetz zusammenschließen können. Wie im vorhergehenden Text beschrieben ist, werden für jedes auftretende Eingabesignal jeweils drei Gewinnerzellen ermittelt. Zwischen diesen sollen neue

Verbindungen erzeugt werden, falls diese noch nicht vorhanden sind. Diese Verbindungen sollen dann mindestens einen neuen Simplex bilden.

Da die Teilnetze ausschließlich an den Randbereichen zusammenwachsen sollen, müssen alle drei Gewinnerzellen auch Randzellen sein. Eine Randzelle entspricht einem Randvertex eines Dreiecksnetzes, wie in Abschnitt 3.2 erläutert ist. Handelt es sich nicht um Randzellen, so wird der Vorgang abgebrochen, da sonst beim Einfügen neuer Verbindungen Netzinkonsistenzen entstehen können.

Nun werden die bestehenden Verknüpfungen zwischen den gewinnenden Zellen untersucht. Es können genau vier verschiedene Fälle auftreten:

1. Es bestehen keinerlei direkte Verbindungen zwischen den Zellen.
2. Es besteht genau eine direkte Verbindung zwischen den Zellen.
3. Es bestehen genau zwei direkte Verbindungen zwischen den Zellen.
4. Es bestehen genau drei direkte Verbindungen zwischen den Zellen.

Der erste Fall soll hier nicht beachtet werden, da durch die fehlenden Verbindungen zwischen den Zellen nicht sichergestellt werden kann, daß sich diese Zellen in einer topologischen Nähe zueinander befinden. Da durch das Verknüpfen der Zellen ein neuer Simplex entstehen soll, ist es vorteilhaft, wenn sich dieser in die schon vorhandene Topologie der Teilnetze einpaßt.

Der zweite und der dritte Punkt stellen die für das Verbinden interessanten Fälle dar. Hier bestehen schon einzelne Verbindungen, welche aber noch keinen vollständigen Simplex formen. Zwei beziehungsweise alle drei der Zellen befinden sich damit schon in einer gewissen topologischen Nähe zueinander. In beiden Fällen können die fehlenden Verbindungen erzeugt und damit ein neuer Simplex gebildet werden. Der zweite Punkt würde beispielsweise beim Zusammentreffen zweier separater Teilnetze (wie in Abbildung 5.1) auftreten. Ein Beispiel für den dritten Fall wäre das Vorhandensein von Einschnitten oder Konkavitäten am Rand des Netzes (siehe Abbildung 5.3).

Für den vierten Punkt erübrigt sich ein Handeln. Die Zellen sind hier schon vollständig miteinander verbunden und bilden somit bereits einen vollständigen Simplex.

Die letzte zu erfüllende Bedingung ist die Lage des korrespondierenden Eingabesignals innerhalb des zu bildenden Simplex. Dies stellt sicher, daß der neu erzeugte Simplex eine Abbildung für diesen Punkt aus dem Eingaberaum ist. Er wird also nur dort erzeugt, wo auch korrespondierende Eingabesignale vorhanden sind. Zwei separate Teilnetze schließen sich nur zusammen, wenn der Eingaberaum in diesen Bereichen definiert ist. Dadurch werden ausschließlich solche Verbindungen erzeugt, welche die Abbildung der Punktwolke auf das Netz verbessern.

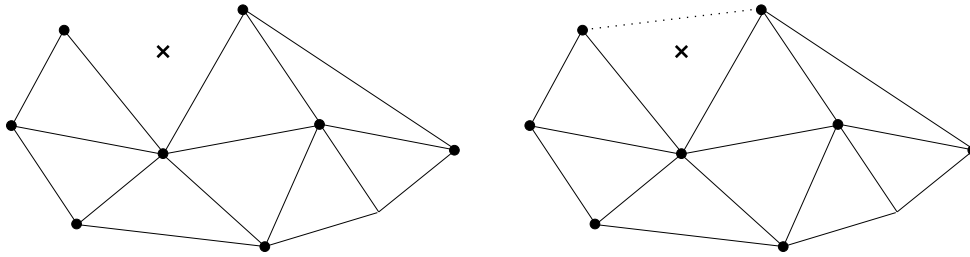


Abbildung 5.3: Schließen einer Konkavität

Schritt 6 - Einfügen von Zellen

Das Einfügen neuer Zellen in das Netz entspricht dem Einfügevorgang im Modell der *Wachsenden Zellstrukturen* (siehe Abschnitt 4.4.3). Dies wird in vorher festgelegten Intervallen beziehungsweise nach einer bestimmten Anzahl von Eingabesignalen durchgeführt. Zuerst muß diejenige Zelle q ermittelt werden, welche den größten lokalen Fehler aufweist. Dieser Fehler enthält den quadratischen Abstand zu allen Eingabesignalen, für welche diese Zelle Gewinner war. Die Zelle mit dem größten lokalen Fehler markiert damit eine Stelle im Netz, an der die Abbildung des Eingaberaumes am schlechtesten ist. Hier wird die neue Zelle eingefügt. Dies erfolgt durch Aufspalten der Verbindung zwischen q und seinem am weitesten entfernten Nachbarn f . Der Gewichtsvektor der neuen Zelle r wird aus den Gewichtsvektoren der Zellen q und f interpoliert.

$$w_r = (w_q + w_f)/2$$

Die Verbindung zwischen q und f wird entfernt und die neue Zelle r wird mit allen gemeinsamen Nachbarn von q und f verbunden. Die lokalen Fehler der Nachbarn der neuen Zelle r werden abhängig von der Anzahl der Nachbarn und einem Parameter α verringert. Der lokale Fehler und die Anzahl der empfangenen Signale von r werden auf die Durchschnittswerte der Nachbarn gesetzt.

Schritt 7 - Löschen von Zellen

Das Löschen von Zellen in Gebieten mit geringer Signaldichte im Eingaberaum erfolgt wie beim Modell der *Wachsenden Zellstrukturen* (siehe Abschnitt 4.4.3). Da die Dichte der Eingabesignale aber meist nicht bekannt ist, muß diese approximiert werden. Dieser Schätzwert basiert auf der Zählung der Eingabesignale, für die eine Zelle Gewinner war. Mit Hilfe dieser Anzahl kann eine relative Signalverteilung h_c berechnet werden. Diese gibt an, wieviele Signale die einzelne Zelle in Relation zur Gesamtzahl der Signale erhalten hat. Setzt man diese relative Verteilung ins Verhältnis zur Größe der Voronoi-Region der Zelle, so erhält

man eine genäherte Signaldichte für diese Region des Eingaberaumes. Wird diese noch geeignet normiert (siehe Abschnitt 4.4.3.2), so kann anhand eines globalen Grenzwertes entschieden werden, welche Zellen zu löschen sind.

Diese Berechnung erfolgt für jede Zelle innerhalb eines vorher festgelegten Intervalls von Eingabesignalen. Alle Zellen, deren Signaldichte unterhalb des Grenzwertes liegt, werden gelöscht. Zusätzlich werden alle Zellen und Verbindungen aus dem Netz entfernt, welche nicht mehr Bestandteil eines Simplex sind. Dadurch bleibt die Netzdimensionalität und die Netzkonsistenz erhalten.

Schritt 9 - Abbruchkriterien

Der Algorithmus verfügt über vier verschiedene Abbruchkriterien. Das unkomplizierteste Kriterium stellt das Erreichen einer vorher festgelegten Anzahl von Eingabesignalen dar. Der Adaptionsprozess läuft solange, bis diese erreicht ist. Sinnvolle Werte dafür sind beispielsweise ganzzahlige Vielfache der Punktzahl der Punktwolke. Hierdurch wird erreicht, daß jeder Punkt mit gleichem Anteil in die Adaption eingeht.

Ein weiteres einfaches Kriterium ist das Vorhandensein einer bestimmten Menge von Zellen im Netz. Der Algorithmus wird bei einer vorher festgelegten Anzahl von Zellen abgebrochen. Dieses Kriterium macht natürlich nur Sinn, wenn bei der Initialisierung des Netzes eine geringere Anzahl von Knoten vorhanden ist. Das Netz wächst dann kontinuierlich, bis die gewünschte Größe erreicht ist.

Ein komplexeres Abbruchkriterium wird durch die maximale Länge der Verbindungen zwischen den Zellen des Netzes bestimmt. Die Anpassung wird beendet, wenn die Länge aller im Netz vorhandenen Verbindungen unter einen angegebenen Grenzwert fällt. Dadurch kann indirekt die Größe der entstehenden Simplexe (Dreiecke) gesteuert werden. Das Verfahren wurde dahingehend erweitert, um die Adaption auf dieses Kriterium zu optimieren. Neue Zellen werden nicht mehr in der Nähe der größten lokalen Fehler eingefügt, sondern immer dort, wo die längste Verbindung auftritt. Damit wird sichergestellt, daß der Adaptionsprozess konvergiert und nach einer endlichen Anzahl von Schritten abbricht.

Das komplexeste der vier Kriterien ist der maximale Abstand zwischen Netz und Punktwolke. Die Lernphase wird abgeschlossen, wenn dieser unter einen vorher festgelegten Grenzwert fällt. Dadurch kann gesteuert werden, wie genau die Abbildung der Punktwolke auf das Netz erfolgen soll. Da neue Knoten in der Nähe des größten lokalen Fehlers eingefügt werden, wird dieser lokale Fehler verringert. Dadurch konvergiert die Adaption nach einer endlichen Anzahl von Schritten.

Die Überprüfung dieses Kriteriums ist relativ rechenaufwendig. Deswegen wird dieses Kriterium, im Gegensatz zu den anderen, nicht nach jedem Adaptionsschritt überprüft. Vielmehr erfolgt dies in größeren Abständen, um den Rechenaufwand gering zu halten.

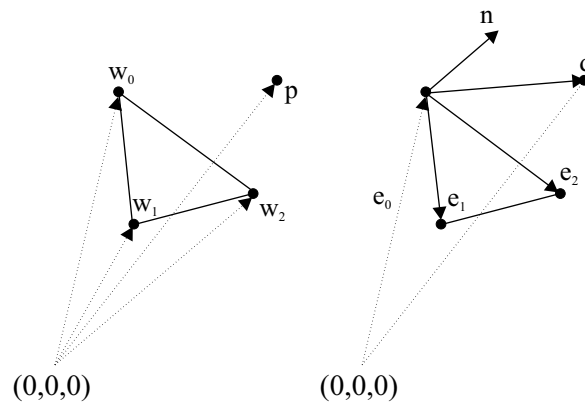


Abbildung 5.4: Abstand eines Punktes zu einem Simplex

Bei dieser Prüfung muß für jeden Punkt der Abstand zum Netz berechnet werden. Dies ist in den meisten Fällen das Lot auf den nächsten Simplex im Netz. Zuerst ist der nächstgelegene Simplex für einen Punkt zu ermitteln. Dafür wird die Annahme getroffen, daß der nächste Simplex zu einem Punkt dessen nächste Zelle als einen seiner Eckpunkte enthält. Dies wird sich natürlich nicht für alle möglichen auftretenden Fälle als richtig erweisen. Das stellt jedoch kein Problem dar. Beispielsweise kann ein Simplex gefunden werden, welcher nicht derjenige mit dem geringsten Abstand zu diesem Punkt ist. In diesem Fall ist der berechnete Abstand größer als der zum nächstgelegenen Simplex. Da das Abbruchkriterium aber mit einer oberen Schranke arbeitet, bedeutet das nur, daß die Adaption noch ein wenig länger durchgeführt wird. Die Suche kann also zunächst auf das Ermitteln der nächsten Zelle im Netz reduziert werden. Über die Verwendung eines Octtree kann dies wiederum auf die geometrische Umgebung des Punktes beschränkt werden. Ist die Zelle mit dem geringsten Abstand gefunden, dann werden alle anliegenden Simplexe überprüft. Zu jedem wird der Abstand des Punktes berechnet und der geringste ermittelte Wert verwendet.

Der Abstand eines Punktes zu einem Simplex wird in mehreren Teilschritten berechnet. Alle Simplexe des Netzes definieren über ihre Eckpunkte (Zellen mit Gewichtsvektoren) eine Ebene im dreidimensionalen Raum. Auf der linken Seite von Abbildung 5.4 sind die drei Zellen des Simplex mit ihren Gewichtsvektoren w_i sowie der Punkt p dargestellt. Auf der rechten Seite sind die Vektoren abgebildet, welche die Ebene genau definieren. Der Vektor e_0 ist der Stützvektor der Ebene und entspricht dem Gewichtsvektor w_0 . Die Vektoren e_1 und e_2 spannen die Ebene im Raum auf. Sie werden aus den Gewichtsvektoren der Zellen berechnet:

$$\vec{e}_1 = \vec{w}_1 - \vec{w}_0$$

$$\vec{e}_2 = \vec{w}_2 - \vec{w}_0$$

Der Normalenvektor n der Ebene wird über das Kreuzprodukt der beiden aufspannenden Vektoren e_1 und e_2 ermittelt.

$$\vec{n} = \vec{e}_1 \times \vec{e}_2$$

Der Punkt p kann nun als Linearkombination der Vektoren e_0 , e_1 , e_2 und n beschrieben werden.

$$\vec{p} = \vec{e}_0 + x_0 \cdot \vec{e}_1 + x_1 \cdot \vec{e}_2 + x_2 \cdot \vec{n}$$

Diese kann in geeigneter Weise umgestellt werden und ergibt ein lineares Gleichungssystem in Matrixschreibweise.

$$\begin{pmatrix} e_{1x} & e_{2x} & n_x \\ e_{1y} & e_{2y} & n_y \\ e_{1z} & e_{2z} & n_z \end{pmatrix} \cdot \begin{pmatrix} x_0 \\ x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} p_x - e_{0x} \\ p_y - e_{0y} \\ p_z - e_{0z} \end{pmatrix}$$

Der Lösungsvektor x des Gleichungssystems enthält die einzelnen Faktoren für die Linearkombination. Wenn der Normalenvektor der Ebene vor der Berechnung normiert wird, so entspricht der Bestandteil x_2 dem Lot des Punktes auf die durch den Simplex definierte Ebene. Anhand der Bestandteile x_0 und x_1 kann festgestellt werden, ob sich das Lot innerhalb des Simplex befindet und damit den korrekten Abstand definiert. Dies ist genau dann der Fall, wenn folgende Bedingungen erfüllt sind:

$$x_0 > 0 \quad \text{und} \quad x_1 > 0 \quad \text{und} \quad x_0 + x_1 \leq 1$$

Für alle anderen Fälle kann das Lot auf die Ebene nicht dem Abstand zum Simplex gleichgesetzt werden. Hier werden die Lote auf alle Kanten (Verbindungen), sowie alle Abstände zu den Eckpunkten (Zelle) des Simplex berechnet. Der Abstand zwischen Punkt und Kante ist nur zulässig, wenn das Lot auf der Kante liegt. Der geringste dieser ermittelten Werte wird als Abstand des Punktes zum Simplex verwendet.

5.3.2 Probleme und Besonderheiten

Während der Implementation und Umsetzung des im vorigen Abschnitt beschriebenen Verfahrens traten einige Probleme auf. Diese sollen hier mitsamt der dafür entwickelten Lösungen erläutert werden.

5.3.2.1 Octtree-basierte Suche

Wie im Abschnitt 5.2.4 und 5.3 beschrieben ist, wurde für die Suche der nächsten Zelle zu einem Punkt ein Octtree benutzt. Dieser enthält die Punkte der Punktwolke sowie die Gewichtsvektoren der Zellen des Netzes. Diese Daten sind jedoch unterschiedlicher Natur.

Die Daten der Punktwolke sind statisch, wogegen sich die Daten des Netzes dynamisch verändern. Die Funktionalität des Octtree mußte deshalb erweitert werden, um diesen nicht bei jedem Adaptionsschritt neu aufbauen zu müssen.

Zusätzlich zum Einsortieren wurde das Löschen und Verschieben von bereits enthaltenen Vektoren implementiert. Beim Verschieben eines Vektors beziehungsweise seines Index im Octtree, muß zunächst geprüft werden, ob sich seine Position auch zwischen den Octtree-Würfeln ändert. Liegt der verschobene Vektor innerhalb desselben Würfels, so kann der Vorgang abgebrochen werden. Im Octtree existieren nur solche Würfel, welche auch Vektoren beziehungsweise deren Indizes enthalten. Fällt der Vektor in einen anderen Würfel, so muß geprüft werden, ob dieser schon angelegt ist. Ist dies nicht der Fall, so wird er erzeugt und der Index des Vektors eingefügt. Abschließend muß noch der Index aus dem alten Würfel entfernt werden, in welchem er vor der Verschiebung enthalten war. Umfaßt dieser danach keinen weiteren Index, so kann er aus dem Octtree entfernt werden. Dadurch werden nur benötigte Würfel angelegt und nicht mehr benötigte können aus dem Octtree gelöscht werden. Dies führt zu einer sehr effizienten Ressourcennutzung. Zusätzlich werden Suchoperationen beschleunigt, da keine unbesetzten Würfel vorhanden sind.

Im Octtree beinhalten die einzelnen Würfel immer nur die jeweiligen Indizes der einsortierten dreidimensionalen Vektoren. Dies führt zu Problemen, wenn zwei Datenmengen (zum Beispiel Punktwolke und neuronales Netz) im gleichen Octtree gespeichert werden. Es wäre beispielsweise nicht mehr eindeutig bestimmbar, welcher Vektor zum Index 0 gehört. Dies könnte der erste Punkt der Punktwolke, aber auch die erste Zelle des Netzes sein. Als Lösung dieses Problems wird beim Einfügen der Indizes der Vektoren eine sehr einfache Hashfunktion verwendet. Diese berechnet den im Octtree zu speichernden Index $i_{Octtree}$ für die Ortsvektoren der Punkte und die Gewichtsvektoren der Zellen.

$$i_{Octtree} = \begin{cases} i_v & \forall v \in P \\ i_v + |P| & \forall v \in Q \end{cases}$$

Die Menge P enthält die Indizes der Punkte und die Menge Q die Indizes der Gewichtsvektoren der Zellen. $|P|$ gibt die Mächtigkeit der Menge P beziehungsweise die Anzahl der Indizes der Punktwolke an. Wird ein Index aus dem Octtree gelesen, so muß nur überprüft werden, ob er größer oder kleiner als $|P|$ ist, um festzustellen, zu welcher Datenmenge er gehört.

5.3.2.2 Zellbewegung außerhalb der Nachbarzellen

Ein weiteres Problem tritt während des Adaptionsprozesses in den Randbereichen des Netzes auf. Dort kommt es aufgrund zu großer Verschiebungen von Zellen dazu, daß sich diese außerhalb ihrer direkten topologischen Nachbarn bewegen. Dadurch entstehen geometri-

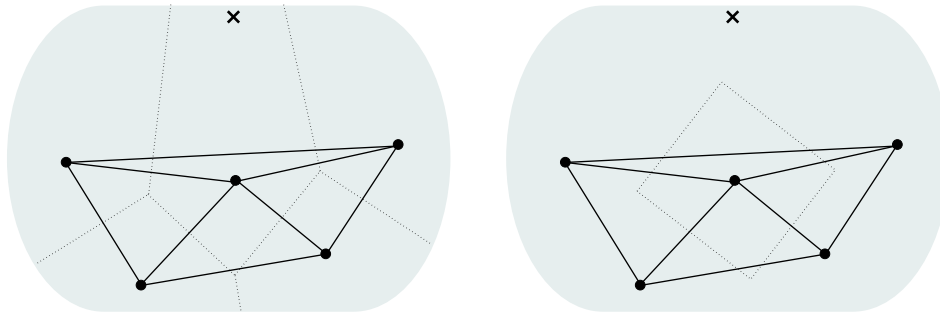


Abbildung 5.5: Zellbewegung außerhalb der Nachbarzellen

sche Inkonsistenzen innerhalb des Netzes.

Die Stärke der Bewegung einer einzelnen Zelle hängt von mehreren Faktoren ab. Der Abstand zum Eingabesignal spielt die größte Rolle. Ein zweiter Faktor ist die Lernrate, die bestimmt, um welchen Bruchteil des Abstandes zum Eingabesignal sich die Zelle bewegt. Ein möglicher Ansatz wäre also, die Lernrate für die Zellen zu verringern. Dies würde jedoch alle Zellen des Netzwerkes betreffen und zu einer schlechteren Anpassungsfähigkeit des Netzes führen.

Normalerweise wird eine Zelle nur Gewinner für diejenigen Signale, welche sich in der von ihr repräsentierten Voronoi-Region befinden. Sie kann sich deshalb also nur innerhalb dieser Region bewegen. Für Zellen im Netzinernen sind die Voronoi-Regionen geschlossen und liegen vollständig innerhalb ihrer direkten topologischen Nachbarn. Bei den Zellen in den Randbereichen des Netzes ist dies nicht der Fall. Es kann also vorkommen, daß sie Gewinner für sehr weit entfernte Eingabesignale werden, welche sich auch außerhalb ihrer topologischen Nachbarn befinden (siehe Abbildung 5.5, linke Seite). Werden diese Zellen nun in Richtung dieses Eingabesignals gezogen, so kommt es zu geometrischen Inkonsistenzen. Für diese Fälle sollte die Bewegung der Zellen in Richtung des Eingabesignals beschränkt werden.

Die Ausdehnung der Voronoi-Region kann durch die Berechnung der mittleren Kantenlänge \bar{l} der ausgehenden Verbindungen der Zelle approximiert werden (siehe Abschnitt 4.4.3). Die approximierte Region ist jedoch kleiner als die real existierende Voronoi-Region (siehe Abbildung 5.5, rechte Seite). Die Zellbewegung wird deshalb auf die approximierte Voronoi-Region beschränkt. Da sich eine Zelle im Zentrum ihrer Voronoi-Region befindet, wird der resultierende Bewegungsvektor b mit der halben Ausdehnung der Voronoi-Region verglichen. Dieser Vektor b berechnet sich aus dem Gewichtsvektor w der Zelle, dem Eingabesignal ξ , sowie der Lernrate ϵ .

$$b = \epsilon \cdot (\xi - w)$$

Ist die Länge des Bewegungsvektors $\|b\|$ größer als die halbe Ausdehnung der Voronoi-

Region, so muß dieser verkürzt werden.

$$b = \frac{\bar{l}}{2} \cdot \frac{b}{\|b\|}$$

Diese Beschränkung gilt zwar für alle Zellen, kommt aber nur bei Zellen in den Randbereichen des Netzes zur Anwendung. Die restlichen Zellen bewegen sich, wie schon erwähnt wurde, nur innerhalb ihrer Voronoi-Region und damit innerhalb ihrer topologischen Nachbarn.

Dieses Vorgehen kann sicherstellen, daß sich jede Zelle nur innerhalb ihrer approximierten Voronoi-Region bewegt. Das Entstehen von geometrischen Inkonsistenzen kann damit nicht in jedem Fall verhindert werden, tritt jedoch weitaus seltener auf. Somit steigt trotzdem die Qualität der entstehenden Netze.

5.3.2.3 Ungeordnete Zellbewegung

Im Anfangsstadium des Adaptionprozesses, beginnend mit einem einzelnen Simplex, besteht das Netz aus einigen wenigen Zellen. Sind diese aber weit über den Eingaberaum verstreut, so bilden sie große Simplexe. Das führt natürlich zu relativ großen Voronoi-Regionen und damit zu großen Bewegungen der einzelnen Zellen. Diese können sehr ungeordnet verlaufen und zu Fehlern führen. Dieser Vorgang könnte wiederum über die Verringerung der Lernrate der Zellen umgangen werden. Da die Lernrate jedoch von Anfang an festgelegt ist, würde sie für den gesamten Anpassungsprozess gelten.

Die ungeordnete Bewegung der Zellen ist aber nur am Anfang vorhanden. Deshalb wird ein neuer Parameter eingeführt. Dieser soll steuern, daß die Zellen nur auf Eingabesignale in ihrer näheren geometrischen Umgebung reagieren. Er definiert den maximalen Abstand, den ein Eingabesignal zu einer Gewinnerzelle haben darf. Ist das Eingabesignal weiter entfernt, so wird die Adaption des Gewinners abgebrochen und mit dem nächsten Eingabesignal fortgefahren. Dies führt dazu, daß sich das Netz langsamer aber auch kontrollierter ausbreitet. Ist das Netz weiter gewachsen, so werden auch Eingabesignale wieder in die Adaption einbezogen, welche vorher zu weit entfernt waren.

5.3.2.4 Unvollständige Abbildung der Punktwolke

Das Ziel der Anpassung ist es, die Punkte der Punktwolke möglichst optimal und vollständig auf das Netz abzubilden. Aufgrund der Eigenschaften des Verfahrens kann dies in den Randbereichen des Netzes jedoch nicht sichergestellt werden. Wie schon beschrieben wurde, repräsentiert jede Zelle des Netzes eine Voronoi-Region. Diese enthält alle Punkte der Punktwolke, deren Abstand zu dieser Zelle geringer ist als zu allen anderen Zellen des Net-

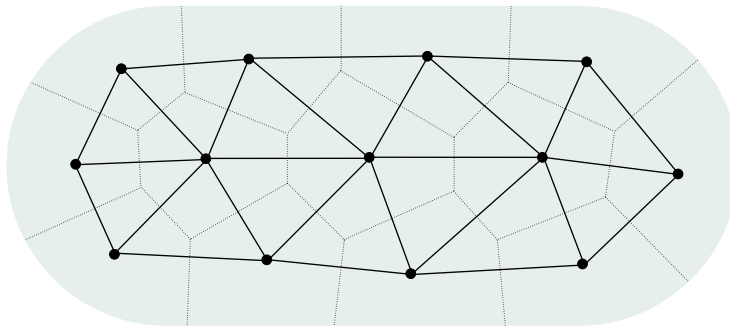


Abbildung 5.6: Unvollständige Abbildung der Punktwolke

zes. Die Zellen befinden sich dabei im Schwerpunkt der in der Voronoi-Region enthaltenen Punkte.

Für alle innerhalb des Netzes liegenden Zellen werden die Punkte der Voronoi-Region durch die anliegenden Simplexe vollständig abgebildet. Dies trifft jedoch nicht für Zellen am Rand des Netzes zu. Da diese nicht vollständig von Simplexen umgeben sind, existieren Bereiche der zugehörigen Voronoi-Region, welche außerhalb des Netzes liegen. Dies ist in Abbildung 5.6 dargestellt. Die graue Fläche markiert den Eingabebereich mit einer gleichmäßigen Signaldichte. Die Punkte stellen die Zellen des Netzes dar, welche miteinander verbunden sind. Die dünnen, gepunkteten Linien deuten die Voronoi-Regionen der Zellen an.

Diese nicht vom Netz abgedeckten Gebiete der Punktwolke treten bei allen Randzellen des Netzes auf. Dadurch bildet sich ein Randstreifen um die Randkanten des Netzes, welcher nicht vom Netz abgedeckt ist. Die Breite dieses Streifens ist von der Größe der entsprechenden Voronoi-Regionen der Randzellen abhängig. Wenn die Anzahl der Zellen wächst, so steigt auch deren Dichte im festen Eingabebereich. Dadurch verkürzen sich die Verbindungen zwischen den Zellen. Das hat wiederum zur Folge, daß auch der nicht abgebildete Rand schmaler wird. Er wird jedoch nie ganz verschwinden, sondern sich immer der Ausdehnung der jeweiligen Voronoi-Regionen anpassen.

Diese Problematik tritt einzig bei nicht geschlossenen Netzen auf, denn nur diese besitzen Randzellen. Die Zellen geschlossener Netze repräsentieren dagegen ausschließlich geschlossene Voronoi-Regionen. Somit können diese Netze den gesamten Eingabebereich abbilden.

5.3.2.5 Verbinden von Zellen

Das Verbinden von Zellen ist eine Operation, die ein konsistentes Netz als Ergebnis haben soll. Es ist recht einfach, die topologische Konsistenz des Netzes zu überprüfen. Beim Erzeugen neuer Verknüpfungen (siehe Abschnitt 5.2.2) sind immer nur maximal drei Zellen

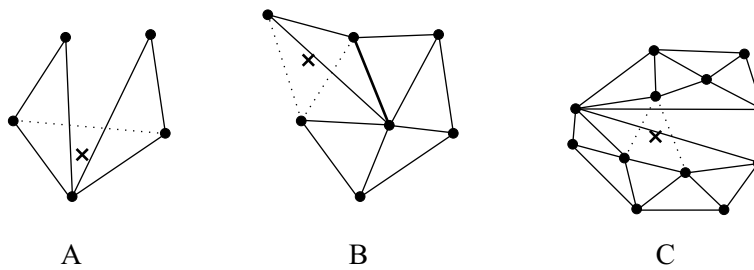


Abbildung 5.7: Geometrische Inkonsistenz beim Verbinden von Zellen

beteiligt. Das Netz kann sich also nur in der unmittelbaren topologischen Umgebung dieser Zellen verändern. Für die Überprüfung der ausgeführten Verbindungsoperation müssen alle von diesen drei Knoten ausgehenden Verbindungen untersucht werden. Ist eine von diesen an mehr als zwei Simplexen des Netzes beteiligt, so müssen die neu erzeugten Verbindungen gelöscht werden. Dadurch wird der konsistente Netzzustand wiederhergestellt.

Ein anderer Aspekt ist die Geometrie des Netzes. Diese beschreibt die Lage der einzelnen Zellen und Simplexe im Raum. Durch das Erzeugen von neuen Verbindungen kann es zur Durchdringung oder Überlagerung einzelner Simplexe kommen. Dies kann auch dann geschehen, wenn diese Operation ein topologisch konsistentes Netz zum Ergebnis hat. Die geometrische Lage der Zellen und ihrer anliegenden Simplexe muß deshalb zusätzlich untersucht werden.

In Abbildung 5.7 sind einige Fälle abgebildet, in denen das Verbinden zu einer geometrischen Inkonsistenz führen würde. Die schon bestehenden Verbindungen zwischen den verschiedenen Zellen sind als durchgängige Linien dargestellt. Das auslösende Eingabesignal ist in Form eines Kreuzes sichtbar. Die neuen Verbindungen sind mit Hilfe von gepunkteten Linien angedeutet. Die abgebildeten Fälle sind auf die für diese Problematik relevanten Teilnetze begrenzt. Sie können wie folgt beschrieben werden:

- In Beispiel A der Abbildung wird nur eine neue Verbindung erzeugt. Es ist zu sehen, daß hier keine topologische Inkonsistenz auftritt. Geometrisch betrachtet, kommt es aber zu einer Überlappung der schon existierenden Simplexe mit dem neu gebildeten Simplex.
- In Beispiel B werden zwei neue Verbindungen zwischen den gewinnenden Zellen gebildet. Auch hier kommt es zu einer Überlagerung neuer und alter Simplexe. Zusätzlich entsteht eine topologische Inkonsistenz. Die hervorgehobene Verbindung ist nun Bestandteil von drei Simplexen. Dies ist jedoch im Rahmen der Netzkonsistenz (siehe Abschnitt 3.2) nicht erlaubt.
- In Beispiel C werden auch zwei neue Verbindungen zwischen den gewinnenden Zel-

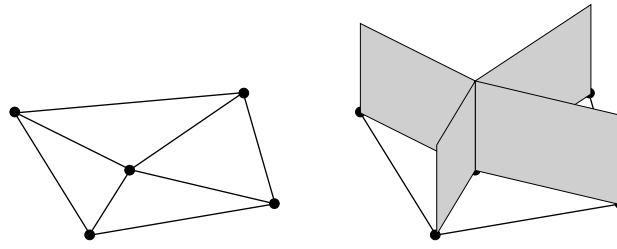


Abbildung 5.8: Raumteilung durch Slots

len erzeugt. Dadurch kommt es zu einer Überlappung mit einem schon existierenden Simplex. Dieser Fall ist jedoch komplexer als die beiden ersten Fälle.

Es müssen verschiedene Kriterien festgelegt werden, um solche und ähnlich gelagerte Fälle zu vermeiden. Diese sind bei jeder Verbindungsoperation zu prüfen, um die Netzkonsistenz sicherzustellen. Die Überprüfung der geometrischen Konsistenz des Netzes im dreidimensionalen Raum ist komplex. Die zu untersuchende Umgebung stellt nur einen kleinen Ausschnitt aus der Gesamtfläche des Netzes dar. Dieser ist meist ein fast ebenes Teilstück. Die Problemstellung wird daher als ein annähernd ebenes Problem aufgefaßt und dadurch erheblich vereinfacht.

Die Lage des Eingabesignals bezüglich des neu entstehenden Simplex ist das erste Kriterium. Dafür wird das Eingabesignal auf die durch die drei gewinnenden Zellen definierte Ebene projiziert. Anschließend wird geprüft, ob sich die Projektion innerhalb des neuen Simplex befindet. Die Berechnung ist in Abschnitt 5.3.1 und Abbildung 5.4 beschrieben. Dies ist ein notwendiges Kriterium, da Zellen nur in solchen Bereichen im Eingaberaum erzeugt werden sollen, in denen auch Punkte vorhanden sind.

Als nächstes muß geprüft werden, ob es Überschneidungen zwischen dem neuen Simplex und schon vorhandenen Simplexen gibt. Dafür wird der Raum um eine Zelle in sogenannte *Slots* unterteilt. Diese werden durch eine Zelle und ihre ausgehenden Verknüpfungen definiert. Sie werden von Ebenen getrennt, welche durch die einzelnen Verbindungen laufen (siehe Abbildung 5.8). Ein Slot ist frei, wenn er durch drei Zellen definiert wird, die keinen Simplex bilden. Das heißt, in diesem Bereich befindet sich noch kein Simplex.

Zuerst müssen für die ausgehenden Verbindungen der Gewinnerzellen geprüft werden, ob sie innerhalb des durch den neuen Simplex definierten Slot liegen. Dazu wird der Endpunkt einer ausgehenden Verbindung, wie schon in Abschnitt 5.3.1 beschrieben ist, auf die durch den neuen Simplex verlaufende Ebene projiziert. Sind die Bestandteile x_0 und x_1 des Lösungsvektors positiv, so liegt diese Verbindung innerhalb des Slots. Das bedeutet, daß dieser Bereich teilweise durch einen schon bestehenden Simplex überdeckt ist. Für die Lage der Slots und des neuen Simplex gibt es drei verschiedene Fälle, welche in Abbildung

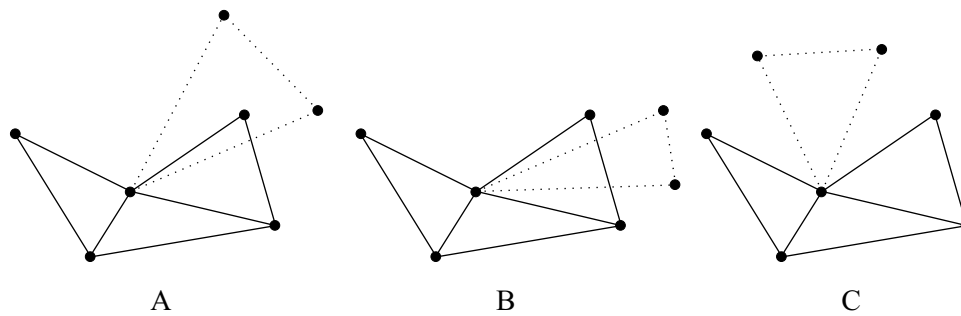


Abbildung 5.9: Simplex - Slot - Überschneidung

5.9 dargestellt sind. Die schon existierenden Verbindungen und Simplexe sind mit durchgezogenen Linien dargestellt. Die neu zu erzeugenden Verknüpfungen beziehungsweise der neue Simplex sind durch gepunktete Linien angedeutet:

- In Fall A liegt mindestens eine schon bestehende Verbindung innerhalb des durch den neuen Simplex definierten Slots. Es können auch durchaus mehrere bestehende Verbindungen innerhalb dieses Slots liegen. Tritt der Fall A ein, so dürfen keine neuen Verbindungen erzeugt werden.
- Wie in Fall B zu sehen ist, kann es auch zu Überschneidungen kommen, wenn keine Verbindungen innerhalb des Slots des neuen Simplex liegen. Hier muß geprüft werden, ob sich eine der neu zu erzeugenden Verknüpfungen innerhalb eines Slots befinden, der durch einen schon bestehenden Simplex definiert wird. Auch in diesem Fall muß die Erzeugung der neuen Verbindungen abgebrochen werden.
- Der Fall C veranschaulicht die einzige zulässige Lage des neuen Simplex. Dieser liegt hier vollständig innerhalb eines freien Slots. Nur wenn dies für alle drei gewinnenden Zellen der Fall ist, können die neuen Verknüpfungen erzeugt werden.

Abschließend ist noch zu prüfen, ob auch die topologische Konsistenz erhalten geblieben ist. Dazu wird jede der drei beteiligten Zellen untersucht. Alle von diesen Zellen ausgehenden Verbindungen müssen mindestens an einem oder höchstens an zwei Simplexen beteiligt sein. Ist diese Bedingung nicht erfüllt, so müssen die neu erzeugten Verknüpfungen wieder gelöscht werden. Damit wird dann der konsistente Ausgangszustand des Netzes wieder hergestellt.

Anhand der hier aufgestellten Kriterien können Fälle, wie die in Abbildung 5.7 dargestellten Beispiele A und B, vermieden werden. Diese können mittels lokaler Informationen wie anliegende Simplexe oder topologische Nachbarschaften untersucht und entschieden werden. Der Fall C der Abbildung 5.7 ist jedoch nicht mit lokalen Informationen zu lösen.

5.4 Ergebnisse

5.4.1 Eine ebene Punktwolke

Dieses Beispiel zeigt eine Punktwolke, deren Punkte in einer Ebene verteilt sind. Sie besteht aus zwei voneinander vollständig getrennten Gebieten. Im Folgenden ist das adaptierte Netz in verschiedenen Wachstumsstadien abgebildet. Die Punkte der Punktwolke sind schwarz, die Zellen des Netzes sind als hellblaue Quadrate und die Verbindungen zwischen den Zellen sind als gelbe Linien dargestellt. Gelöschte Zellen des Netzes werden an ihrer letzten Position durch violette Quadrate repräsentiert.

Die in Abbildung 5.10 dargestellten Teilschritte zeigen den Adaptionprozess nach der Initialisierung, nach 1000, nach 4000, nach 10000, nach 30000 und nach 50000 Eingabesignalen. Die für die Adaption verwendeten Parameter sind in Tabelle 5.1 aufgeführt. Der Adaptionprozess wurde jeweils nach der Anzahl der angegebenen Eingabesignale unterbrochen und später mit gleichen Parametern weitergeführt.

Anzahl Signale nachdem Knoten eingefügt wird	200
Anzahl Signale nachdem Knoten gelöscht wird	5000
Lernrate ϵ_b für Gewinner	0.02
Lernrate ϵ_n für Umgebung	0.006
Faktor für Fehlersenkung der Umgebung	0.2
Faktor für Fehlersenkung aller Zellen	0.05
maximaler Abstand Zelle - Eingabesignal	—

Tabelle 5.1: Parameter für die Triangulation der ebenen Punktwolke

Wie in den ersten vier dargestellten Schritten deutlich zu sehen ist, breitet sich das Netz zunächst über die beiden separaten Gebiete aus. Dabei überspannt es auch Bereiche im Eingaberaum, in denen keine Punkte vorhanden sind. Von dort werden deshalb auch keine Eingabesignale generiert. Zellen, welche sich in diesen Bereichen befinden, werden nicht Gewinner und deswegen nicht direkt adaptiert. Sie können höchstens im Rahmen der Adaption der direkten topologischen Umgebung einer anderen gewinnenden Zelle bewegt werden.

Im fünften Stadium sind drei der vier Zellen, welche sich in Bereichen ohne Eingabesignale befanden, gelöscht. Die vierte Zelle befindet sich scheinbar noch nahe genug an einem der separaten Gebiete der Punktwolke, um noch Eingabesignale zu empfangen. Diese wird dann langsam zu den anderen Zellen des Teilnetzes gezogen. Durch das Löschen der Zellen sind an beiden Teilnetzen Einschnitte entstanden.

Im letzten Teilbild haben sich die beiden Teilnetze weiter über die beiden Teilgebiete der Punktwolke ausgebreitet. Es ist zu erkennen, daß sich die Einschnitte in den beiden Netzen

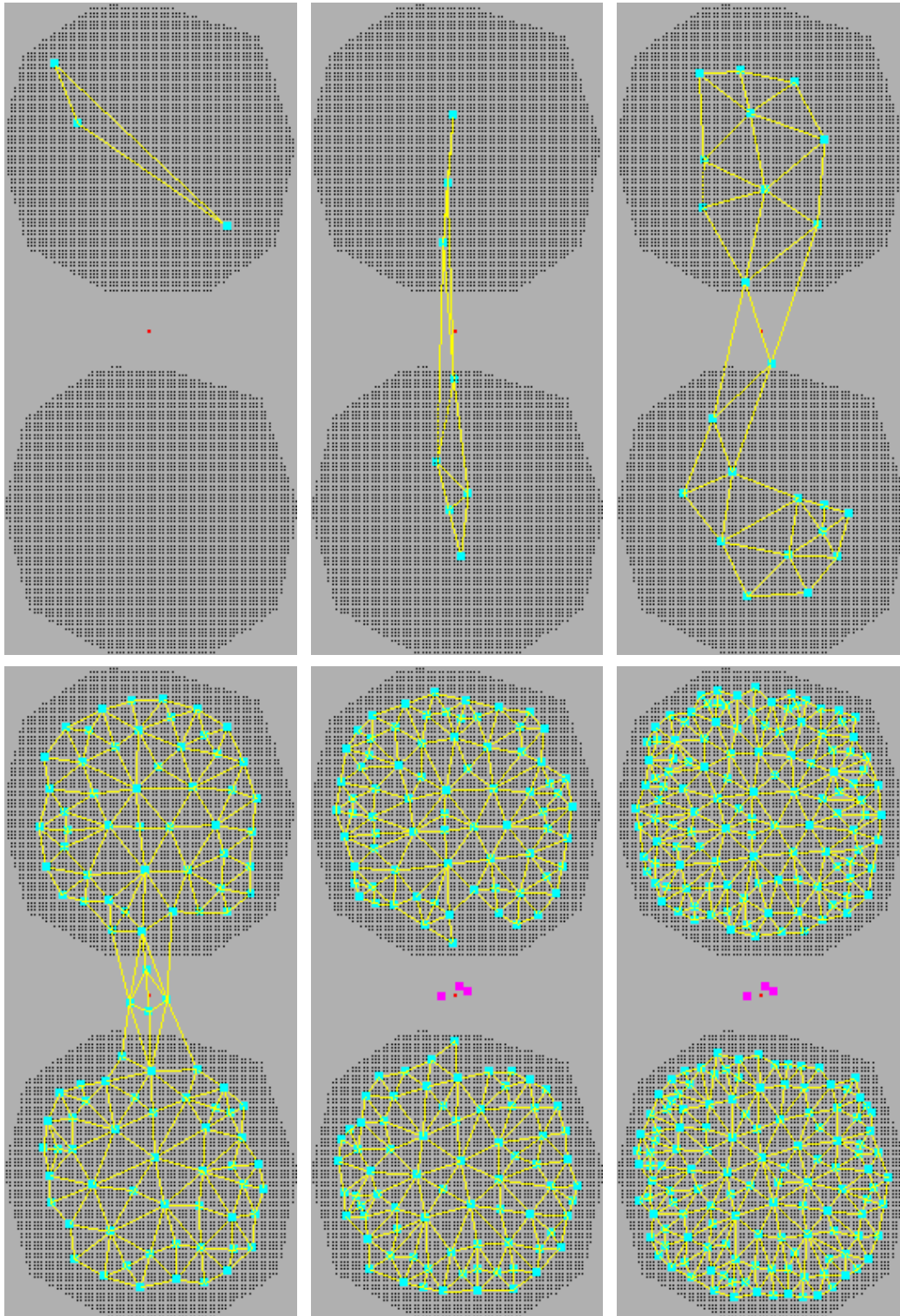


Abbildung 5.10: Triangulation einer ebenen Punktwolke mit separaten Teilgebieten

wieder geschlossen haben.

Weiterhin wird bei diesem Beispiel die in Abschnitt 5.3.2 beschriebene Problematik der unvollständigen Abdeckung der Punktwolke durch das Netz ersichtlich. Da die beiden Teilnetze nicht geschlossen sind, bleiben Teile der Punktwolke an den Randbereichen des Netzes frei.

5.4.2 Ein gescannter Zahnstumpf

Für dieses Beispiel wurde die Punktwolke eines gescannten Zahnstumpfes verwendet (siehe Abbildung 5.11). Die Punktwolke, die Zellen des Netzes und die Verbindungen zwischen den Zellen sind mit den gleichen Farben wie in Abbildung 5.10 dargestellt. Es sind jeweils sechs verschiedene Stadien des Adaptionsprozesses aus der Vorder- sowie aus der Seitenansicht abgebildet.

Die verschiedenen Teilschritte in der Abbildung 5.11 zeigen die Adaption nach der Initialisierung, nach 30000, nach 60000, nach 90000, nach 120000 sowie nach 180000 Eingabesignalen. Die dabei verwendeten Parameter sind in Tabelle 5.2 aufgelistet.

Anzahl Signale nachdem Knoten eingefügt wird	200
Anzahl Signale nachdem Knoten gelöscht wird	5000
Lernrate ϵ_b für Gewinner	0.02
Lernrate ϵ_n für Umgebung	0.005
Faktor für Fehlersenkung der Umgebung	0.5
Faktor für Fehlersenkung aller Zellen	0.1
maximaler Abstand Zelle - Eingabesignal	1mm

Tabelle 5.2: Parameter für die Triangulation des gescannten Zahnstumpfes

In den ersten drei Teilschritten beginnt das Netz, sich von einer Seite der Punktwolke auszuweiten. In den nächsten Schritten wächst das Netz seitlich und nach oben an der Punktwolke entlang. Dies geschieht solange, bis sich die Ränder des Netzes an der Rückseite des Zahns treffen und sich dort verbinden. Dadurch wird die Punktwolke langsam umschlossen.

Auf der linken Seite der Abbildung 5.12 ist nochmal das Ergebnis nach 180000 Eingabesignalen dargestellt. Die rechte Seite zeigt das aus dem adaptierten Netz erzeugte Dreiecksnetz. Auch hier ist wieder der frei bleibende Randbereich der Punktwolke deutlich zu erkennen.

Aus der Anzahl der für die Adaption benötigten Eingabesignale ist ersichtlich, daß der Prozess relativ langsam verläuft. Dies ist auf den Parameter zurückzuführen, welcher in der Tabelle 5.2 an letzter Stelle steht. Dieser besagt, daß die einzelnen Zellen nur auf Eingabesignale reagieren, wenn diese maximal einen Millimeter entfernt sind. Das führt dazu, daß in den Anfangsstadien der Adaption auf viele generierten Eingabesignale keine Adaption erfolgt. Dadurch läuft die Bewegung der Zellen geordneter, aber auch langsamer ab.

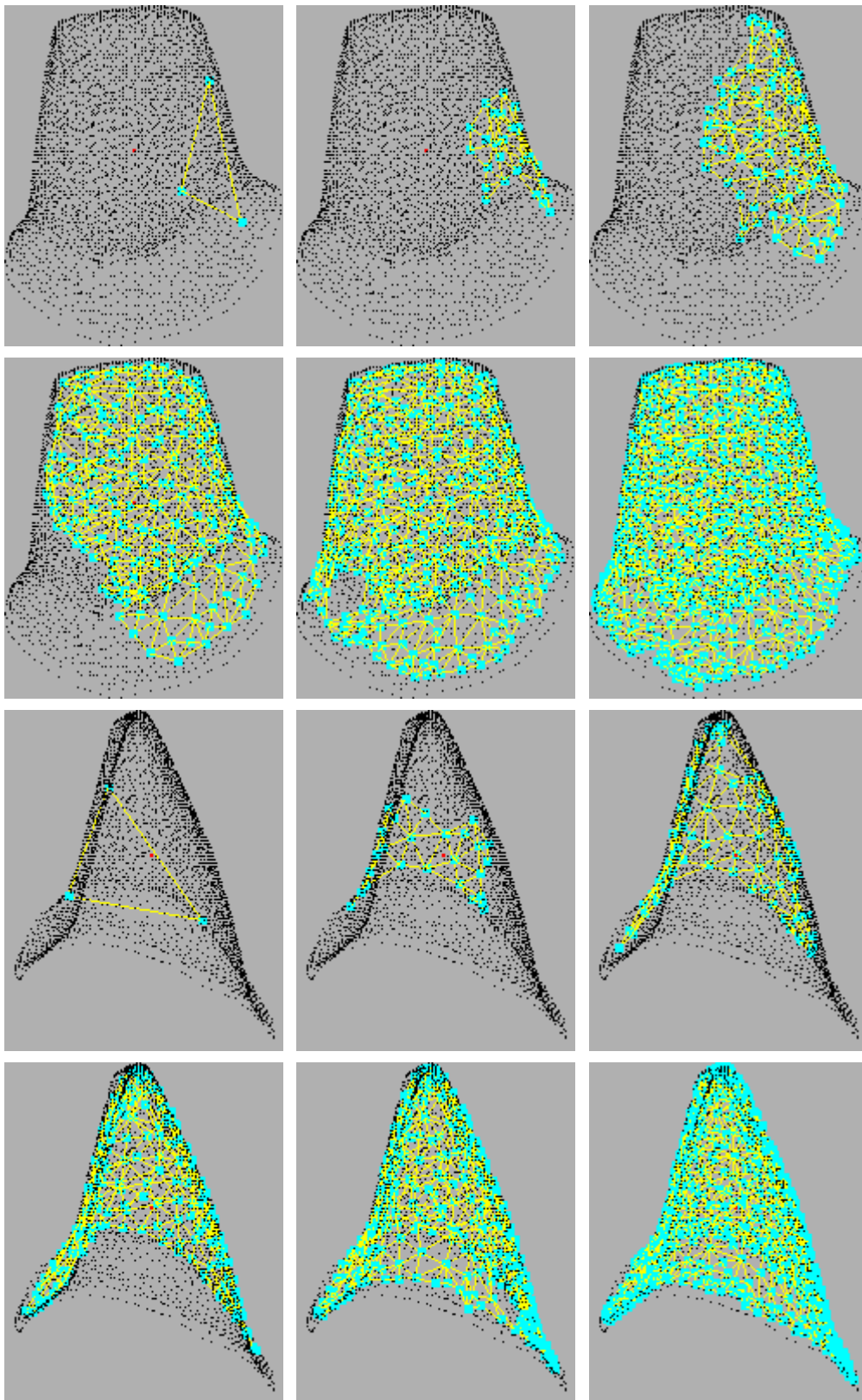


Abbildung 5.11: Triangulation eines gescannten Zahnstumpfes (Vorder- und Seitenansicht)

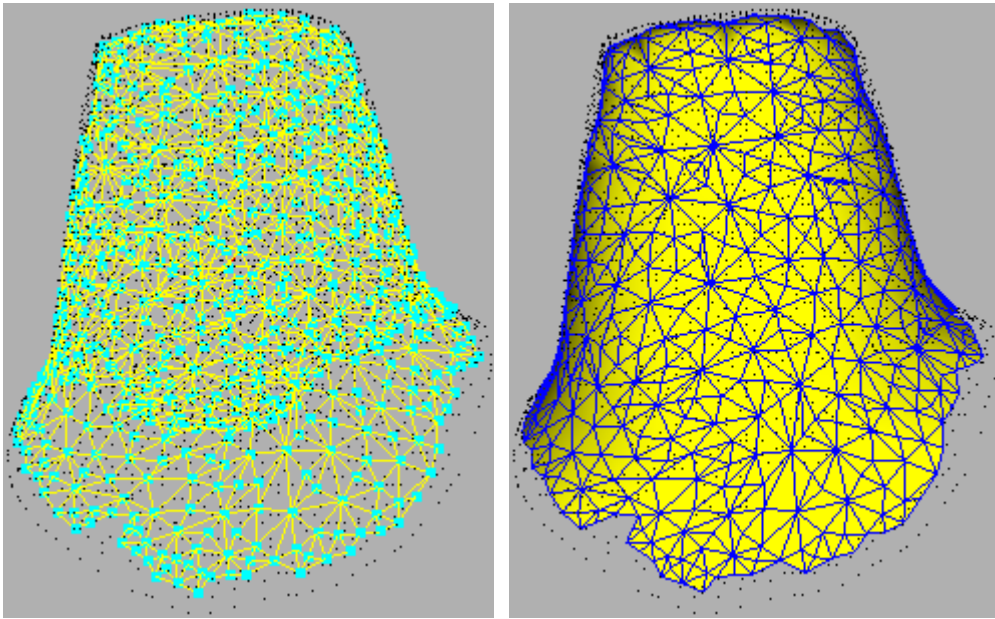


Abbildung 5.12: Triangulation eines gescannten Zahnstumpfes (Endstadium)

5.4.3 Adaption eines vorgefertigten Netzes

Im letzten Beispiel wird die Adaption eines vorgefertigten Dreiecksnetzes die Punktwolke eines gescannten Zahns dokumentiert. Die in Abbildung 5.13 dargestellten Teilschritte zeigen den Fortschritt nach der Initialisierung des Netzes, nach 5000, nach 10000, nach 25000, nach 55000 und nach 75000 Eingabesignalen. Die für den Adaptionprozess verwendeten Parameter sind in Tabelle 5.3 aufgelistet.

Anzahl Signale nachdem Knoten eingefügt wird	–
Anzahl Signale nachdem Knoten gelöscht wird	–
Lernrate ϵ_b für Gewinner	0.02
Lernrate ϵ_n für Umgebung	0.01
Faktor für Fehlersenkung der Umgebung	0.5
Faktor für Fehlersenkung aller Zellen	0.1
maximaler Abstand Zelle - Eingabesignal	–

Tabelle 5.3: Parameter für die Adaption des vorgefertigten Dreiecksnetzes

Die Topologie des Netzes wird während der Adaption nicht verändert. Deshalb wurden die Parameter so gewählt, daß weder neue Knoten eingefügt, noch bestehende Knoten gelöscht werden. Auch die Beschränkung des maximalen Abstandes zum korrespondierenden Eingabesignal wurde aufgehoben, um auch weiter entfernte Knoten an die Punktwolke zu adaptieren.

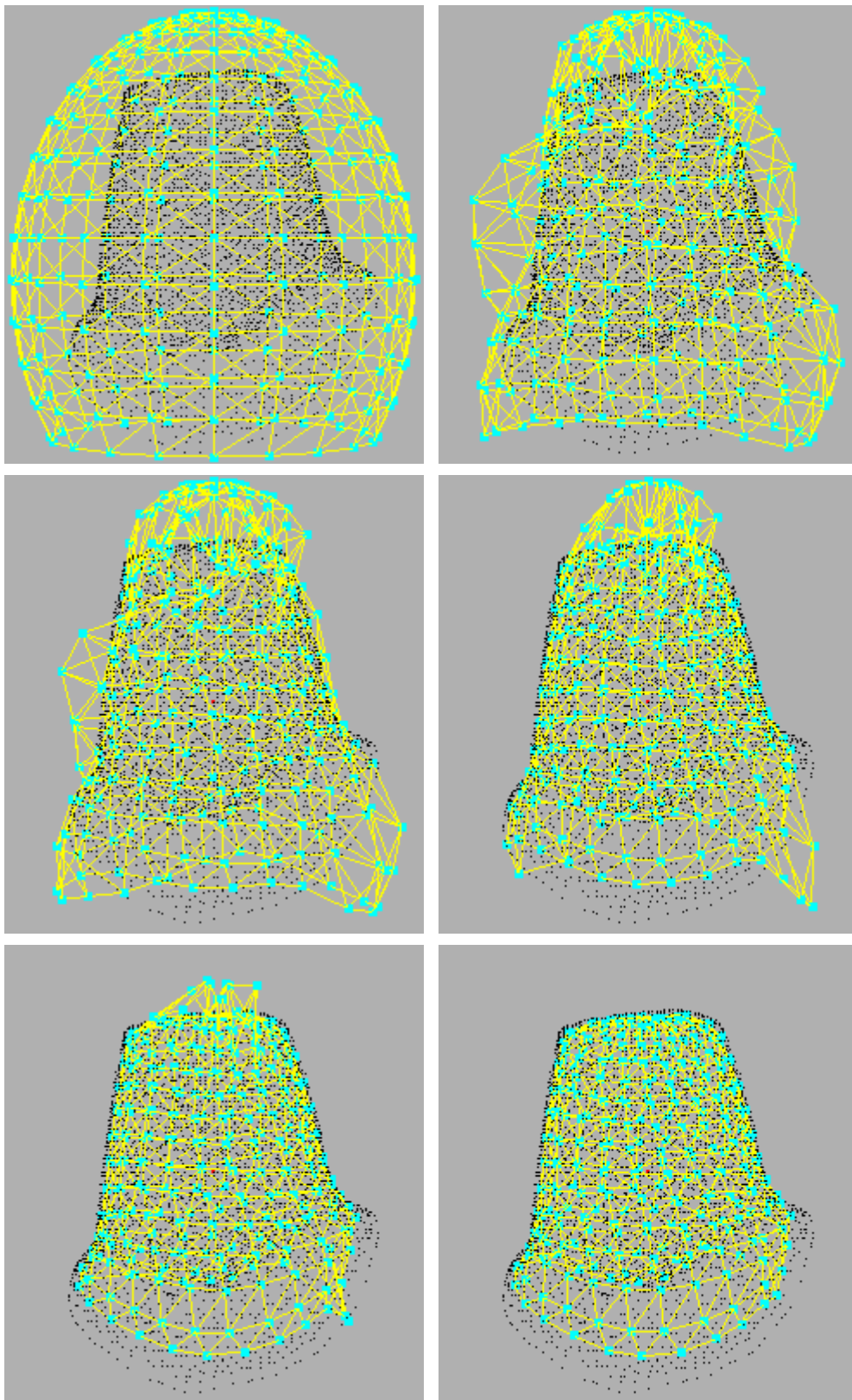


Abbildung 5.13: Adaption eines vorgefertigten Dreiecksnetzes (Teilschritte)

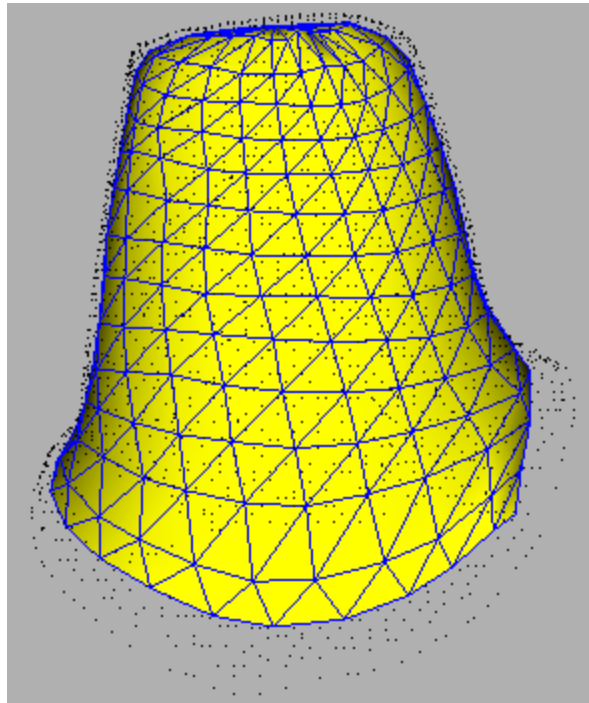


Abbildung 5.14: Adaption eines vorgefertigten Dreiecksnetzes (Endstadium)

In den einzelnen Stadien ist deutlich zu erkennen, daß die weiter von der Punktwolke entfernten Zellen des Netzes am langsamsten adaptiert werden. Durch ihren größeren Abstand werden sie in den Anfangsstadien auch sehr selten Gewinner für ein Eingabesignal. Dadurch können sie sich nur im Rahmen der Adaption der topologischen Umgebung einer anderen Zelle in Richtung der Punktwolke bewegen. Da die Lernrate für diese umgebenden Zellen aber geringer als für die gewinnende Zelle ist, bewegen sie sich auch langsamer.

Wie auch bei den beiden vorangegangenen Beispielen wird die Punktwolke nicht vollständig auf das Netz abgebildet. Da auch dieses Netz nicht geschlossen ist, bildet sich an den Randbereichen ein Streifen der Punktwolke, welcher von Zellen frei bleibt.

5.5 Fazit und Ausblick

Im Rahmen dieser Diplomarbeit wurde ein adaptives Triangulationsverfahren entwickelt. Als Basis diente das neuronale Netzmodell der *Wachsenden Zellstrukturen*. Dieses wurde in geeigneter Weise erweitert und angepaßt, um die geforderten Eigenschaften zu erhalten.

Die an das Verfahren gestellten Anforderungen konnten zu einem Großteil erfüllt werden. Die Sicherstellung der geometrischen Konsistenz eines Netzes ist nicht vollständig gelöst. Es müßten weitere globale Kriterien herangezogen werden, um eine Entscheidung über das geometrisch konsistente Erzeugen neuer Verbindungen zu treffen. Dies würde allerdings einen großen Rechenaufwand bedeuten, da diese Entscheidung bei der Adaption jeder einzelnen Zelle getroffen werden muß. Da solche Spezialfälle äußerst selten auftreten, kann aber nicht von einer größeren Schwäche des Algorithmus gesprochen werden.

Das entwickelte Verfahren hat die Fähigkeit, beliebige Punktwolken in flexibler Art und Weise zu triangulieren. Es kann genutzt werden, um ausgehend von einem einzelnen Simplex, ein vollständiges Dreiecksnetz aus einer Punktwolke zu erzeugen. Es bietet aber auch die Möglichkeit, beliebig geformte vorgefertigte Dreiecksnetze an Punktwolken zu adaptieren. Ein großer Vorteil dabei ist, daß Dreiecksnetze nach ihrer primären Generierung jederzeit an ähnliche oder geänderte Punktwolken angepaßt werden können. Außerdem sind diese Netze auch später noch an steigende Genauigkeitsanforderungen adaptierbar. Vorstellbar ist zum Beispiel eine Netzrepräsentation für die Objektklasse *Gesicht*, die iterativ an unterschiedliche Scandaten von verschiedenen Gesichtern angepaßt wird. Ebenso wäre es möglich, die Anpassung nur in definierten Bereichen (zum Beispiel der Nase) zuzulassen. In anderen Anwendungen ist beispielsweise auch das Erzeugen von Zwischenformen zweier Datensätze (Morphing-Effekt) von Interesse.

Dieses Verfahren kann als eine prototypische Lösung betrachtet werden. Daraus ergeben sich Möglichkeiten, den Algorithmus weiter zu verbessern und zu optimieren. Denkbare weitere Entwicklungen können in Richtung der hundertprozentigen geometrischen Konsistenz der erzeugten Dreiecksnetze gehen. Es sind aber auch durchaus noch Geschwindigkeitsoptimierungen möglich. Diese könnten beispielsweise durch Ausnutzung der natürlichen Parallelität der neuronalen Netze erreicht werden.

Eine weitere sehr interessante Erweiterung wäre die Erzeugung einer krümmungsabhängigen Topologie der Netze. In relativ ebenen Bereichen der Punktwolke werden nur einige wenige Dreiecke gebildet. In Teilen der Punktwolke, in denen viele Informationen beziehungsweise relativ viele Details enthalten sind, ergibt sich eine entsprechend höhere Dreiecksdichte.

Es sind viele mögliche Erweiterungen vorstellbar. Die entwickelte Software ist problemlos zu erweitern, so daß vielfältige Anwendungsbereiche erschlossen werden können.

Literaturverzeichnis

- Rüdiger Brause. *Neuronale Netze : Eine Einführung in die Neuroinformatik*. Teubner, Stuttgart, 1991.
- Bernd Breuckmann. *Bildverarbeitung und optische Meßtechnik in der industriellen Praxis*. Franzis, München, 1993.
- Peter Coveney and Roger Highfield. *Anti-Chaos : Der Pfeil der Zeit in der Selbstorganisation des Lebens*. Rowohlt, Reinbek bei Hamburg, first edition, 1992.
- Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry : Algorithms and Applications*. Springer, Berlin, second edition, 1997.
- Bernd Fritzke. Growing cell structures - a self-organizing network unsupervised und supervised learning. Technical report, International Computer Science Institute Berkeley, 1993.
- Bernd Fritzke. Some competitive learning methods. Technical report, Institut for Neural Computation Ruhr-Universität Bochum, 1997.
- Reinhard Klette, Andreas Koschan, and Karsten Schlüns. *Computer Vision : Räumliche Informationen aus digitalen Bildern*. Vieweg, Braunschweig/Wiesbaden, 1996.
- Teuvo Kohonen. *Self-Organizing Maps*. Springer, Berlin, third edition, 2001.
- Thomas Luhmann. *Nahbereichsphotogrammetrie : Grundlagen, Methoden und Anwendungen*. Wichmann, Heidelberg, 2000.
- Thomas Martinetz and Klaus Schulten. A neural gas network learns topologies. Technical report, Beckman- Institute and Department of Physics University of Illinois at Urbana-Champaign, 1991.
- Dan W. Patterson. *Künstliche neuronale Netze : Das Lehrbuch*. Prentice Hall, München, 1996.

- Lothar Paul. Competitive Mesh Adaption - alternative Modellierungsalgorithmen für komplexe Freiformflächen aus Punktwolken. In *GFaI Jahresbericht*, 2000.
- William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery. *Numerical Recipes in C : The Art of Scientific Computing*. Cambridge University Press, Cambridge, second edition, 1992.
- Gerhard Rigoll. *Neuronale Netze : Eine Einführung für Ingenieure, Informatiker und Naturwissenschaftler*. expert, Renningen-Malmsheim, 1994.
- Helge Ritter, Thomas Martinetz, and Klaus Schulten. *Neuronale Netze : Eine Einführung in die Neuroinformatik selbstorganisierender Netzwerke*. Addison-Wesley, Bonn, 1991.
- Eberhard Schöneburg, Nikolaus Hansen, and Andreas Gawelczyk. *Neuronale Netzwerke : Einführung, Überblick und Anwendungsmöglichkeiten*. Markt&Technik, Haar bei München, 1990.
- Geoff Wyvill, Craig McPheers, and Brian Wyvill. Data structure for soft objects. In *The visual computer*, volume 2, pages 227 – 234, 1986.

Index

A

Aktivierungsfunktion 34
 Ausgabefunktion 34
 Ausgabeschicht 35
 Axon 31

B

Belusow-Shabotinski-Reaktion 30

C

chemische Uhr 30–31
 Competitive Hebbian Learning .. 49–50
 Competitive Mesh Adaption 23–24

D

Delaunay-Triangulation 17–21
 Dendriten 31

E

Eingabeschicht 35

F

Feedback-Netz 35
 Feedforward-Netz 35

G

Geometrie
 konsistente 17
 Gray-Code 10–11
 Growing Cell Structures 44–48
 Growing Neural Gas 50–53

H

Hard-Limiter 34
 Hebb'sche Lernregel 39

I

Isofläche 21

K

Konvexität 17

L

Lernen 36, 38–41
 überwachtes 38, 41
 unüberwachtes 39, 41
 Lernrate 40
 konstante 40
 variable 40
 exponentiell sinkende 41
 k-means 40
 Lichtpunktstereoanalyse 6–7
 Lichtpunkttechnik 5–6
 Lichtschnitttechnik
 binärcodierte 10–11
 einfache 7–8
 farbcodierte 12–13

M

Marching-Cubes 21–22
 Meßbrauschen 14
 Moiré-Muster 11
 Moiré-Technik 11–12

N

nachbarschaftserhaltend *siehe*
 topologieerhaltend
 Neural Gas 48–49
 Neuron
 biologisches 31

künstliches *siehe* Zelle

O

Octtree 14, 22

Octtree-Würfel 14

P

Propagierungsfunktion 33

Punktwolke 13

R

Rayleigh-Benárd-Instabilität 29–30

S

Schwellwertfunktion 34

selbstorganisierende Merkmalskarte *siehe*
Self-Organizing Feature Map

Self-Organizing Feature Map 42–43

Sigmoid-Funktion 34

Simplex 44

SOFM *siehe* Self-Organizing Feature
Map

Stereoanalyse mit Farbe 13

Streifenprojektion 8–10
 dynamische 9–10
 statische 8

T

Tiefenkarte 5, 8, 9, 14

Topologie

 konsistente 16

topologieerhaltend 42

Triangulation 17–27

V

verdeckte Schicht 35

Vermessung 3–13

Vertex 16

Vertizes *siehe* Vertex

Voronoi-Diagramm 18–19

Voronoi-Polygon 18–19

Voronoi-Region 44

W

Wachsende Zellstrukturen *siehe* Growing
Cell Structures

winner-takes-all 34, 40

winner-takes-most 49

Z

Zelle 32–40

Zellkern 31

Anhang A

Programmablaufplan

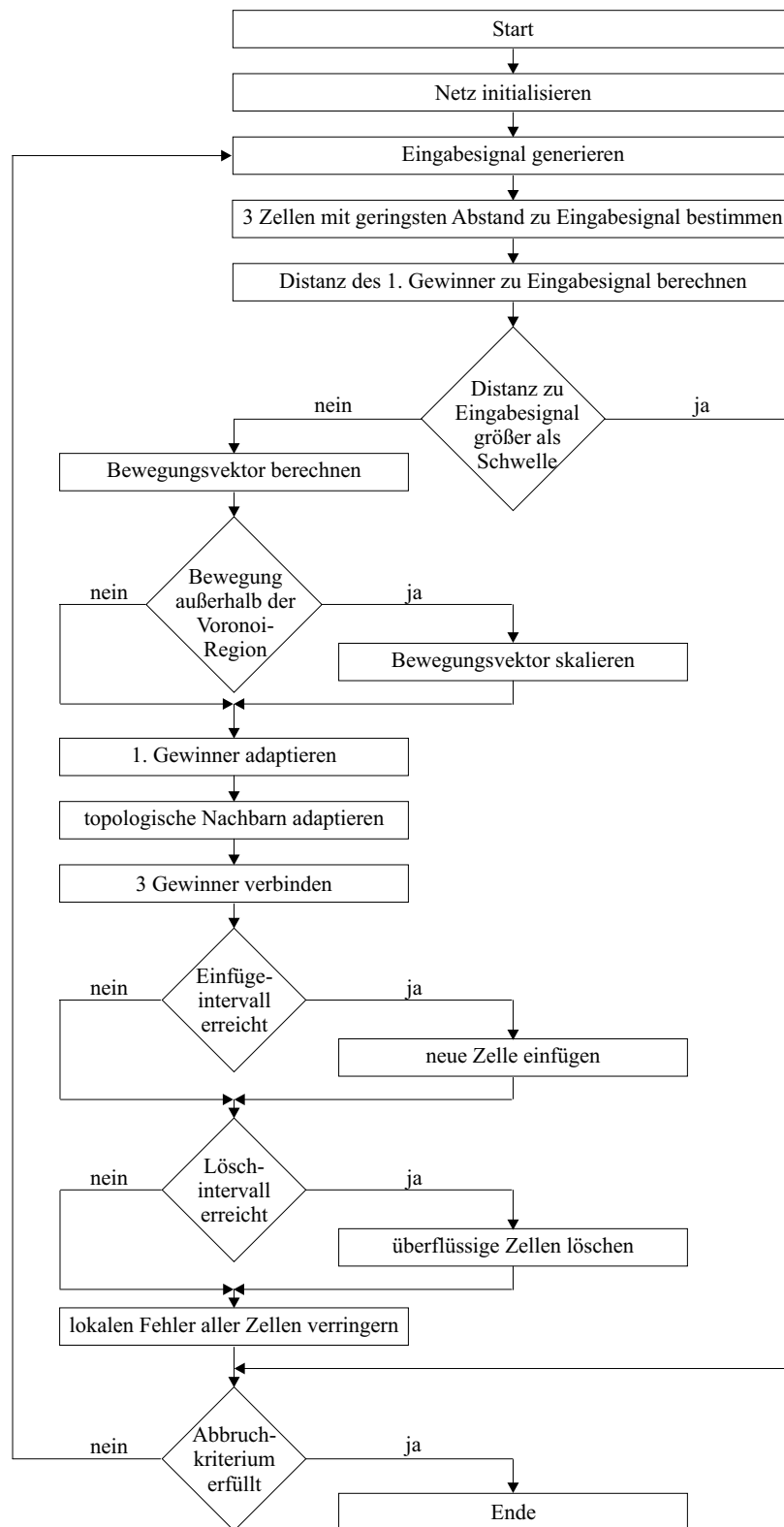


Abbildung A.1: Programmablaufplan des adaptiven Triangulationsverfahrens

Anhang B

Eigenständigkeitserklärung

Hiermit versichere ich, daß ich die vorliegende Diplomarbeit selbstständig und nur unter Verwendung der angegebenen Quellen und Hilfsmittel verfaßt habe. Die Arbeit wurde bisher in gleicher oder ähnlicher Form keiner anderen Prüfungsbehörde vorgelegt.

Berlin, den 4. November 2002

Jan Hambrecht